Introduction to Apache. 5.0

The **Apache web server**, one of the most popular applications in the world. Especially important to this competition. In this chapter I will be simultaneously installing, configuring, and hardening this service with **Debian and CentOS**. So what is the Apache web server? Well, it's an application that is used to host websites and development servers. Apache is everywhere and can be implemented beyond just a web server. I have seen apache in embedded garage door systems and even Wifi extender hardware. Although i won't be getting into the vast uses for Apache. Ill let you google that and watch you learn the various implementations of Apache. For this competition and this competition only, Apache is used as web server hosting a **ecommerce** site. My overall goal for this chapter is to show you guys how to find the certain web applications through the configuration files, hardening the server though mod_security, running multiple instances of apache through different ports via **VirtualHosts** and the overall general information you need to know about the apache web server.   Apache is a huge topic, and i will not be able to cover all of the information. So in a sense I expect you to do some research on our own.

Always remember the differences between Debian's apache directory & file structure VS Redhat's. Apache is implemented differently and have different config files and subdirectories. But essentially these configuration files are the same once it comes down to editing the variables inside the configuration files. Meaning the main two config files **apache2.conf(Debian)** and **httpd.conf(Redhat)** have the same syntax and variables. They won't be two completely different files. This is also true for other distros such as **Arch and Gentoo** and I will later cover them. From now on
Debian = Red
CentOS=Blue .
Universal commands that can be done on both systems = Orange.

First lets install apache really quickly with our package managers

```
[lu@awesome]$ sudo apt-get install apache2
[lu@acecent var]$ sudo yum install httpd
```

Now lets start the server!

```
sudo apt-get update && apt-get install libapache2-mod-security2
[lu@acecent var]$ sudo systemctl start httpd.service
```

Pro tip: We can start both service with apaches universal command. If there are any errors apache wont start nor will it give us any errors with this command.
That's when we look in our /var/log/httpd and /var/log/apache2

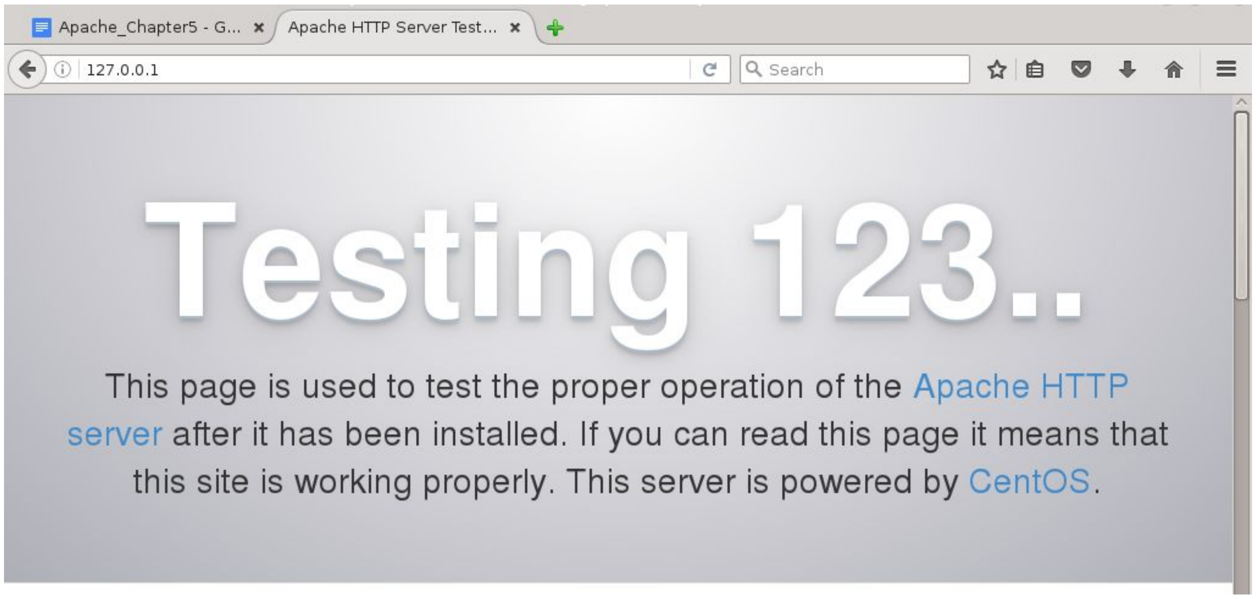
```
[lu@awesome]$    sudo apachectl start(stop|restart)
[lu@acecent var]$ sudo apachectl start(stop|restart)
```

Here are some more useful arguments for apachectl, just make you you add them to the arguments. Like so

```
[lu@acecent var]$ sudo apachectl -t
```

status - shows what Apache is doing; requires mod_status
start - starts parent web server
stop - children shut down then parent exits; terminates in-progress connections
graceful-stop - in-progress requests completed, then exit; parent shuts down
restart - parent doesn't exist, children stop, config re-read, children respawn
graceful - children finish what they're doing then shutdown, config re-read, children respawn
-V - show configuration settings used when httpd was compiled
-t - test configuration
-t -D DUMP_MODULES - list all enabled modules

