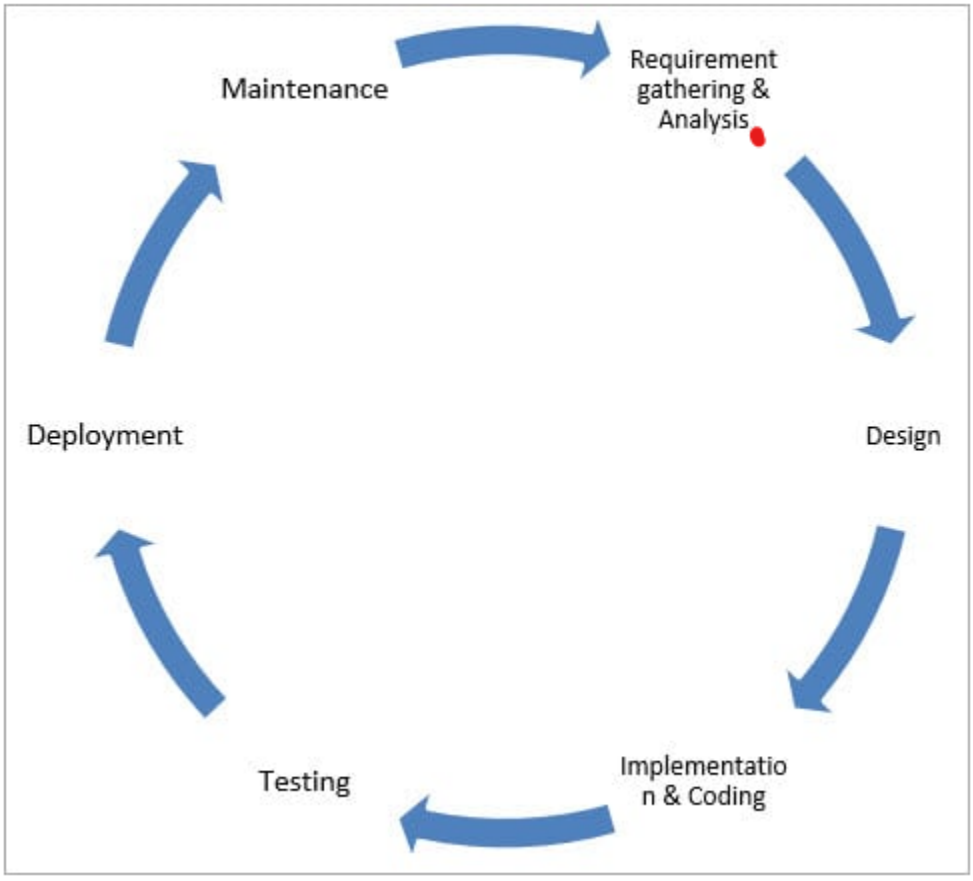
Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

SDLC Cycle represents the process of developing software.

Below is the diagrammatic representation of the SDLC cycle:



#1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product a core understanding or knowledge of the product is very important.

For Example, A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.

Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

#2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

#3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

#4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

Retesting, regression testing is done until the point at which the software is as per the customer's expectation. Testers refer SRS document to make sure that the software is as per the customer's standard.

#5) Deployment

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

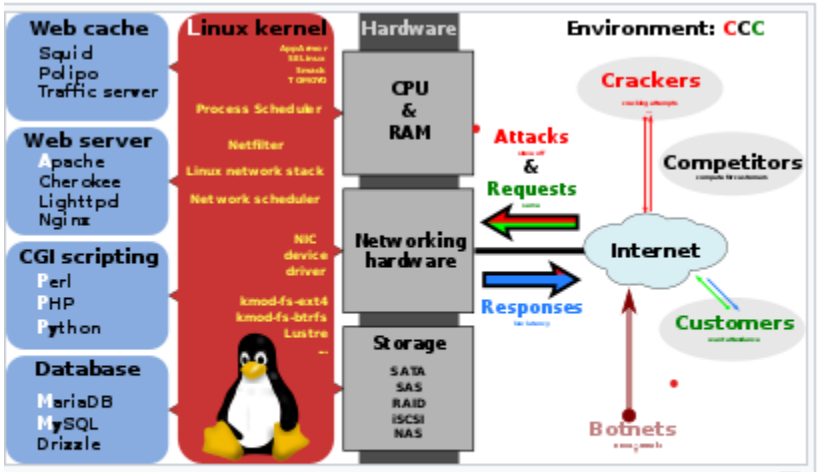
#6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

LAMP (software bundle)

LAMP (**L**inux, **A**pache, **M**ySQL, **P**HP/**P**erl/**P**ython) is a very common example of a web service stack, named as an acronym of the names of its original four open-source components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building dynamic web sites and web applications.

