How To Install the Apache Web Server on Ubuntu 22.04

Darshita Daga

Choose a different version or distribution

Introduction

Before we begin talking about how to install Apache Web Server on Ubuntu 22.04, let's briefly understand – **What is Apache Web Server?**

Apache Web Server is a powerful and widely used open-source web server software. It enables the hosting of websites and applications on the internet. With its flexibility and stability, Apache has become a popular choice among developers and businesses. It supports multiple platforms and boasts a strong security framework, ensuring the safety of user data.

Apache Web Server offers excellent performance and scalability, making it ideal for handling high traffic and serving dynamic content. Whether you are a beginner or an experienced developer, Apache Web Server provides the tools and features necessary to deliver reliable and efficient web hosting solutions.

In this tutorial, you will learn how to install an Apache web server on your Ubuntu 22.04 server. We will also address a few FAQs on how to install Apache Web Server on Ubuntu 22.04.

Advantages of Apache Web Server

1. **Reliability:** Apache Web Server is known for its stability and robustness, ensuring uninterrupted website availability.
2. **Security:** Apache offers a strong security framework, protecting websites and applications from various threats.
3. **Flexibility:** It supports multiple platforms and can be easily customized to meet specific hosting needs.
4. **Performance:** Apache excels in delivering high performance, efficiently handling large volumes of web traffic.
5. **Scalability:** With its scalable architecture, Apache can seamlessly accommodate growing website demands and user traffic.

Prerequisites To Install the Apache Web Server on Ubuntu 22.04

You will need an Ubuntu 22.04 server set up with a non-**root** user with `sudo` privileges and a firewall installed to block unnecessary ports before starting this tutorial.

Once you've finished configuring this, log in as your non-**root** user and move on to step one.

Step 1 - Installing Apache

Due to the fact that Apache is a component of Ubuntu's default software repositories, it may be installed using standard package management tools.

Let's start by updating the local package index to match the most recent upstream changes:

```
sudo apt update
```

After that, install the `apache2` package:

```
sudo apt install apache2
```

`apt` will install Apache and all necessary dependencies once the installation has been approved.

Step 2 - Adjusting the Firewall

Before testing Apache, it's crucial to adjust the firewall settings to allow outside access to the standard web ports. Assuming you followed the necessary steps carefully, you should have a UFW firewall configured to restrict access to your server.

Apache registers with UFW after installation in order to offer a few application profiles that can be used to enable or

prevent access to Apache through the firewall.

For a list of all `ufw` application profiles, enter the following:

```
sudo ufw app list
```

A list of the application profiles will be sent to you:

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

The output reveals that Apache is compatible with three different profiles:

- **`Apache`**: With this profile, only port `80` (used for normal, unencrypted web traffic) is open.
- **`Apache Full`**: This profile opens port `443` for TLS/SSL-encrypted communication and port `80` for regular, unencrypted web traffic.
- **`Apache Secure`**: The only port that is open with this profile is port `443` (TLS/SSL encrypted traffic).

It is advised that you enable the most restrictive profile that will still accept the traffic you've set. Since SSL hasn't yet been set up on our server in this guide, all traffic on port `80` needs to be permitted:

```
sudo ufw allow 'Apache'
```

You may check the change by typing:

```
sudo ufw status
```

There will be a list of permitted HTTP traffic in the output:

```
Output
Status: active

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

The profile has been activated to permit access to the Apache web server, as shown by the output.

Step 3 - Checking your Web Server

Apache is started by Ubuntu 20.04 once the installation process is finished. The web server must be operational.

Type the following command to verify that the service is active with the `systemd` init system:

```
sudo systemctl status apache2
```

```
Output
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese>
     Active: active (running) since Tue 2022-04-26 15:33:21 UTC; 43s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 5089 (apache2)
      Tasks: 55 (limit: 1119)
     Memory: 4.8M
        CPU: 33ms
     CGroup: /system.slice/apache2.service
             ├─5089 /usr/sbin/apache2 -k start
             ├─5091 /usr/sbin/apache2 -k start
             └─5092 /usr/sbin/apache2 -k start
```

This output shows that the service launched properly. The best way to test this, though, is to request a page from Apache.

You can check the program's functionality by visiting the default Apache landing page using your IP address. There are several ways to obtain your server's IP address from the command line if you are unaware of it.

On your server, type the following into the command prompt:

```
hostname -I
```

Multiple addresses will be returned, each one separated by a space. You can check to see if each one works in your web browser.

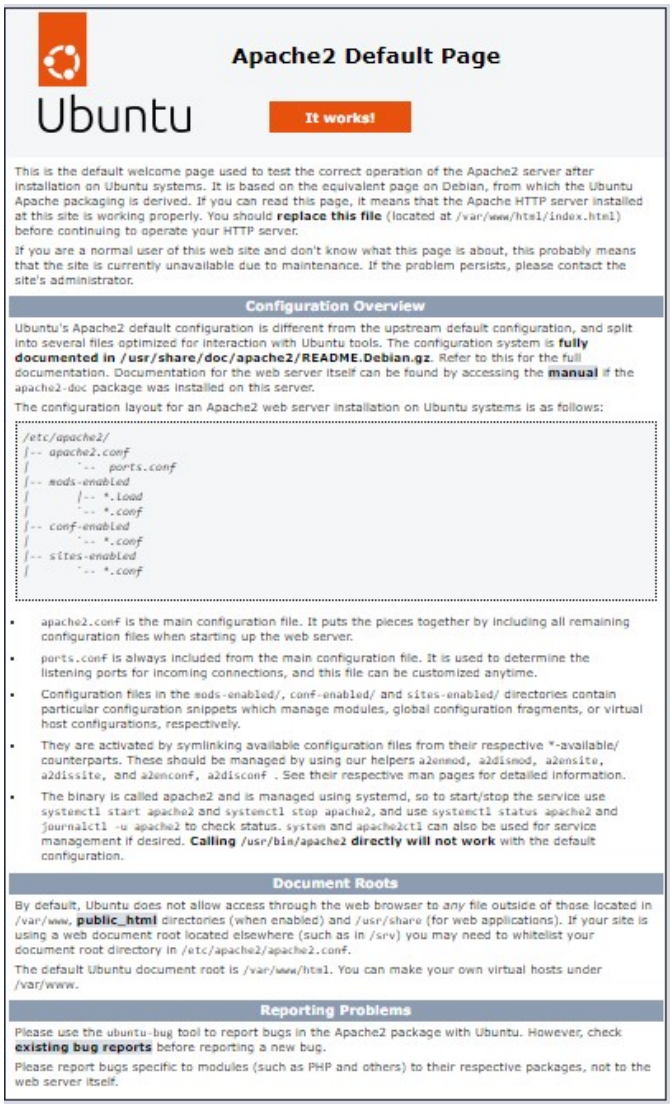
You could also use the free `icanhazip.com` utility. When you visit this website, your computer's public IP address as retrieved from another location on the internet is returned:

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

Once you are aware of your server's IP address, enter it into your browser's address bar as shown below:

```
http://your_server_ip
```

The default Ubuntu 22.04 Apache web page ought to appear:



This page shows that Apache is operating properly. It also gives some basic details about the locations of key Apache files and directories.

Step 4 - Managing the Apache Process

Now that your web server is operational, let's review some fundamental `systemctl` management commands.

Type the following to terminate your web server:

```
sudo systemctl stop apache2
```

To restart the web server once it has been shut down, enter:

```
sudo systemctl start apache2
```

To stop and then restart the service, enter:

```
sudo systemctl restart apache2
```

If you only make configuration changes, Apache may typically reload without losing connections. Use this command to accomplish this:

```
sudo systemctl reload apache2
```

Apache is by default configured to start immediately when the server boots. Disable this behaviour if it is not what you want by typing:

```
sudo systemctl disable apache2
```

Enter the following command to make the service bootable once more:

```
sudo systemctl enable apache2
```

Apache should now start up immediately when the server restarts.

Step 5 - Configuring Virtual Hosts (Recommended)

To encapsulate configuration information and host several domains from a single server when using the Apache web server, you can use *virtual hosts* (similar to server blocks in Nginx). **your_domain** will be the domain name we create; however, you should **substitute your own domain name here.**

Info: Please refer to our Networking Documentation if you are setting up a domain name with DigitalOcean.

One server block is enabled by default in Apache on Ubuntu 22.04, and it is set up to serve files from the `/var/www/html` directory. While this is effective for a single site, hosting many sites might make it cumbersome. Instead of making changes to `/var/www/html`, let's construct a directory structure within `/var/www` for a site under **your_domain**, leaving `/var/www/html` in place to act as the default directory if a client request doesn't match any other sites.

The directory for **your_domain** should be created as follows:

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

Next, use the `$USER` environment variable to specify who owns the directory:

```
sudo chown -R $USER:$USER /var/www/your_domain
```

If you haven't changed the umask value, which determines the default file permissions, the permissions of your web roots should be accurate. You can enter the following command to verify that your permissions are valid and provide the owner read, write, and execute access to the files while only providing groups and other users read and execute access:

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

Next, using `nano` or your preferred editor, make a sample `index.html` page as follows:

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

Add the following example HTML inside:

```
<html>
    <head>
        <title>Welcome to Your_domain!</title>
    </head>
    <body>
        <h1>Success!  The your_domain virtual host is working!</h1>
    </body>
</html>
```

When you're done, save and close the file. You can do this in nano by pressing `CTRL + X`, then `Y`, and `ENTER`.

It is important to construct a virtual host file with the appropriate directives in order for Apache to deliver this content. Let's create a new configuration file at `/etc/apache2/sites-available/your_domain.conf` rather than directly alter the default one at `/etc/apache2/sites-available/000-default.conf`:

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

The configuration block below should be pasted in; it is similar to the default but adjusted 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>
```

You'll see that we've changed `ServerAdmin` to an email that **your_domain** site administrator can access and `DocumentRoot` to point to our new directory. Additionally, we've introduced two directives: `ServerName`, which provides the base domain that must match for the definition of this virtual host, and `ServerAlias`, which specifies additional names that must match as if they were the base name.