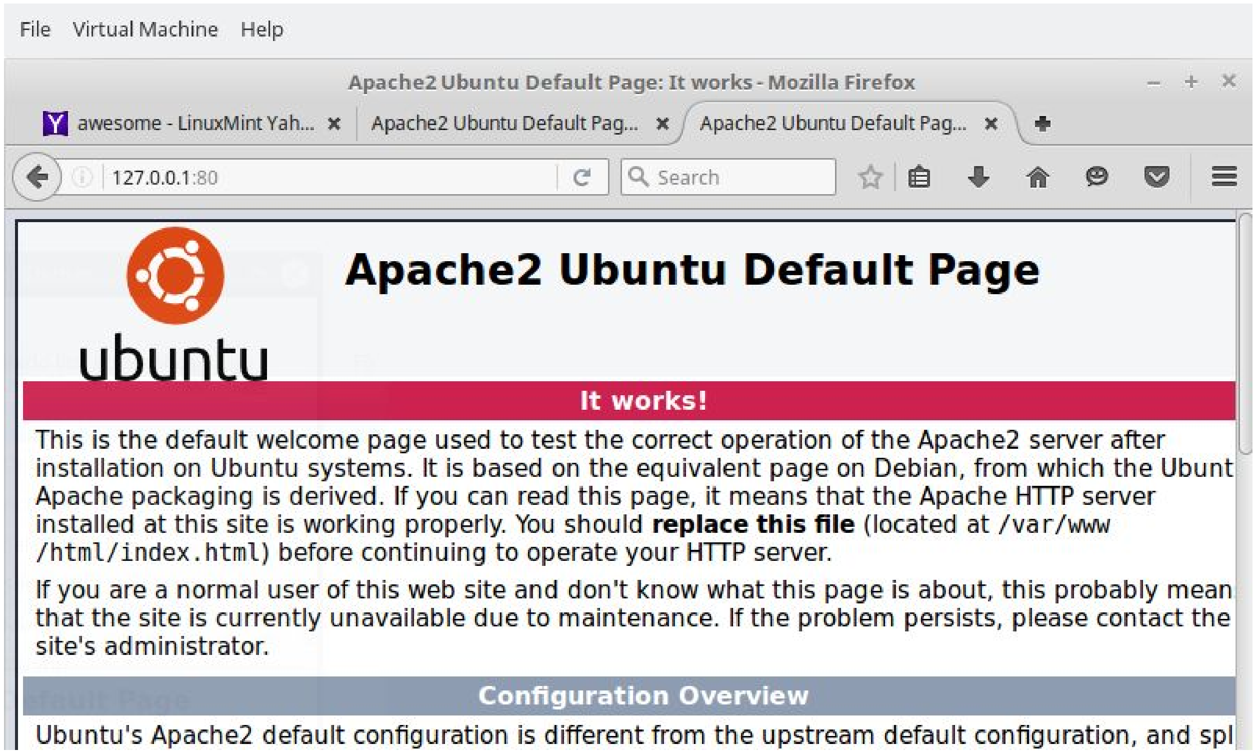
Now open a web browser and navigate to these two different but same destination URL. you should see the following pages
http://127.0.0.1/ or http://localhost/



Testing 123..

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

OR on ubuntu/ Mint. Note Apache listens on port 80 by default. So typying in 127.0.0.1:80 or localhost:80 will result in the same page. Little later on in this chapter I will show you how to change this port.


File Virtual Machine Help
Apache2 Ubuntu Default Page: It works - Mozilla Firefox
awesome - LinuxMint Yah...
Apache2 Ubuntu Default Pag...
Apache2 Ubuntu Default Pag...
127.0.0.1:80
Search
ubuntu

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubunt Apache packaging is derived. 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 mean that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

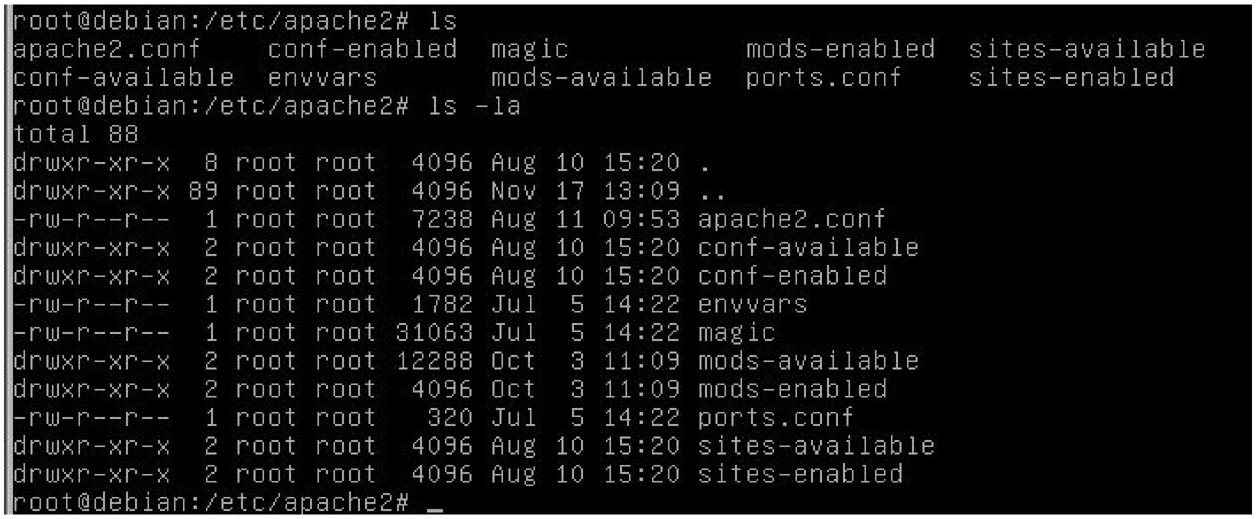
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and spl

Let’s Look at these two apache structures, remember all config files are inside **/etc** *(reference chapter 2)*

Debian :

/etc/apache
- apache2.conf
- magic
- envvars
- ports.conf
- conf-enabled
  - Apache2-doc.conf
  - Charset.conf
  - Localized-error-pages.conf
  - Other-vhosts-access-log.conf
  - Security.conf
  - serve-cgi-bin.conf
- Conf-available
  - SAME AS CONF-ENABLED
- Mods-enabled
  - A whole bunch of mods

mods-available
    Only activated modules are listed.
Sites-available
    000-default.conf
    default-ssl.conf
Sites-enabled
    Same as sites-available