Since its creation, the LAMP model has been adapted to other componentry, though typically consisting of free and open-source software. For example, an equivalent installation on the Microsoft Windows family of operating systems is known as WAMP and an equivalent installation on macOS is known as MAMP.



A high-level overview of LAMP's building blocks and overall system environment, displayed here in combination with optionally used web caches.

Project Summary

1. Download, install and configure Virtual Box.
   Go to https://www.virtualbox.org/wiki/Downloads and select platform package (windows host in my case)
   Double click to install the downloaded file.
   Open the Virtual box application

2. Download, install and configure Ubuntu 20.04
   Search for Ubuntu download and select version 20.04
   On the virtual Box, go to tools and click on new to create a new virtual machine.
   Follow system prompts and set up memory (RAM) @2gb and virtual hard disk @ 10gb
   Load the ISO Image of Ubuntu 20.04 and complete the installation.

Step 1 — Installing Apache and Updating the Firewall

After installing the Apache2, I did not get SSH but instead CUPS. The output for $ sudo ufw app list yielded the result below..

```
emumenwa@emserver:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
```

I ran the commands below to install and enable SSH on my Apache

```
sudo apt install openssh-server
sudo systemctl status ssh
sudo ufw allow ssh
```

```
emumenwa@emserver:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: >
     Active: active (running) since Thu 2020-12-17 04:56:29 EST; 24h ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 684 (sshd)
      Tasks: 1 (limit: 2319)
     Memory: 2.0M
     CGroup: /system.slice/ssh.service
             └─684 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Dec 17 04:56:28 emserver systemd[1]: Starting OpenBSD Secure Shell server...
Dec 17 04:56:29 emserver sshd[684]: Server listening on 0.0.0.0 port 22.
Dec 17 04:56:29 emserver sshd[684]: Server listening on :: port 22.
Dec 17 04:56:29 emserver systemd[1]: Started OpenBSD Secure Shell server.
lines 1-15/15 (END)
```