Once you're done, save and close the document.

Let's use the `a2ensite` tool to enable the file:

```
sudo a2ensite your_domain.conf
```

The default site specified in `000-default.conf` should be disabled:

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

Let's now check for configuration errors:

```
sudo apache2ctl configtest
```

You should get the following output:

```
Output
. . .
Syntax OK
```

To put your modifications into effect, restart Apache:

```
sudo systemctl restart apache2
```

Your domain name ought should now be served by Apache. You may verify this by going to `http://your_domain`, where you ought to see the following:

Step 6 – Getting Familiar with Important Apache Files and Directories

You should spend some time getting acquainted with a few significant directories and files now that you are aware of how to control the Apache service itself.

Content

- `/var/www/html`: The `/var/www/html` directory is used to serve actual web content, which by default just consists of the Apache default page you previously saw. By making changes to the Apache configuration files, this can be adjusted.

Server Configuration

- `/etc/apache2`: The Apache configuration directory. Here are all the configuration files for Apache.
- `/etc/apache2/apache2.conf`: The main configuration file for Apache. This can be changed to make changes to the Apache global configuration. This file is in charge of loading several of the other files in the configuration directory.
- `/etc/apache2/ports.conf`: The ports that Apache will listen on are listed in this file. When a module with SSL capabilities is activated, Apache also listens on port `443` in addition to port `80` by default.
- `/etc/apache2/sites-available/`: The directory in which per-site virtual hosts can be placed. The configuration files in this directory must be linked to the `sites-enabled` directory in order for Apache to use them. Typically, all server block configuration is completed in this directory, and the other directory is linked with the `a2ensite` command to enable it.
- `/etc/apache2/sites-enabled/`: The directory in which activated per-site virtual hosts are stored. These are often made by using the `a2ensite` to connect to configuration files located in the `sites-available` directory. When

Apache begins or reloads, it scans the configuration files and links contained in this directory to create a complete setup.

- `/etc/apache2/conf-available/`, `/etc/apache2/conf-enabled/`: These directories have the same relationship as the `sites-available` and `sites-enabled` directories, however they are intended to hold configuration fragments that do not belong in a virtual host. Using the `a2enconf` and `a2disconf` commands, you can enable and disable files in the `conf-available` directory.
- `/etc/apache2/mods-available/`, `/etc/apache2/mods-enabled/`: These directories include the modules that are both available and enabled. Files ending in `.load` contain fragments for loading specific modules, whereas files ending in `.conf` include configuration for those modules. Using the `a2enmod` and `a2dismod` commands, modules can be enabled and disabled.

Server Logs

- `/var/log/apache2/access.log`: Unless Apache is set to do otherwise, every request to your web server is by default logged in this log file.
- `/var/log/apache2/error.log`: By default, this file stores all errors. In the Apache configuration, the `LogLevel` directive defines how much detail the error logs will contain.

FAQs to Install the Apache Web Server on Ubuntu 22.04

Where is the configuration file for Apache located in Ubuntu 22.04?

The main configuration file for Apache on Ubuntu 22.04 is located at `/etc/apache2/apache2.conf`.

How do I start and stop the Apache Web Server on Ubuntu 22.04?

To start Apache, use the command `sudo systemctl start apache2`. To stop it, run `sudo systemctl stop apache2`.

How can I enable or disable the Apache service on system boot in Ubuntu 22.04?

To enable Apache on system boot, execute `sudo systemctl enable apache2`. To disable it, run `sudo systemctl disable apache2`.

Where are the website files stored in Apache on Ubuntu 22.04?

By default, the website files for Apache on Ubuntu 22.04 are stored in the directory `/var/www/html/`.

How do I test if Apache is running correctly on Ubuntu 22.04?

Open a web browser and enter `localhost` or the IP address of your server. If you see the Apache default page, it is running correctly.

How do I enable SSL/TLS encryption for Apache on Ubuntu 22.04?

Install the SSL module with `sudo apt-get install openssl` and then generate a certificate and configure Apache to use it.

Conclusion

We hope this detailed tutorial helped you understand how to install Apache Web Server on Ubuntu 22.04.

If you have any suggestions or queries, kindly leave them in the comments section.