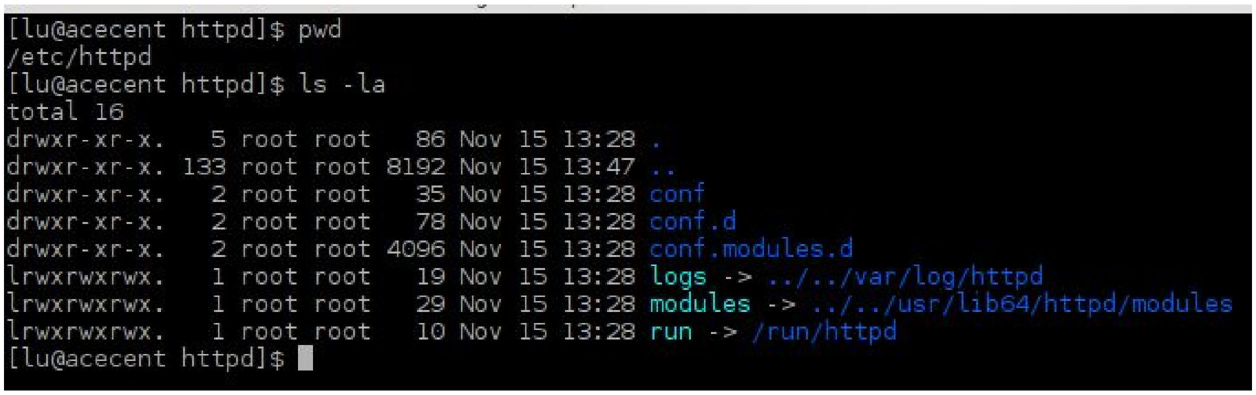
```
root@debian:/etc/apache2# ls
apache2.conf    conf-enabled  magic           mods-enabled  sites-available
conf-available  envvars       mods-available  ports.conf    sites-enabled
root@debian:/etc/apache2# ls -la
total 88
drwxr-xr-x  8 root root  4096 Aug 10 15:20 .
drwxr-xr-x 89 root root  4096 Nov 17 13:09 ..
-rw-r--r--  1 root root  7238 Aug 11 09:53 apache2.conf
drwxr-xr-x  2 root root  4096 Aug 10 15:20 conf-available
drwxr-xr-x  2 root root  4096 Aug 10 15:20 conf-enabled
-rw-r--r--  1 root root  1782 Jul  5 14:22 envvars
-rw-r--r--  1 root root 31063 Jul  5 14:22 magic
drwxr-xr-x  2 root root 12288 Oct  3 11:09 mods-available
drwxr-xr-x  2 root root  4096 Oct  3 11:09 mods-enabled
-rw-r--r--  1 root root   320 Jul  5 14:22 ports.conf
drwxr-xr-x  2 root root  4096 Aug 10 15:20 sites-available
drwxr-xr-x  2 root root  4096 Aug 10 15:20 sites-enabled
root@debian:/etc/apache2# _
```

CentOS

/etc/httpd
    conf/
        httpd.conf
        magic
    conf.d/
        autoindex.conf
        README
        userdir.conf
        Welcome.conf
    conf.modules.d/
        00-base.conf
        00-dav.conf
        00-lua.conf
        00-mpm.conf
        00-proxy.conf
        00-systemd.conf
        01-cgi.conf

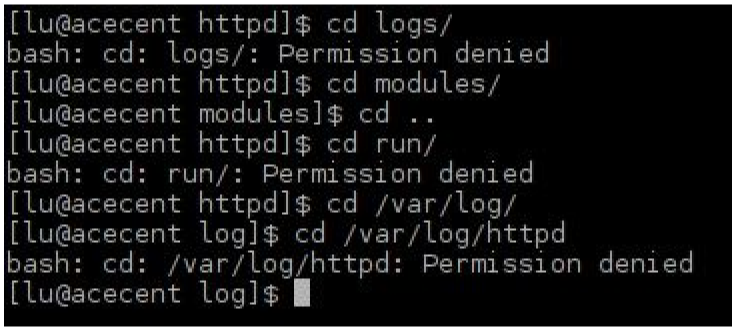
logs/
modules/
run/

As you can see the last three folders are symbolic links. These are links that are pointing to other directories in the file system



```
[lu@acecent httpd]$ pwd
/etc/httpd
[lu@acecent httpd]$ ls -la
total 16
drwxr-xr-x.   5 root root   86 Nov 15 13:28 .
drwxr-xr-x. 133 root root 8192 Nov 15 13:47 ..
drwxr-xr-x.   2 root root   35 Nov 15 13:28 conf
drwxr-xr-x.   2 root root   78 Nov 15 13:28 conf.d
drwxr-xr-x.   2 root root 4096 Nov 15 13:28 conf.modules.d
lrwxrwxrwx.   1 root root   19 Nov 15 13:28 logs -> ../../var/log/httpd
lrwxrwxrwx.   1 root root   29 Nov 15 13:28 modules -> ../../usr/lib64/httpd/modules
lrwxrwxrwx.   1 root root   10 Nov 15 13:28 run -> /run/httpd
[lu@acecent httpd]$
```

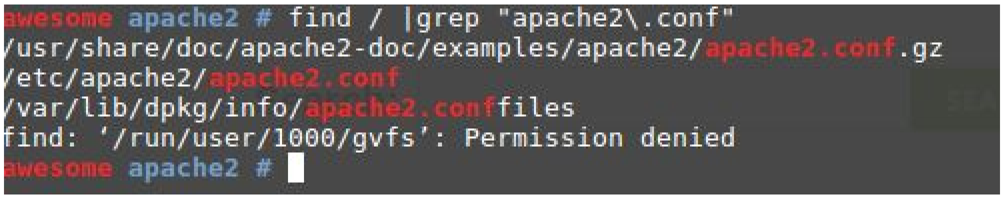
If you try and cd into these without root privileges you will get an access denied. Except for modules because it's inside the /usr folder.hs is where file permissions come into play. Notice in debian a user is allowed to **cd /var/log/apache2** . In centOS it is different. CentOS permissions are a bit more strict. Notice this moving forward.



```
[lu@acecent httpd]$ cd logs/
bash: cd: logs/: Permission denied
[lu@acecent httpd]$ cd modules/
[lu@acecent modules]$ cd ..
[lu@acecent httpd]$ cd run/
bash: cd: run/: Permission denied
[lu@acecent httpd]$ cd /var/log/
[lu@acecent log]$ cd /var/log/httpd
bash: cd: /var/log/httpd: Permission denied
[lu@acecent log]$
```

If for some reason these files have been moved, and the default installation is not where it's supposed to be. We can search for the files and find our way to these commands. Note this works for any file you are trying to find. Just make sure escape the . with the \

[lu@awesome]$ sudo find / |grep "apache2\.conf"
[lu@acecent var]$ sudo find / |grep "httpd\.conf"