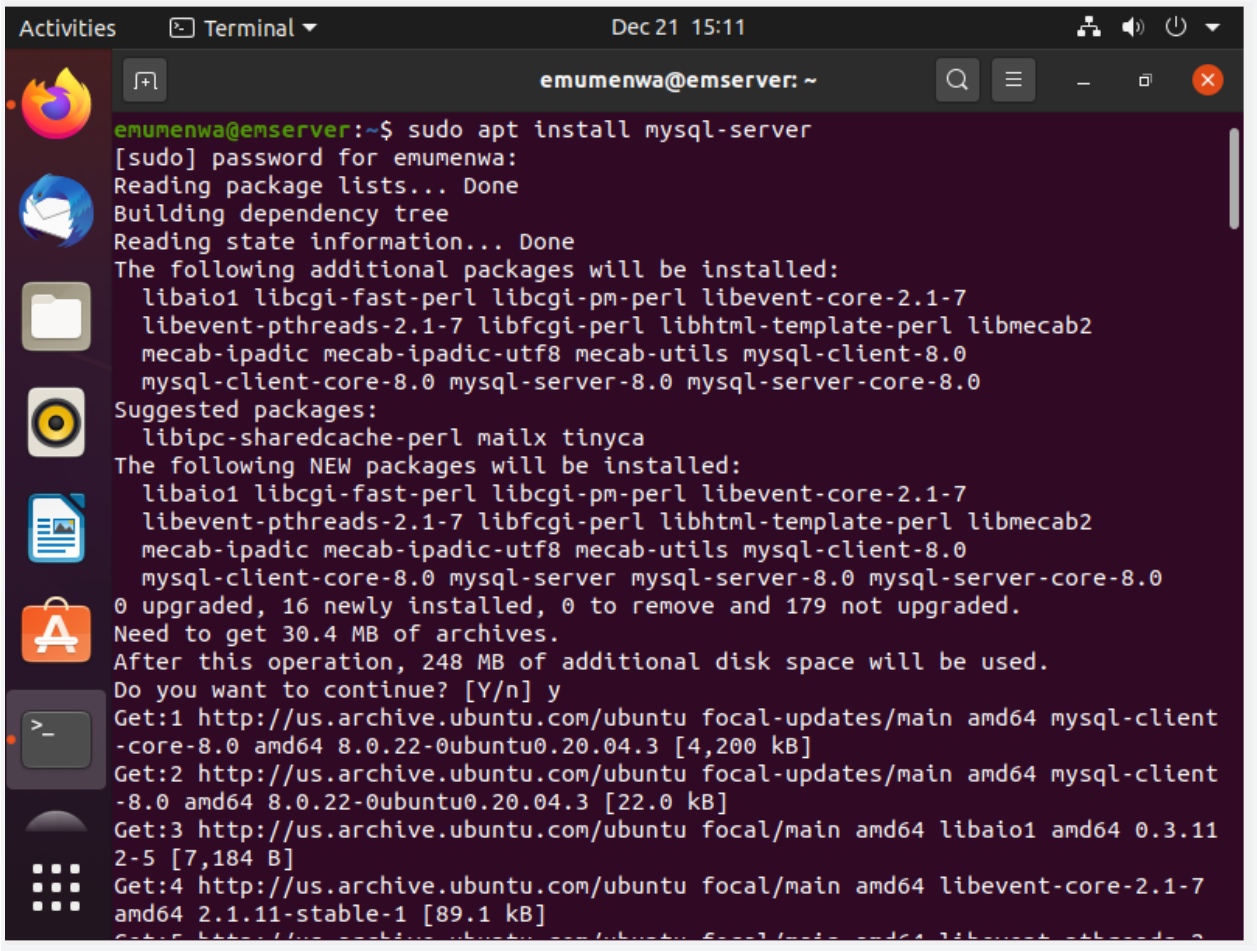
```
emumenwa@emserver:~$ sudo ufw allow ssh
Skipping adding existing rule
Skipping adding existing rule (v6)
emumenwa@emserver:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
Apache                     ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
Apache (v6)                ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)

emumenwa@emserver:~$
```

