

How To Install the Apache Web Server on Debian 9



Posted September 5, 2018  47.7k [APACHE](#) [DEBIAN 9](#)

By: Justin Ellingwood By: Kathleen Juell By: Hanif Jetha

Not using **Debian 9**? Choose a different version:

Ubuntu 18.04 [>](#)

Ubuntu 16.04 [>](#)

Automated: Docker [request](#)

Introduction

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features including dynamically loadable modules, robust media support, and extensive integration with other popular software.

In this guide, we'll explain how to install an Apache web server on your Debian 9 server.

Prerequisites

Before you begin this guide, you should have a regular, non-root user with sudo privileges configured on your server. Additionally, you will need to enable a basic firewall to block non-essential ports. You can learn how to configure a regular user account and set up a firewall for your server by following our [initial server setup guide for Debian 9](#).

When you have an account available, log in as your non-root user to begin.

Step 1 — Installing Apache

Apache is available within Debian's default software repositories, making it possible to install it using conventional package management tools.

Let's begin by updating the local package index to reflect the latest upstream changes:

```
$ sudo apt update
```

Then, install the `apache2` package:

```
$ sudo apt install apache2
```

After confirming the installation, `apt` will install Apache and all required dependencies.

Step 2 — Adjusting the Firewall

Before testing Apache, it's necessary to modify the firewall settings to allow outside access to the default web ports. Assuming that you followed the instructions in the prerequisites, you should have a UFW firewall configured to restrict access to your server.

During installation, Apache registers itself with UFW to provide a few application profiles that can be used to enable or disable access to Apache through the firewall.

List the `ufw` application profiles by typing:

```
$ sudo ufw app list
```

You will see a list of the application profiles:

Output

```
Available applications:
```

```
  AIM
  Bonjour
  CIFS
  . . .
  WWW
  WWW Cache
  WWW Full
  WWW Secure
  . . .
```

The Apache profiles begin with `WWW`:

- **WWW:** This profile opens only port 80 (normal, unencrypted web traffic)

- **WWW Cache:** This profile opens only port 8080 (sometimes used for caching and web proxies)
- **WWW Full:** This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)
- **WWW Secure:** This profile opens only port 443 (TLS/SSL encrypted traffic)

It is recommended that you enable the most restrictive profile that will still allow the traffic you've configured. Since we haven't configured SSL for our server yet in this guide, we will only need to allow traffic on port 80:

```
$ sudo ufw allow 'WWW'
```

You can verify the change by typing:

```
$ sudo ufw status
```

You should see HTTP traffic allowed in the displayed output:

Output

```
Status: active
```

To	Action	From
--	-----	----
OpenSSH	ALLOW	Anywhere
WWW	ALLOW	Anywhere
OpenSSH (v6)	ALLOW	Anywhere (v6)
WWW (v6)	ALLOW	Anywhere (v6)

As you can see, the profile has been activated to allow access to the web server.

Step 3 — Checking your Web Server

At the end of the installation process, Debian 9 starts Apache. The web server should already be up and running.

Check with the `systemd` init system to make sure the service is running by typing:

```
$ sudo systemctl status apache2
```

Output

```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: SCROLL TO TOP
   Active: active (running) since Wed 2018-09-05 19:21:48 UTC; 13min ago
     Main PID: 12842 (sshd)
     CGroup: /system.slice/apache2.service
```

Main PID: 12849 (apache2)

CGroup: /system.slice/apache2.service

└─12849 /usr/sbin/apache2 -k start

└─12850 /usr/sbin/apache2 -k start

└─12852 /usr/sbin/apache2 -k start

Sep 05 19:21:48 apache systemd[1]: Starting The Apache HTTP Server...

Sep 05 19:21:48 apache systemd[1]: Started The Apache HTTP Server.

As you can see from this output, the service appears to have started successfully. However, the best way to test this is to request a page from Apache.

You can access the default Apache landing page to confirm that the software is running properly through your IP address. If you do not know your server's IP address, you can get it a few different ways from the command line.

Try typing this at your server's command prompt:

```
$ hostname -I
```

You will get back a few addresses separated by spaces. You can try each in your web browser to see if they work.

An alternative is using the `curl` tool, which should give you your public IP address as seen from another location on the internet.

First, install `curl` using `apt`:

```
$ sudo apt install curl
```

Then, use `curl` to retrieve `icanhazip.com` using IPv4:

```
$ curl -4 icanhazip.com
```

When you have your server's IP address, enter it into your browser's address bar:

`http://your_server_ip`

You should see the default Debian 9 Apache web page:



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in [/usr/share/doc/apache2/README.Debian.gz](#)**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

This page indicates that Apache is working correctly. It also includes some basic information about important Apache files and directory locations.

Step 4 — Managing the Apache Process

Now that you have your web server up and running, let's go over some basic management commands.

To stop your web server, type:

```
$ sudo systemctl stop apache2
```

SCROLL TO TOP

To start the web server when it is stopped, type:

```
$ sudo systemctl start apache2
```

To stop and then start the service again, type:

```
$ sudo systemctl restart apache2
```

If you are simply making configuration changes, Apache can often reload without dropping connections. To do this, use this command:

```
$ sudo systemctl reload apache2
```

By default, Apache is configured to start automatically when the server boots. If this is not what you want, disable this behavior by typing:

```
$ sudo systemctl disable apache2
```

To re-enable the service to start up at boot, type:

```
$ sudo systemctl enable apache2
```

Apache should now start automatically when the server boots again.

Step 5 — Setting Up Virtual Hosts (Recommended)

When using the Apache web server, you can use *virtual hosts* (similar to server blocks in Nginx) to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called **example.com**, but you should **replace this with your own domain name**. To learn more about setting up a domain name with DigitalOcean, see our [Introduction to DigitalOcean DNS](#).