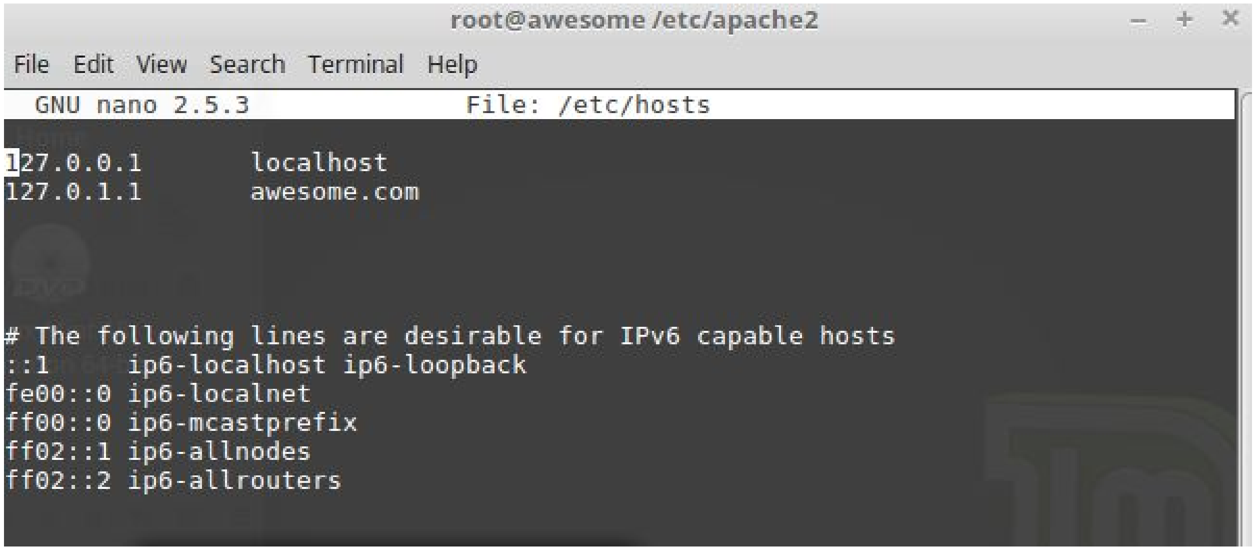
```
awesome apache2 # find / |grep "apache2\.conf"
/usr/share/doc/apache2-doc/examples/apache2/apache2.conf.gz
/etc/apache2/apache2.conf
/var/lib/dpkg/info/apache2.conffiles
find: '/run/user/1000/gvfs': Permission denied
awesome apache2 #
```

5.1 Host File

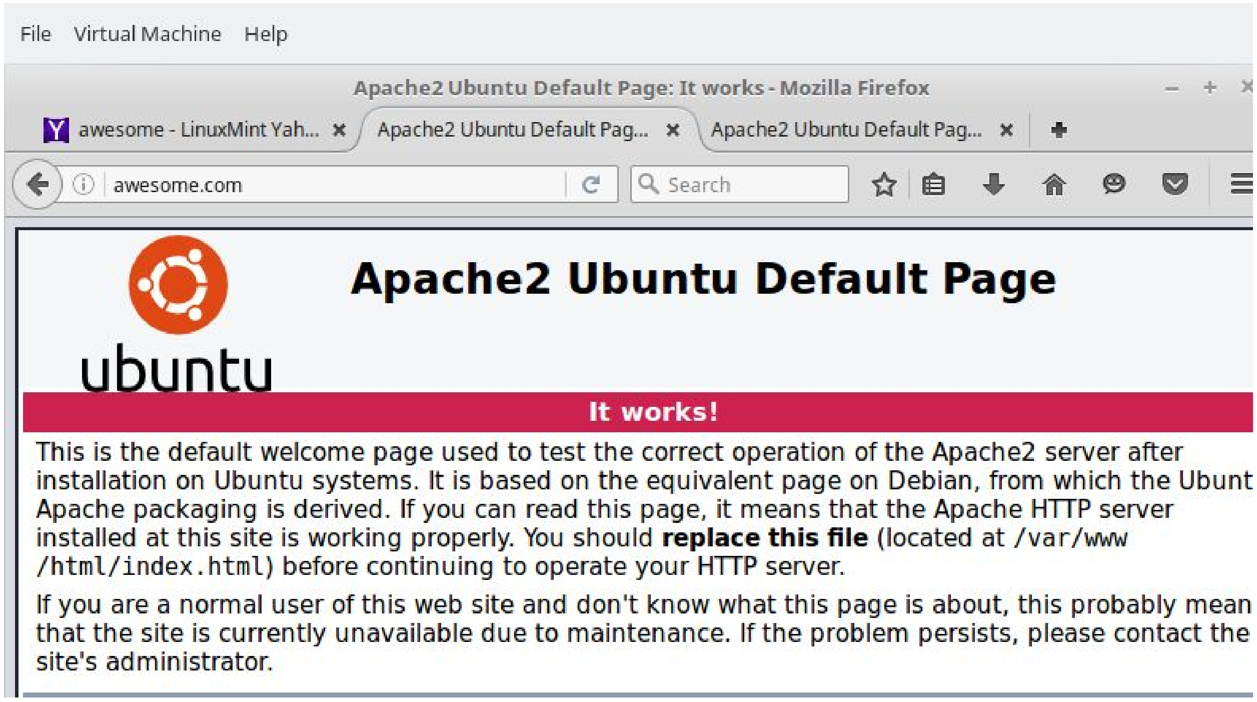
Now that we have the directory structures out of the way, how do we know to type in localhost or 127.0.0.1 where does this come from? Lets check the host file. We will need this when we create a virtual host. A host file list the IP address of an apache virtual host and its domain name.

```
[lu@awesome]$ sudo /etc/hosts
```

If I edit this file and add the IP address 127.0.1.1 and set the domain name to awesome.com .


root@awesome /etc/apache2
File Edit View Search Terminal Help

```
  GNU nano 2.5.3                  File: /etc/hosts

127.0.0.1       localhost
127.0.1.1       awesome.com



# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```


File Virtual Machine Help
Apache2 Ubuntu Default Page: It works - Mozilla Firefox
awesome - LinuxMint Yah...
Apache2 Ubuntu Default Pag...
Apache2 Ubuntu Default Pag...
awesome.com
Search
ubuntu

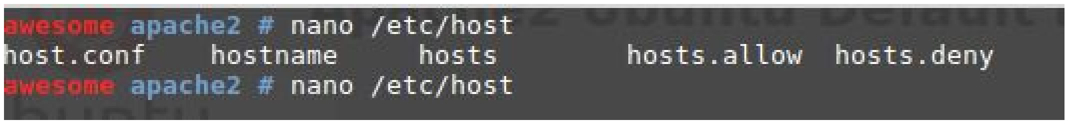
Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubunt Apache packaging is derived. 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 mean that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

This will be the result. Awesome.com will direct me to my default apache page. Pretty cool right? Well locally this is good to go. But if you really want to host this awesome.com , you would need to a lot more work than just your host file.
Dont confuse **/etc/hosts** with any of the files listed here, they have different functions