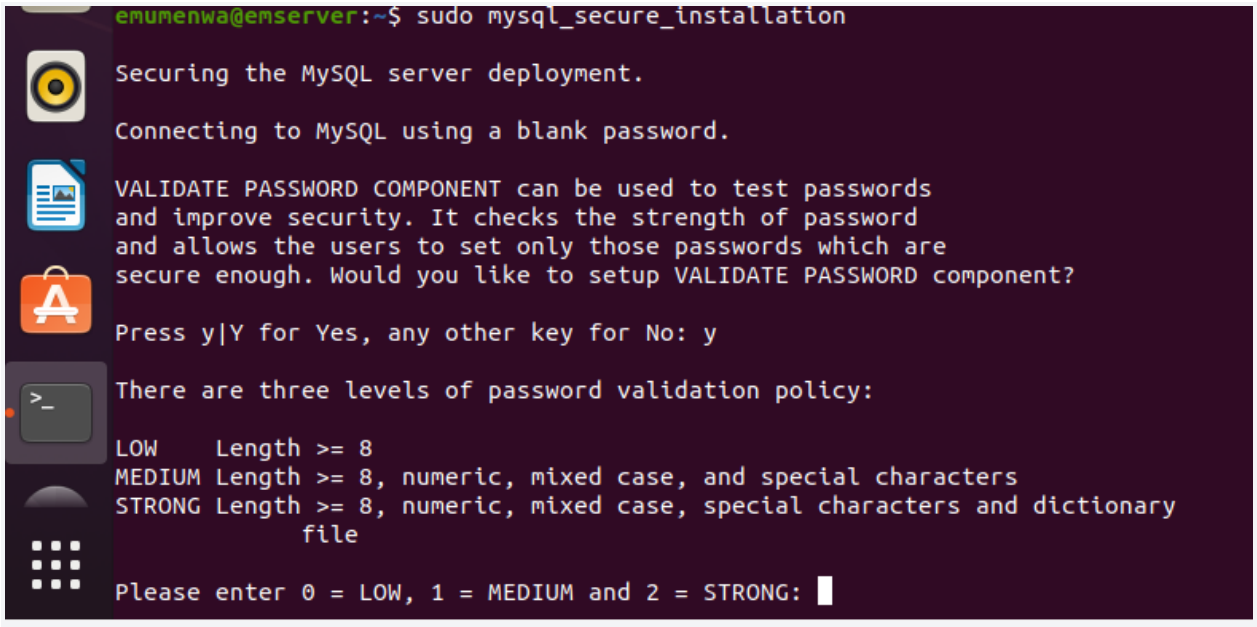
```
$ hostname -I
10.0.2.15
```

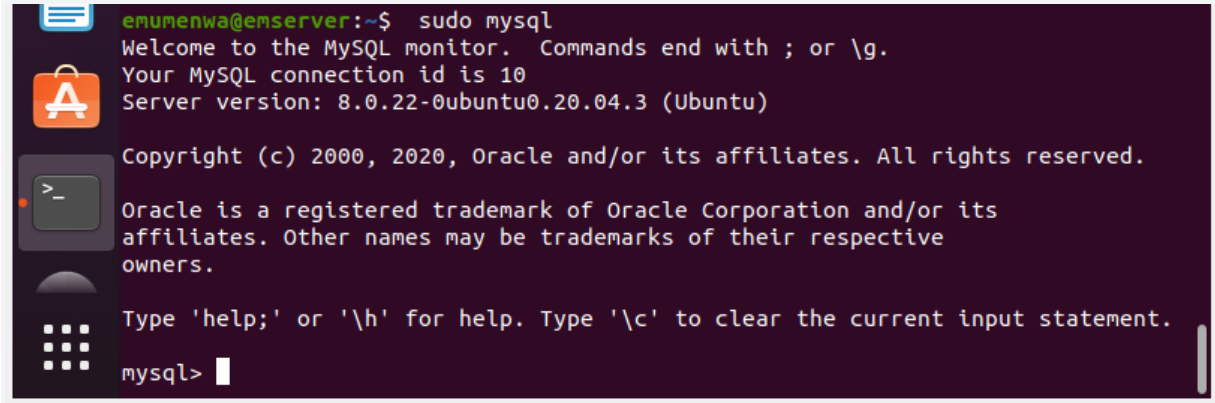
I entered the IP address `http://10.0.2.15/` on my browser to get the result below


Activities
Firefox Web Browser
Dec 21 14:36
Apache2 Ubuntu Default Pa
10.0.2.15
Apache2 Ubuntu Default Page
ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server afte
installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the U
Apache packaging is derived. If you can read this page, it means that the Apache HTTP server
at this site is working properly. You should replace this file (located at /var/www/html/index
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
that the site is currently unavailable due to maintenance. If the problem persists, please conta
site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, a
into several files optimized for interaction with Ubuntu tools. The configuration system is fully
documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full
documentation. Documentation for the web server itself can be found by accessing the manua
apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follow
/etc/apache2/

