Week 9 – Apache Web Server

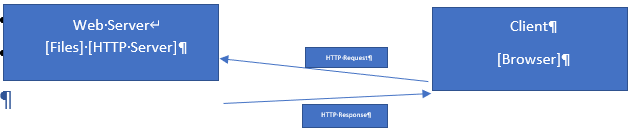
# Part 1 – Basics

Two parts of running a website:

1. Server administration
2. Content creation

We’ll be looking at the sever admin side of things – you’ve already done some content creation. A web server does just that, serves web page responses to requests (client). On the hardware side, any computer can typically be configured as a web server. Normally, organizations will host their web sites, web applications, and other online content on dedicated web server (potentially virtualized).

A “web server” is just software installed that allows a computer to accept HTTP requests and send responses. These responses will be html files, media, JavaScript, and potentially file streams. The client’s browser receives this response and renders the page accordingly.



There’s a tonne of configuration that you can do on the web server end of things. You can open different ports, use encryption, and host multiple sites from a single server. We’ll talk about this in this learning outcome.

Depending on who you trust for your information, between 65% and 80% of all webservers run some UNIX or Linux variant. (<https://w3techs.com/technologies/details/os-linux>)

Apache servers account (serve) anywhere around 30%-40% of those.

* Development work began in 1995 , and in 1999 the Apache Software Foundation (ASF) was formed.
* Under the Apache License, compatible with GPL v3.0

“Why was the name 'Apache' chosen?

The name 'Apache' was chosen from respect for the various Native American nations collectively referred to as [Apache](http://en.wikipedia.org/wiki/Apache), well-known for their superior skills in warfare strategy and their inexhaustible endurance. It also makes a cute pun on "a patchy web server" -- a server made from a series of patches -- but this was not its origin. The group of developers who released this new software soon started to call themselves the "Apache Group".”

Reference: <https://www.apache.org/foundation/faq.html#name>

Apache uses modules for flexibility – Apache is skeletal all by itself, but it uses DSOs (dynamic share objects – sort of like plug-ins) to provide more functionality. There are a lot of DSO’s, and a lot of them are installed by default by different distributions. For example Fedora will have different DSOs than Ubuntu.

**Apache** is the name of the server, sometime it is called **httpd**, this is the server daemon, it is called “apache2” on many distros, and people will of use “apache2” and “httpd” interchangeably.

* Ubuntu – Apache2
* CentOS/RedHat – httpd

# Part 2 – Installing Apache

The first thing we want to do is update the local package index to ensure we get the latest version of Apache and any dependencies. This is done via the command: **sudo apt update.**

Graphical user interface, text

Description automatically generated

Next let’s install the apache2 package (again this will potentially be call httpd in other distros).

Run the following command: **sudo apt install apache2**

Text

Description automatically generated

Once it’s installed, Ubuntu will start Apache. The web server should already be up and running.

We can check with the init system to make sure the service is up and running by typing the following:

**sudo systemctl status apache2**

We should see a screen that looks like this: (press “q” to quit)

Text

Description automatically generated with low confidence

The service has started successfully! However the best way to test this is to request a webpage.

We can access the landing page for the Apache (default) by using our IP address. If we don’t know our servers IP address we can get this via several ways (ifconfig, hostname, ip addr)

Type in the following: **hostname -I** (capital i)

A picture containing text, bottle, sign

Description automatically generated

And if we open a browser in Linux and point to one of your IP addresses, we’ll get the default Apache2 website that was created when we installed Apache2.

Graphical user interface, application

Description automatically generated

When we first install Apache, our default home directory for our web server is **/var/www/**. There is an HTML directory there that contains a single **index.html** file that is server to any visitors.

Text

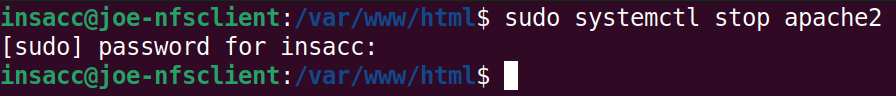
Description automatically generated

Go ahead **less index.html** – what do you see? Remember that content creation is a whole other can of worms. This should look familiar and readable though.

# Part 3 – Managing the Apache Process

Now that we have our web server up and running, we can take a look at some commands to manage the Apache web server.