```
awesome apache2 # nano /etc/host
host.conf     hostname      hosts         hosts.allow   hosts.deny
awesome apache2 # nano /etc/host
```

hostname

- is the file that list the hostname, in this case its in red “awesome”

host.allow

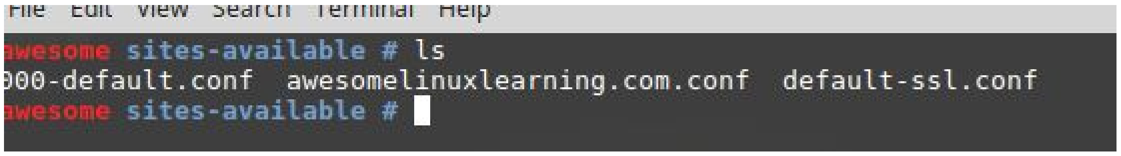
- a whitelist file allowing other machines be able to have access to this machine. (SSH, HTTP, FTP, PING) ETC

host.deny

- a black list that does the same as host.allow

5.3 VirtualHost

Now lets create a new virtual host. So what exactly is a virtual host? Well a virtual host is just an instance of the apache web server, meaning it's a copy of the original server but virtualized with its own settings. Let’s navigate to our sites-available folder and copy the **000-default.conf** to **awesomelinuxlearning.com.conf**



File Edit View Search Terminal Help

```
awesome sites-available # ls
000-default.conf  awesomelinuxlearning.com.conf  default-ssl.conf
awesome sites-available # 
```

Now there are a few things we want to change in the new virtual host configuration file.
[awesome@ sites-available]$ nano awesomelinuxlearning.com.conf
We need to edit the very first line

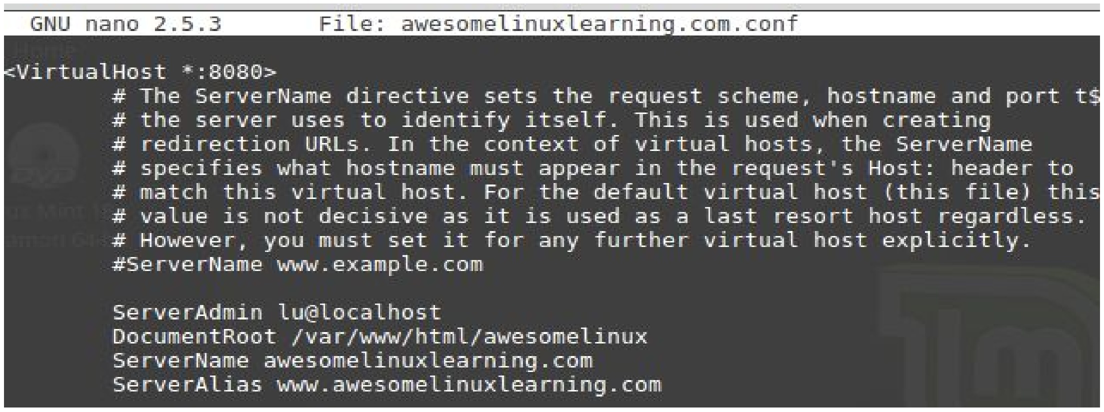
- **<VirtualHost *:80> to <VirtualHost *:8080>**

And edit the

- **DocumentRoot to /var/www/html to /var/www/html/awesomelinux**

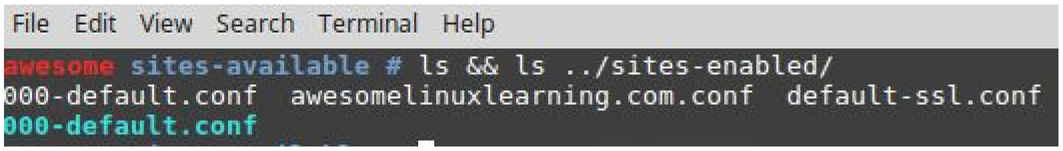
Because we already have the default host listening on port 80 and serving its files in /var/www/html . **Also add the following lines**

- **ServerName awesomelinuxlearning.com**
- **ServerAlias www.awesomelinuxlearning.com**



```
 GNU nano 2.5.3        File: awesomelinuxlearning.com.conf

<VirtualHost *:8080>
        # The ServerName directive sets the request scheme, hostname and port t$
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin lu@localhost
        DocumentRoot /var/www/html/awesomelinux
        ServerName awesomelinuxlearning.com
        ServerAlias www.awesomelinuxlearning.com
```

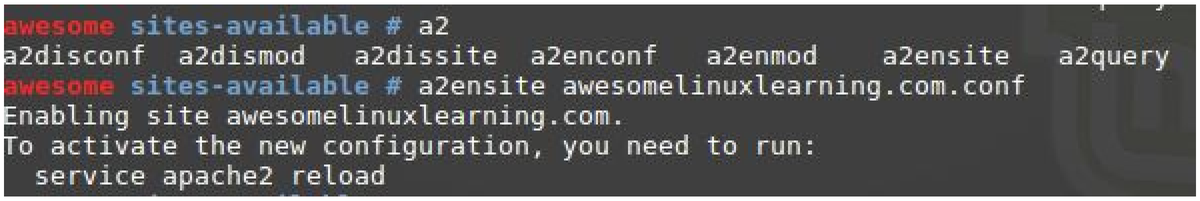
Now that we have our virtualhost set up we need to enable the site. Note, look at sites-available and sites-enabled



```
awesome sites-available # ls && ls ../sites-enabled/
000-default.conf  awesomelinuxlearning.com.conf  default-ssl.conf
000-default.conf
```

Our new **awesomelinuxlearning.com.conf** is not listed in our **sites-enabled**. Lets enable it by executing

[awesome@ sites-available]$ sudo a2ensite awesomelinuxlearning.com.conf