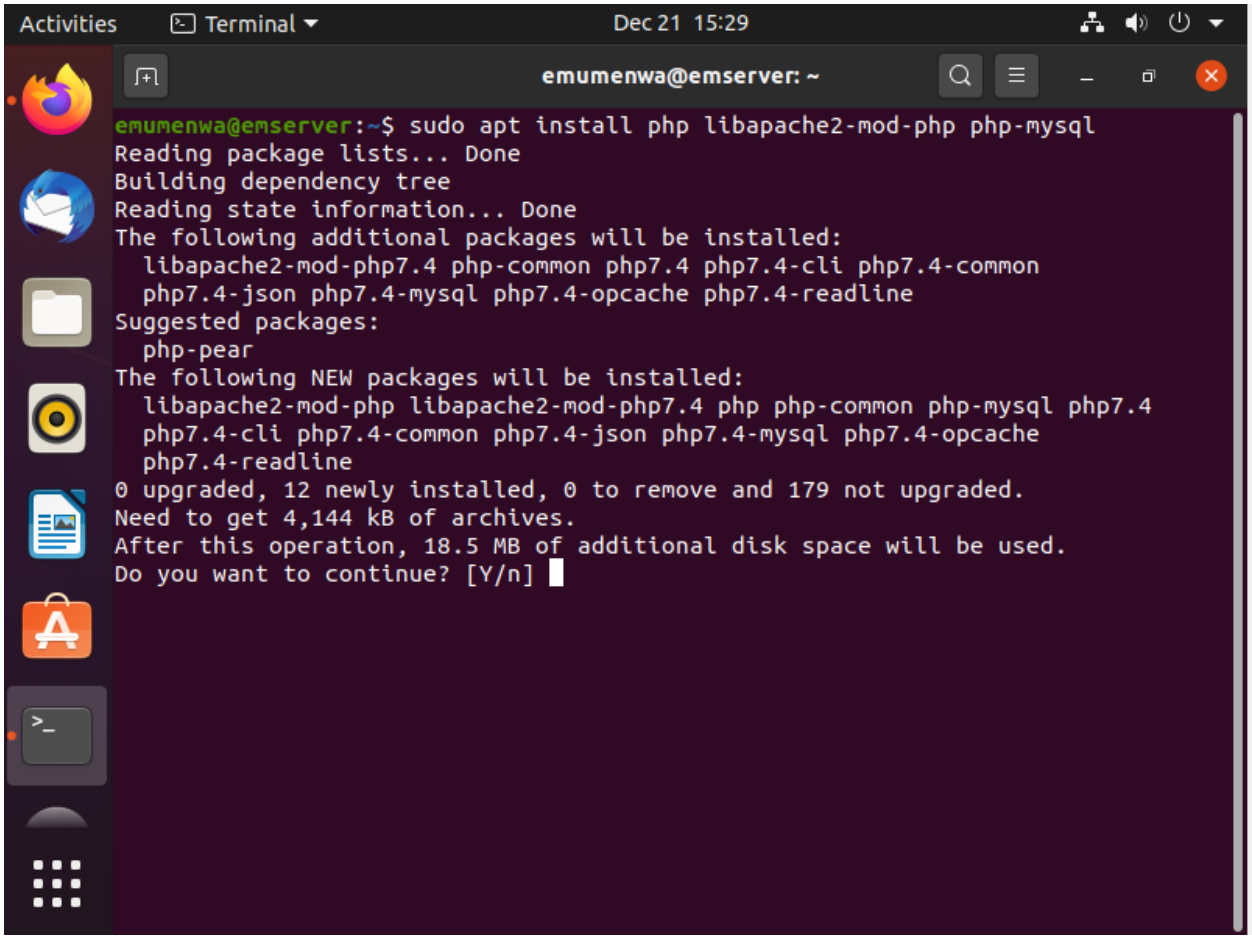
Step 2 — Installing MySQL

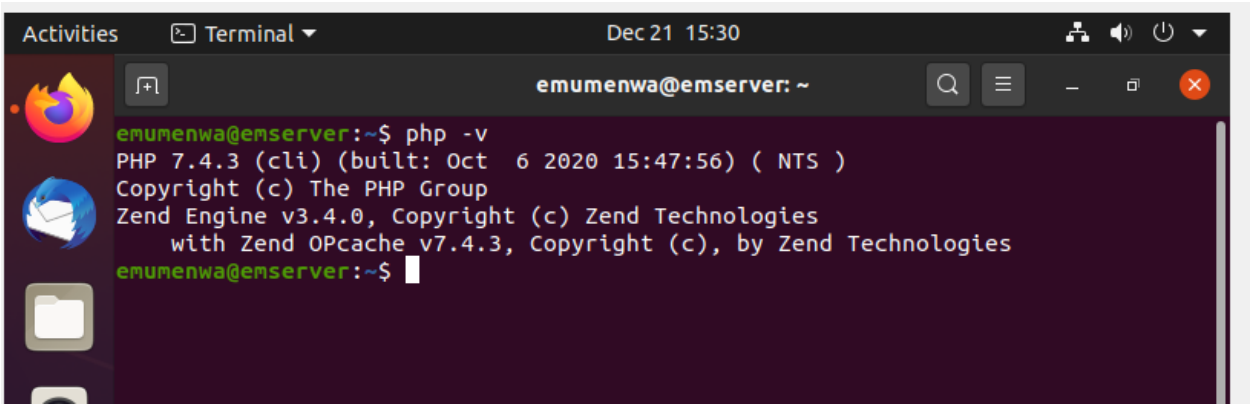






Step 3 — Installing PHP

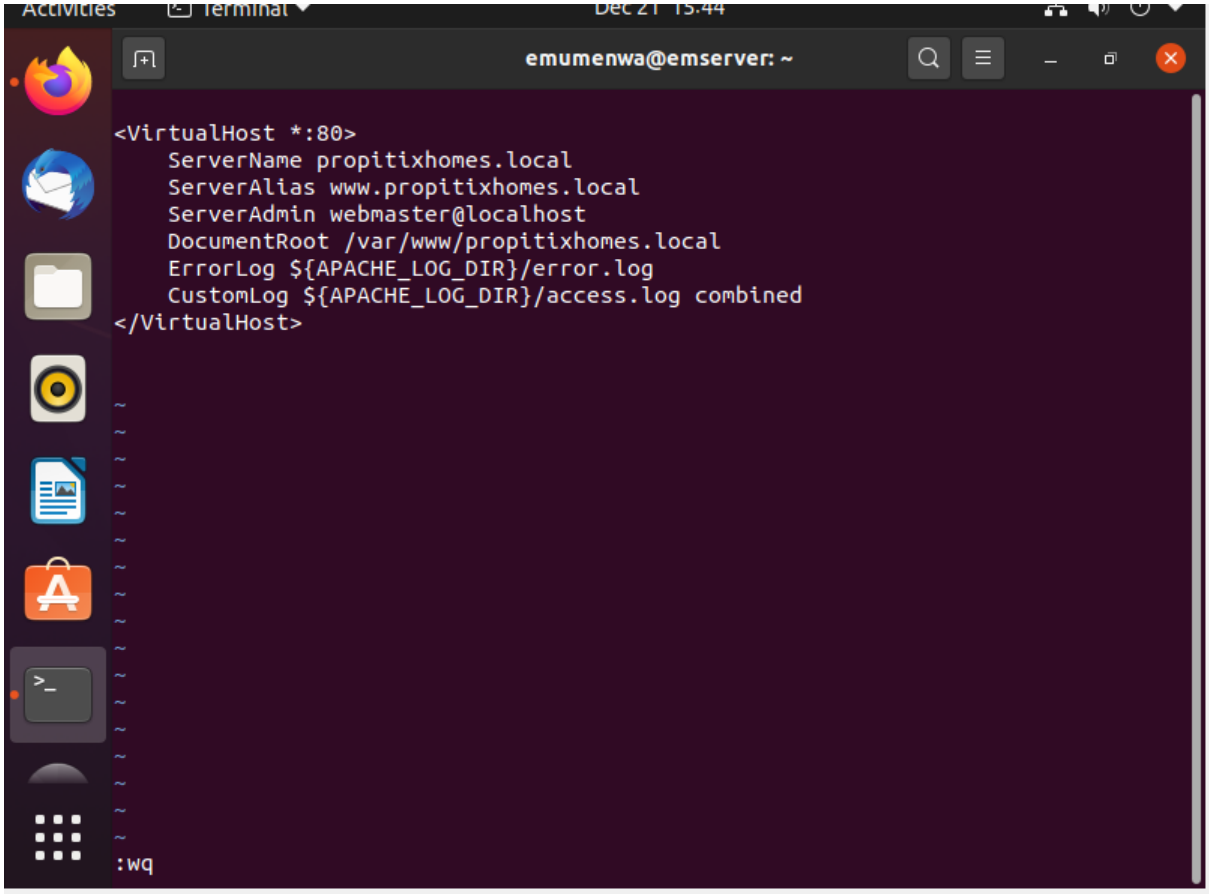




Step 4 — Creating a Virtual Host for your Website using Apache

```
emumenwa@emserver:~$ sudo mkdir /var/www/propitixhomes.local
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ sudo chown -R $USER:$USER /var/www/propitixhomes.local
emumenwa@emserver:~$
```




```
emumenwa@emserver:~$ sudo ls /etc/apache2/sites-available
000-default.conf  default-ssl.conf  propitixhomes.local.conf
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ sudo a2ensite propitixhomes.local
Enabling site propitixhomes.local.
To activate the new configuration, you need to run:
  systemctl reload apache2
emumenwa@emserver:~$
```

