How To Install the Apache Web Server on Ubuntu 20.04

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, i'll explain how to install an Apache web server on your Ubuntu 20.04 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 Initial server setup guide for Ubuntu 20.04.

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

Step 1 – Installing Apache

Apache is available within Ubuntu'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 receive a list of the application profiles:

```
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

As indicated by the output, there are three profiles available for Apache:

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

- **Apache Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)

- **Apache 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 status
```

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Apache (v6)                ALLOW       Anywhere (v6)
```

As indicated by the output, the profile has been activated to allow access to the Apache web server.

Step 3 – Checking your Web Server

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

```
sudo systemctl status apache2
```

```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-04-20 16:08:19 UTC; 3 days ago
     Docs: https://httpd.apache.org/docs/2.4/
 Main PID: 29435 (apache2)
    Tasks: 55 (limit: 1137)
   Memory: 8.0M
   CGroup: /system.slice/apache2.service
           ├─29435 /usr/sbin/apache2 -k start
           └─29437 /usr/sbin/apache2 -k start
           └─29438 /usr/sbin/apache2 -k start
```

As confirmed by this output, the service has 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 determine if they work.

Another option is to use the Icanhazip tool, which should give you your public IP address as read from another location on the internet:

```
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 Ubuntu 20.04 Apache web page:

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 using **systemctl**.

To stop your web server, type:

```
sudo systemctl stop apache2
```

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 **your_domain**, but you should **replace this with your own domain name**.

Apache on Ubuntu 20.04 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 a **your_domain** 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 **your_domain** as follows:

```
sudo mkdir /var/www/your_domain
```

Next, assign ownership of the directory with the **$USER** environment variable:

```
sudo chmod -R 755 /var/www/your_domain
```

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

```
sudo nano /var/www/your_domain/index.html
```

Inside, add the following sample HTML:

```
<html>
    <head>
        <title>Welcome to Your_domain!</title>
    </head>
    <body>
        <h1>Success!  The your_domain virtual host 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/your_domain.conf**:

```
sudo nano /etc/apache2/sites-available/your_domain.conf
```

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

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName your_domain
    ServerAlias www.your_domain
    DocumentRoot /var/www/your_domain
    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 **your_domain** 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 your_domain.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 receive the following 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://your_domain**, where you should see something like this:

Success! The your_domain 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/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.

- **/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.

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.

DOWNLOAD AS PDF