```
awesome sites-available # a2
a2disconf  a2dismod   a2dissite  a2enconf   a2enmod    a2ensite   a2query
awesome sites-available # a2ensite awesomelinuxlearning.com.conf
Enabling site awesomelinuxlearning.com.
To activate the new configuration, you need to run:
  service apache2 reload
```

Next once that is all well and done, we need to create the Document root. In our **awesomelinuxlearning.com.conf** file we listed it as **/var/www/html/awesomelinux**

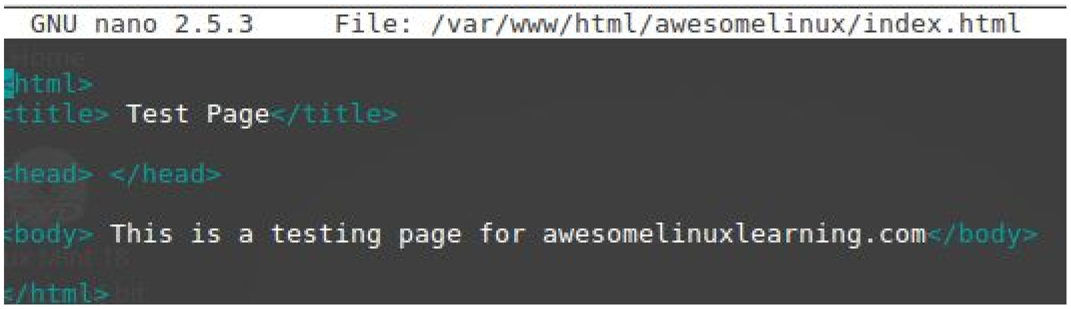
[awesome@ sites-available]$ mkdir /var/www/html/awesomelinux

Also create a basic html file, so we know what we have the correct webpage being served

[awesome@ sites-available]$ touch /var/www/html/awesomelinux/index.html

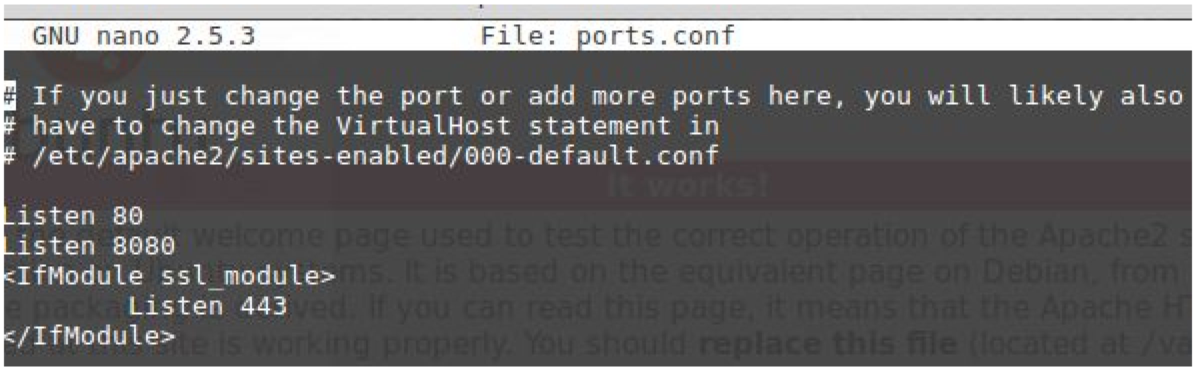
[awesome@ sites-available]$ nano /var/www/html/awesomelinux/index.html

Edit the file like so.



```
 GNU nano 2.5.3      File: /var/www/html/awesomelinux/index.html

<html>
<title> Test Page</title>

<head> </head>

<body> This is a testing page for awesomelinuxlearning.com</body>

</html>
```

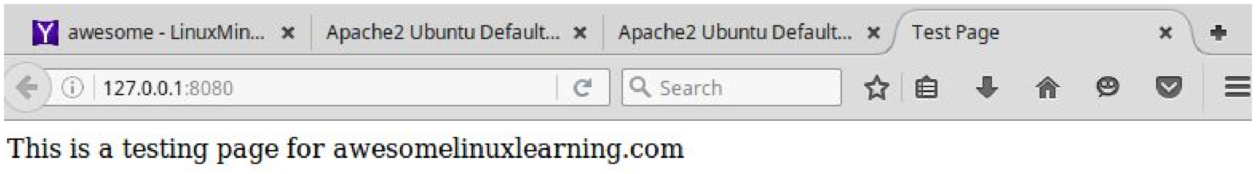
Last thing we need to make sure our port is listening, in our case port 8080 needs to be added in */etc/apache2/ports.conf*


```
  GNU nano 2.5.3              File: ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 8080
<IfModule ssl_module>
        Listen 443
</IfModule>
```

Finally restart the server

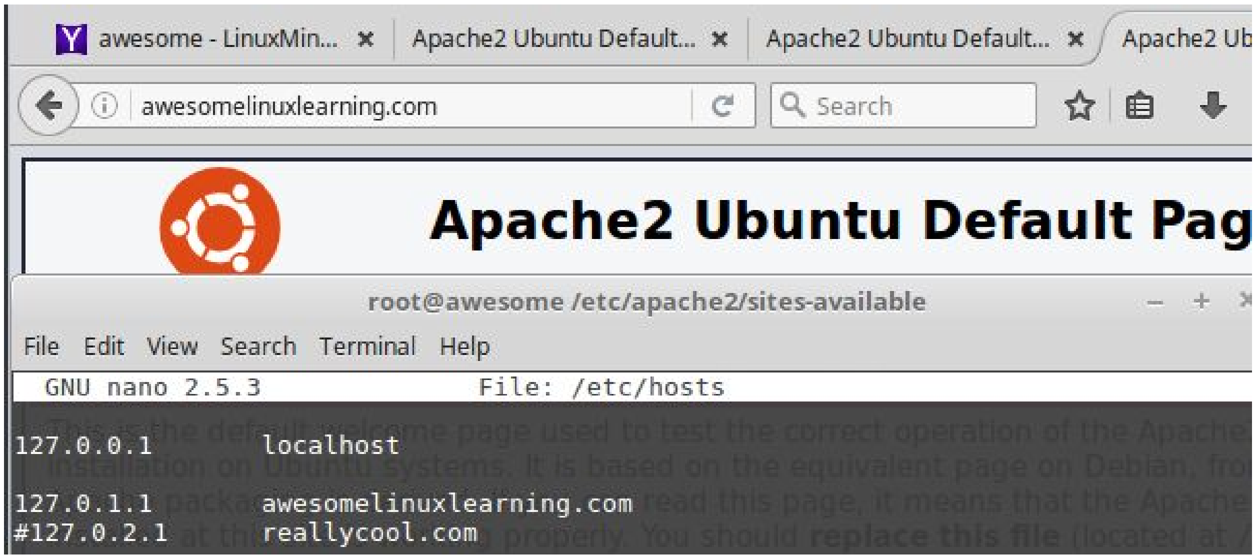
[lu@awesome]$ sudo apachectl restart

Now navigate to 127.0.0.1:8080 or localhost:8080 and you shall see the html file you created!!
Congrats! You created your first apache website. Im so proud of you :)



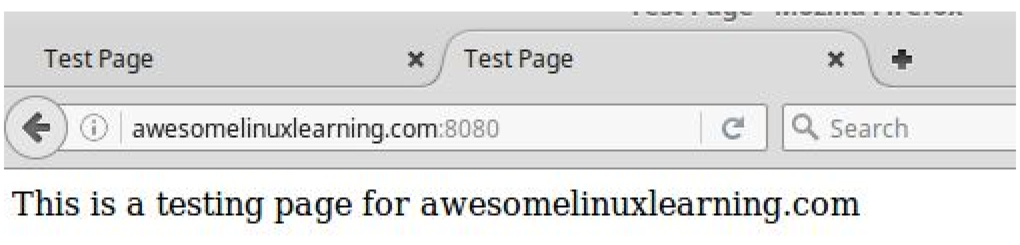
This is a testing page for awesomelinuxlearning.com

Now if we edit our **/etc/hosts** file we can do the following. Add the following line
127.0.1.1 awesomelinuxlearning


awesome - LinuxMin...
Apache2 Ubuntu Default...
Apache2 Ubuntu Default...
Apache2 Ub
awesomelinuxlearning.com
Search
Apache2 Ubuntu Default Pag
root@awesome /etc/apache2/sites-available
File
Edit
View
Search
Terminal
Help