To stop the webserver, we can type: **sudo systemctl stop apache2** (**man systemctl** to learn more about this command)



We can verify the service has been stopped with the **sudo systemctl status apache2** command:

Graphical user interface, text

Description automatically generated

Since our webserver is stopped, it won’t server any content for any HTTP requests.

If we try to connect to our web server via a browser, we won’t be able to:

Graphical user interface, text, application, email

Description automatically generated

To fix this, we will start our web server, let’s type the command: **sudo systemctl start apache2**. This will start the apache2 service, verify, again:

Graphical user interface

Description automatically generated

If you make configuration changes, or perhaps want to invalidate client caches depending on the programming language your are using to store thing in a cache, a typical task will be to restart Apache.

To do that, execute the following command: **sudo systemctl restart apache2**

We would do this so that our web server can reflect any configuration or module changes.



By default, Apache is configured to start automatically when the server boots. To disable automatic start, we can type in **sudo systemctl disable apache2** and then to auto start on boot we can use the command **sudo systemctl enable apache2**.

The **apachectl** command is used to control the httpd directly. (Look at **man apachectl** note the various options, ex: start, stop, restart, status, and graceful.)

We can use the command **sudo apachectl status** to view more relevant statuses for our web server.

Text

Description automatically generated

# Part 4 – Configuring Apache

When we run **systemctl status apache2** we’ll notice we have an error/warning regarding ServerName directive in the configuration.

A picture containing text

Description automatically generated

Here Apache has noticed that we don’t have a certain directive in it’s main config file. It’s config is located in **/etc/apache2**:

Text

Description automatically generated with medium confidence

We can edit **apache2.conf** to add additional options/directives/directories to our web server.

For now, we should be creating a backup of the this file before we edit it. Let’s just copy apache2.conf to apache2.conf.back using **sudo cp apache2.conf{,.bak}**

Text

Description automatically generated with medium confidence

Let’s take a look inside **apache2.conf** using VIM as our editor, as VI as it’s derivatives offer special syntax highlighting for directives, which makes them easier to spot.

Use the following command: **sudo vim /etc/apache2/apache2.conf**

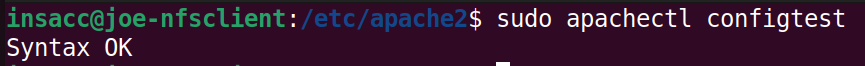
Let’s add the **ServerName** directive to out apache2.conf.

Text

Description automatically generated

To test our configuration changes to make sure our syntax is correct, we use the following command:

**sudo apachectl configtest**



Let’s purposely make some incorrect directive calls in our config file so that we can see how Apache behaves:

Text

Description automatically generated

Text

Description automatically generated

Remember to always keep backups of your config files! THIS JUST ISN’T FOR APACHE IT’S FOR EVERYTHING. Potentially , we may have multiple backups of our conf file that have been created over the years.

Let’s see what apache2 tells us when we start the server with this error, and then also check the status:

A picture containing graphical user interface

Description automatically generated

Let’s fix our blunder, and revert the changes in the apache2.conf file.

A picture containing graphical user interface

Description automatically generated

Note that the FQDN is no longer present. Yay!

# Apache Logging

By default, Apache logs all errors (warn or above) as well as any access to the web server via HTTP.

These log files are contained in the directory **/var/log/apache2**. Note that by default, we are able to change the location of the log files on a PER SITE basis.

Text

Description automatically generated

We’ll see 3 log files her by default, we’ll concentrate on **access.log** and **error.log**.

Let’s view our error log via **cat error.log**

A picture containing text

Description automatically generated

This dumps all the errors that have been logged by Apache2. We can view this file it we have trouble with a module or some other configuration. Here we see all the times we stopped and started the Apache service.

Let’s **cat access.log**

A screen shot of a computer

Description automatically generated with low confidence

The access.log will log all visitors that access pages on our web server. Each entry contains IP address, timestamp, request made, HTTP status code, number of bytes sent, referer, and browser statistics of the visitor (Google “user agent string” if you want to learn more).



This entry tells me that on March 8th, 2021 at 4:10PM (server time) the web server received a GET request to the root “/” via a Mozilla-based browser running on Ubuntu.

Let’s hit a 404 on our web server and see how that’s logged.

Graphical user interface, text, application, email

Description automatically generated

A picture containing timeline

Description automatically generated

We can see the 404 access error in the access.log.