Apache on Debian 9 has one server block enabled by default that is configured to serve documents from the `/var/www/html` directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying `/var/www/html`, let's create a directory structure within `/var/www` for our **example.com** site, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the directory for **example.com** as follows, using the `-p` flag to create any necessary parent directories:

```
sudo mkdir -p /var/www/example.com/html
```

Next, assign ownership of the directory with the `$USER` environmental variable:

SCROLL TO TOP

```
$ sudo chown -R $USER:$USER /var/www/example.com/html
```

The permissions of your web roots should be correct if you haven't modified your `unmask` value, but you can make sure by typing:

```
$ sudo chmod -R 755 /var/www/example.com
```

Next, create a sample `index.html` page using `nano` or your favorite editor:

```
$ nano /var/www/example.com/html/index.html
```

Inside, add the following sample HTML:

```
                                /var/www/example.com/html/index.html

<html>
  <head>
    <title>Welcome to Example.com!</title>
  </head>
  <body>
    <h1>Success! The example.com server block is working!</h1>
  </body>
</html>
```

Save and close the file when you are finished.

In order for Apache to serve this content, it's necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file located at `/etc/apache2/sites-available/000-default.conf` directly, let's make a new one at `/etc/apache2/sites-available/example.com.conf`:

```
$ sudo nano /etc/apache2/sites-available/example.com.conf
```

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

```
                                /etc/apache2/sites-available/example.com.conf

<VirtualHost *:80>
  ServerAdmin admin@example.com
  ServerName example.com
  ServerAlias www.example.com
  DocumentRoot /var/www/example.com/html
  ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Notice that we've updated the `DocumentRoot` to our new directory and `ServerAdmin` to an email that the **example.com** site administrator can access. We've also added two directives: `ServerName`, which establishes the base domain that should match for this virtual host definition, and `ServerAlias`, which defines further names that should match as if they were the base name.

Save and close the file when you are finished.

Let's enable the file with the `a2ensite` tool:

```
$ sudo a2ensite example.com.conf
```

Disable the default site defined in `000-default.conf`:

```
$ sudo a2dissite 000-default.conf
```

Next, let's test for configuration errors:

```
$ sudo apache2ctl configtest
```

You should see the following output:

Output

Syntax OK

Restart Apache to implement your changes:

```
$ sudo systemctl restart apache2
```

Apache should now be serving your domain name. You can test this by navigating to `http://example.com`, where you should see something like this:

Success! The example.com virtual host is working!

Step 6 – Getting Familiar with Important Apache Files and Directories

Now that you know how to manage the Apache service itself, you should take a few minutes to familiarize yourself with a few important directories and files.

Content

- `/var/www/html` : The actual web content, which by default only consists of the default Apache page you saw earlier, is served out of the `/var/www/html` directory. This can be changed by altering Apache configuration files.

Server Configuration

- `/etc/apache2` : The Apache configuration directory. All of the Apache configuration files reside here.
- `/etc/apache2/apache2.conf` : The main Apache configuration file. This can be modified to make changes to the Apache global configuration. This file is responsible for loading many of the other files in the configuration directory.
- `/etc/apache2/ports.conf` : This file specifies the ports that Apache will listen on. By default, Apache listens on port 80 and additionally listens on port 443 when a module providing SSL capabilities is enabled.
- `/etc/apache2/sites-available/` : The directory where per-site virtual hosts can be stored. Apache will not use the configuration files found in this directory unless they are linked to the `sites-enabled` directory. Typically, all server block configuration is done in this directory, and then enabled by linking to the other directory with the `a2ensite` command.
- `/etc/apache2/sites-enabled/` : The directory where enabled per-site virtual hosts are stored. Typically, these are created by linking to configuration files found in the `sites-available` directory with the `a2ensite`. Apache reads the configuration files and links found in this directory when it starts or reloads to compile a complete configuration.
- `/etc/apache2/conf-available/` , `/etc/apache2/conf-enabled/` : These directories have the same relationship as the `sites-available` and `sites-enabled` directories, but are used to store configuration fragments that do not belong in a virtual host. Files in the `conf-available` directory can be enabled with the `a2enconf` command and disabled with the `a2disconf` command.
- `/etc/apache2/mods-available/` , `/etc/apache2/mods-enabled/` : These directories contain the available and enabled modules, respectively. Files ending in `.load` contain fragments to load specific modules, while files ending in `.conf` contain the configuration for those modules. Modules can be enabled and disabled using the `a2enmod` and `a2dismod` command.

Server Logs


- `/var/log/apache2/access.log` : By default, every request to your web server is recorded in this log file unless Apache is configured to do otherwise.
- `/var/log/apache2/error.log` : By default, all errors are recorded in this file. The `LogLevel` directive in the Apache configuration specifies how much detail the error logs will contain.

Conclusion

Now that you have your web server installed, you have many options for the type of content you can serve and the technologies you can use to create a richer experience.

If you'd like to build out a more complete application stack, you can look at this article on [how to configure a LAMP stack on Debian 9.](#)

By: Justin Ellingwood By: Kathleen Juell By: Hanif Jetha

 Upvote (0)  Subscribe  Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

How To Migrate your Apache Configuration from 2.2 to 2.4 Syntax.

How To Get Started With mod_pagespeed with Apache on a CentOS and Fedora Cloud Server

How To Use the .htaccess File

How To Set Up Mod_Rewrite (page 2)

How To Create a Custom 404 Page in Apache

0 Comments

Leave a comment...

SCROLL TO TOP

Log In to Comment



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)

SCROLL TO TOP