```
  GNU nano 2.5.3              File: /etc/hosts

127.0.0.1          localhost

127.0.1.1          awesomelinuxlearning.com
#127.0.2.1         reallycool.com
```

So now why does our 000-default web page come up? Because we cannot specify the port!
Simply access it by awesomelinuxlearning.com:8080



This is a testing page for awesomelinuxlearning.com

5.4 User directory hosting

What's really annoying about the default apache installation is that everything inside /var/www/html is owned by root. So we need super user access to edit, create and read these files. This could be really annoying if you are just trying to get a development server going. There are two solutions to this.

1) We can change the owner and permissions of /var/www/html to the www-data /apache group and all of its group members, Then add our user to the www-data / apache group
2) We can tell apache to host a DocumentRoot based on user home directories.

Let's go with option 2, since I already have shown you how to change folder ownership and file permissions. (If you want to see how to do this, ask me).

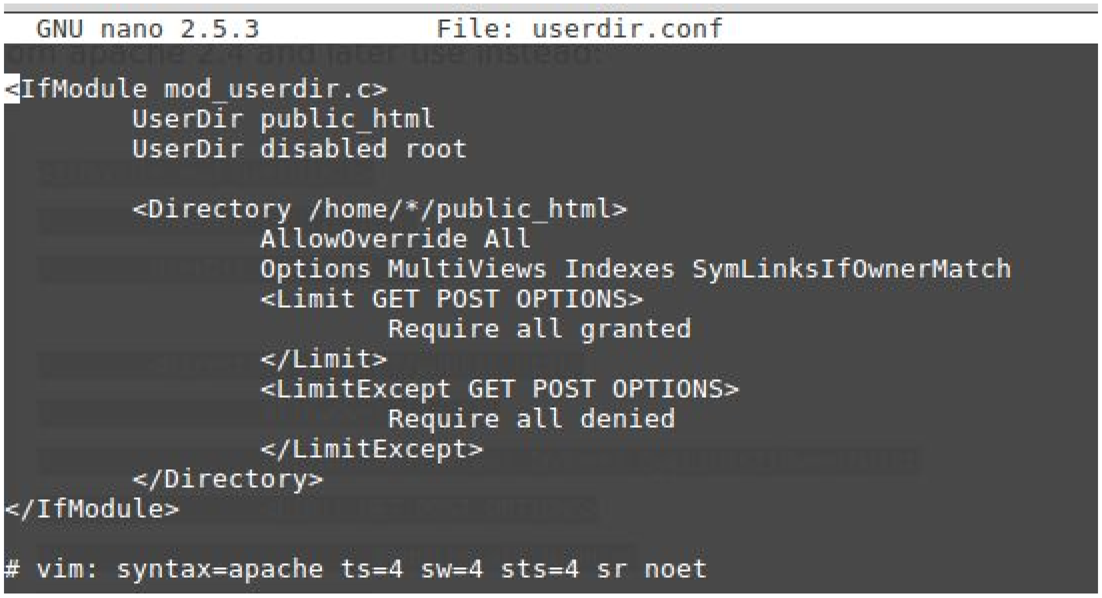
There are a few commands we can use to help manage apache. We have used one of these already **a2ensite,** there are a few others.

a2disconf : apache2 disable configuration
a2dismod: apache2 disable module
a2dissite: apache2 disable site
a2enconf: apache2 enable configuration
a2enmod: apache2 enable module
a2ensite: apache2 enable site
a2query : find values in local apache install

We need to enable the home directory module in order for our home directory hosting to function properly

```
[awesome@ sites-available]$ a2enmod userdir
```

Now we need to edit the userdir module configuration and tell it to look four hour home directories. Lets edit the file **/etc/apache2/mods-enabled/userdir.conf** and make it look exactly like this.



```
  GNU nano 2.5.3              File: userdir.conf

<IfModule mod_userdir.c>
        UserDir public_html
        UserDir disabled root

        <Directory /home/*/public_html>
                AllowOverride All
                Options MultiViews Indexes SymLinksIfOwnerMatch
                <Limit GET POST OPTIONS>
                        Require all granted
                </Limit>
                <LimitExcept GET POST OPTIONS>
                        Require all denied
                </LimitExcept>
        </Directory>
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Now Lets create a public_html folder inside a user home directory. **Without using sudo or root**

```
[lu@awesome ~]$ mkdir ~/home/lu/public_html
```

