How to setup an Apache Web Server on Ubuntu Linux

Level of Difficulty: Intermediate

Jason Zheng

March 2, 2023

Contents

1 Introduction

1.1 What is a web server?

A web server is a computer that stores web server software and a website's component files (for example, HTML documents, images, and JavaScript files). The web server connects to the Internet and supports physical data interchange with other devices connected to the web.

On the software side, there are applications such as Apache that understands HTTP(the protocol used by browsers to render a webpage) that can store your website files and send this back to your users/clients.

For example, an user is trying to access Stony Brook University's website. The user will go to stonybrook.edu into their favorite browser, Chrome. Chrome will then find the web server that is hosting Stony Brook University's site and request (HTTP Request) the website files. Stony Brook's web server would then send back the site's data (HTTP Response), which is then read by the user's browser and displayed as a web page.

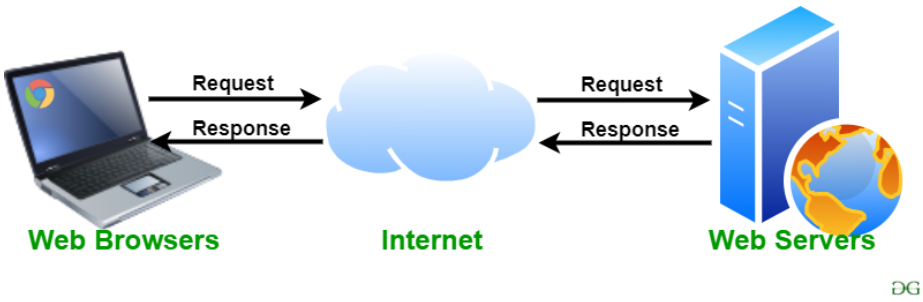

Figure 1: How a web server works

1.2 Prerequisites

An Ubuntu operating system is required. If you do not have one, you can rent a cloud server. At this point, you should already have an existing Ubuntu server and a non root user with sudo privileges. If this has not been achieved yet, use this tutorial by Digital Ocean*. For this tutorial, we'll be using Ubuntu Linux 20.04 hosted on Digital Ocean.

*https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04

2 Tutorial

2.1 Installing Apache

Since Apache is available within Ubuntu's default software repository, we begin by updating the local package index to reflect the latest changes:

```
$ sudo apt update
```

Then install the "apache2" package:

```
$ sudo apt install apache2
```

2.2 Adjusting the Firewall:

It's necessary to modify the firewall to allow outside access to the web server. Firewall can be compared as a security guard or a bouncer. At a bar, a bouncer may be told to restrict customers who are under the age of 18 at all times and under the age of 21 from 10PM to closing. Firewalls perform the same duty, restrict access to the system based on rule(s) defined by the system administrator.

In this step we will allow access to the ports used by Apache web server:

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

Then you can verify the changes by typing:

```
$ sudo ufw status
```

This will output the current firewall profiles active on the system:

```
Output
Status: active

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

If Apache shows up in the output, you have successfully opened your web server to the public and now ready to send files to your client.

2.3 Verifying your web server is running

At this point, your web server should be up and running. We can check by running this command:

```
$ sudo systemctl status apache2
```

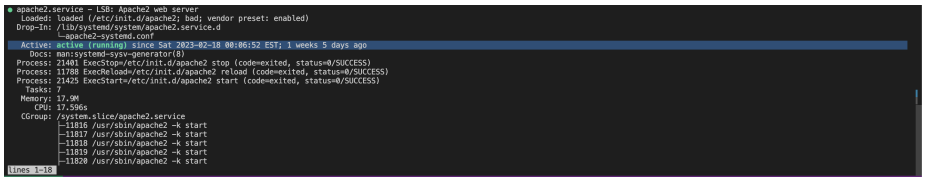

Figure 2: Apache is active and running

This confirms our server is running, however we should still verify the server responds to clients and renders the webpage correctly. To do this we'll need to find the server's public IP address:

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

Sample resulting output:

```
130.245.30.73
```

When you have the IP address, enter it into a browser like:

```
http://publicIP
```

where publicIP is server's public IP address. For example:

```
http://130.245.30.73
```

You should see the default Ubuntu Apache webpage:

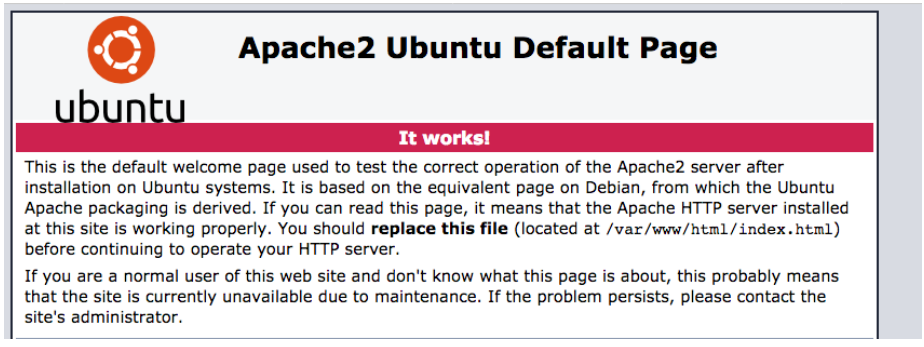

Figure 3: Default webpage

2.4 Editing your web page

All of your website data is stored in the /var/www/html directory. There is already an existing index.html file which contains the html for the default Ubuntu Apache web page. Any files within /var/www/html can now be served to your clients who tries to access your web server at your public facing IP address.

Let's try to create a sample HTML page using vim or any other editor of your choice:

```
$ sudo vim /var/www/html/index.html
```

Press the "i" key to enter "Insert" mode:

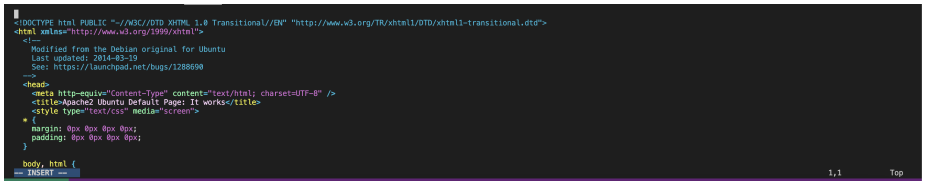

Figure 4: VIM editor

Notice once you're in insert mode, "– Insert –" shows up in the bottom left corner. Now you can edit the file and add your own HTML data:

```
<html>
    <head>
        <title>Welcome to Jason's Tutorial!</title>
    </head>
    <body>
        <h1>How to install Apache on Ubuntu</h1>
    </body>
</html>
```

To save the file, press "esc" key, and type ":wq". This will save the file and quit the editor. Now we must restart Apache to propagate the changes:

```
$ sudo systemctl restart apache2
```

Congratulation you have successfully configured a web server. To create an application with more functionality and richer experience, consider exploring the LAMP stack.

3 Additional Resources

3.1 Cloud Providers

Amazon Web Services https://aws.amazon.com/
Digital Ocean https://www.digitalocean.com/
Linode https://www.linode.com/

3.2 Web Documentation:

Mozilla MDN https://developer.mozilla.org/en-US/