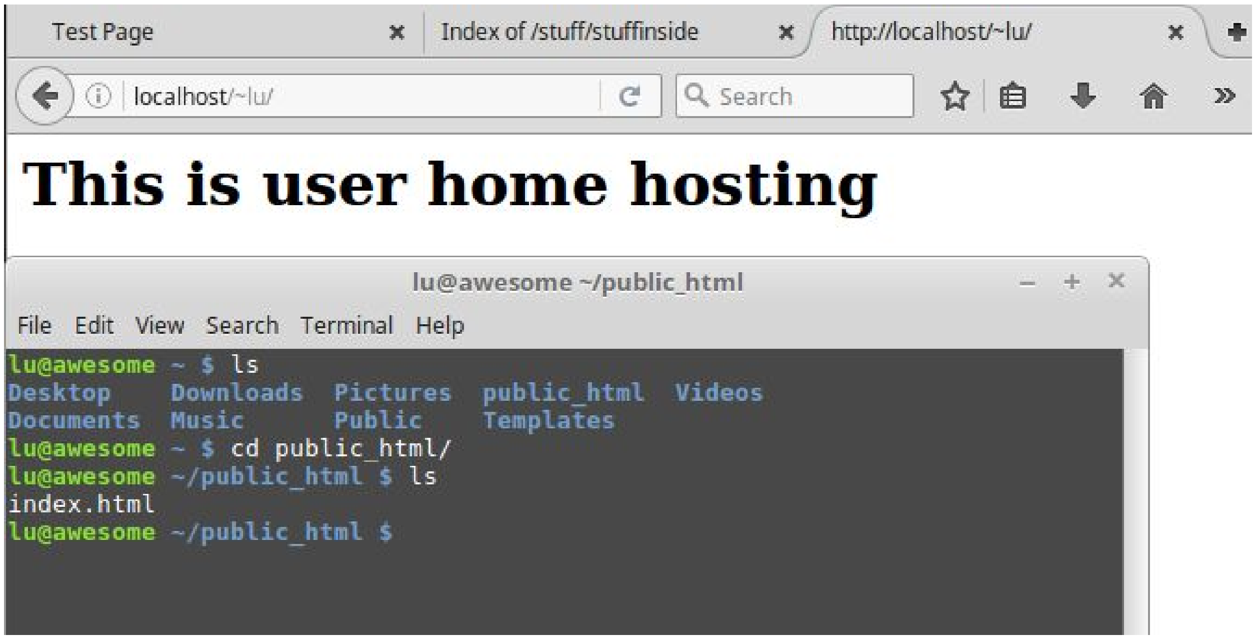
Change permissions

```
[lu@awesome ~]$ sudo chgrp www-data /home/lu/public_html
[lu@awesome ~]$ touch /home/lu/public_html/index.html
[lu@awesome ~]$ sudo chmod -R 755 /home/lu/public_html
```

Restart server

```
[lu@awesome ~]$ sudo apachectl restart
```

Navigate to localhost/~YOURUSERNAME , in my case its localhost/~lu



This is user home hosting

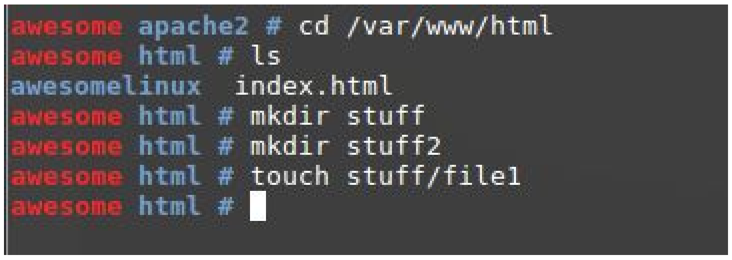
```
lu@awesome ~ $ ls
Desktop    Downloads  Pictures  public_html  Videos
Documents  Music      Public    Templates
lu@awesome ~ $ cd public_html/
lu@awesome ~/public_html $ ls
index.html
lu@awesome ~/public_html $
```

The beauty of using userdir is that once its setup, we can have hundreds of users host their content on a common domain without having any of these users mess with the /var/www/html directory. Take for example these hypothetical users, they would not need any super user access, nor be apart of the sudo group to host their content. This is one way to limit multiple users. We could even setup an FTP user with FTP accounts and have these users upload their content that way. Totally eliminating the need for these to access any other part of the file system. This is essentially how websites like Facebook, Blogger, Mail servers work.

/home/lu/public_html
/home/goku/public_html
/home/naruto/public_html
/home/spiderman/public_html
/home/buzzlightyear/public_html

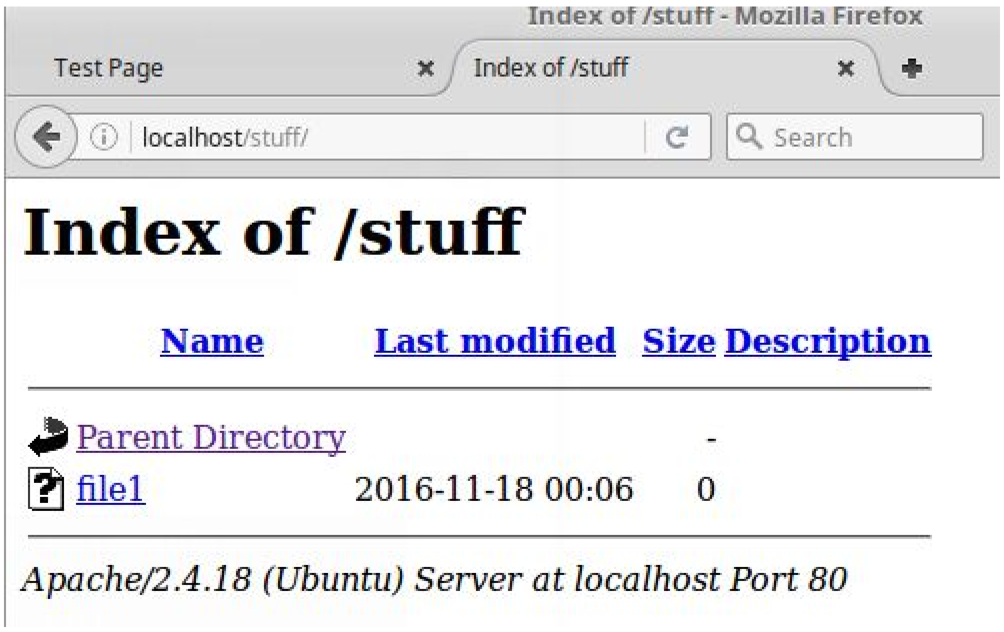
5.6 Hardening Apache

Lets create the following files. Navigate to your document root and



```
awesome apache2 # cd /var/www/html
awesome html # ls
awesomelinux  index.html
awesome html # mkdir stuff
awesome html # mkdir stuff2
awesome html # touch stuff/file1
awesome html # 
```

Now navigate to locahost/stuff



Index of /stuff

Name	Last modified	Size	Description
Parent Directory		-	
file1	2016-11-18 00:06	0	

Apache/2.4.18 (Ubuntu) Server at localhost Port 80

There are a couple of major security issues on this web page.

1) File1 can be downloaded if it is clicked on
2) Apache Index is enabled and that is a big NO GO. If this server were to be hosting anything important, Users would be able to download everything in the directories . since indexing is allowed. Anyone can navigate through the folders and its sub folders .
3) There is the *ApacheToken* being displayed. There is no reason why we should ever have the string "***Apache/2.4.18 (Ubuntu) Server at localhost Port 80*** " Lets not give the hackers a freebe and give them the system we are running. We need to make them work for it. So there is a couple things we can do
    a) Disable indexing on apache
    b) Use .htcaccess files
    c) Lock everything down with mod_security

This will be covered in chapter 6. Securing the apache server.