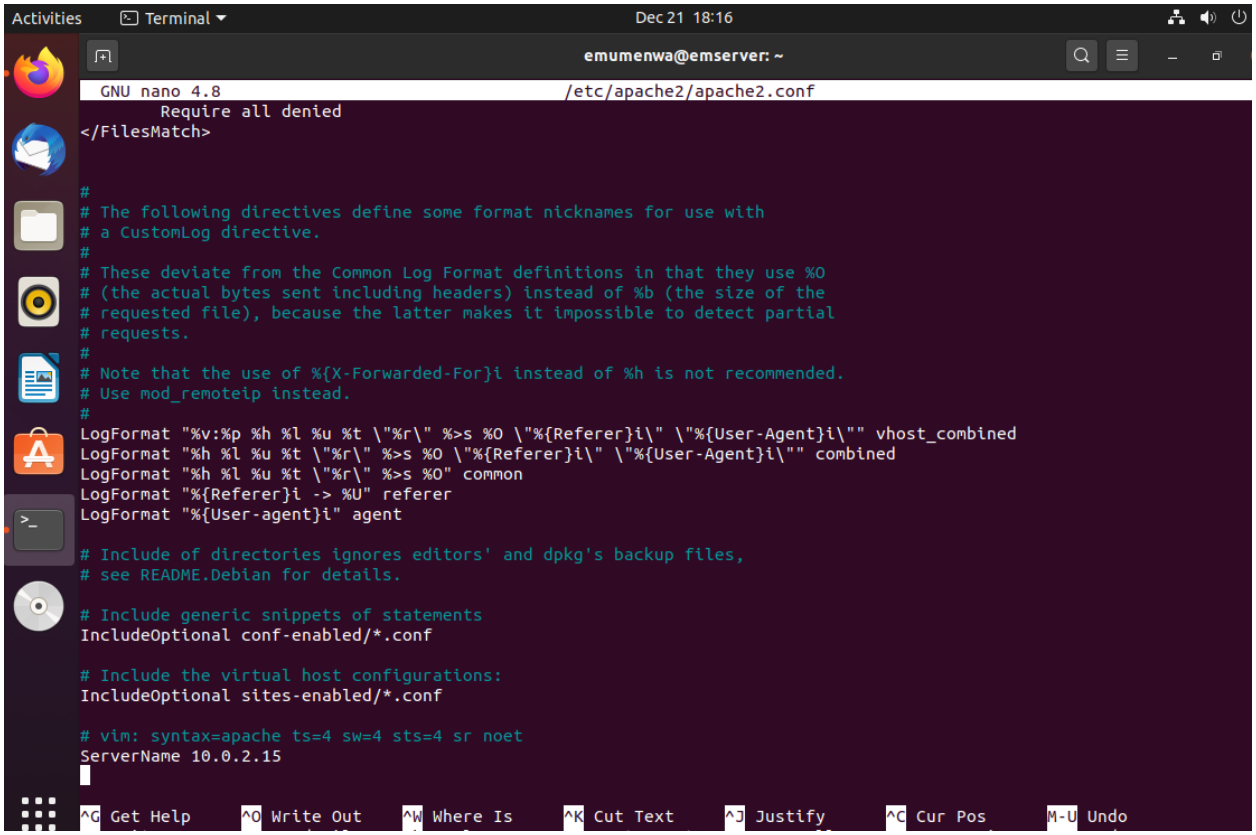
```
emumenwa@emserver:~$ sudo a2ensite propitixhomes.local
[sudo] password for emumenwa:
Site propitixhomes.local already enabled
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ sudo a2dissite 000-default
Site 000-default already disabled
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualified dom
ain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress
this message
Syntax OK
emumenwa@emserver:~$
```

To clear the message above, I ran the command shown next added a line containing the server name

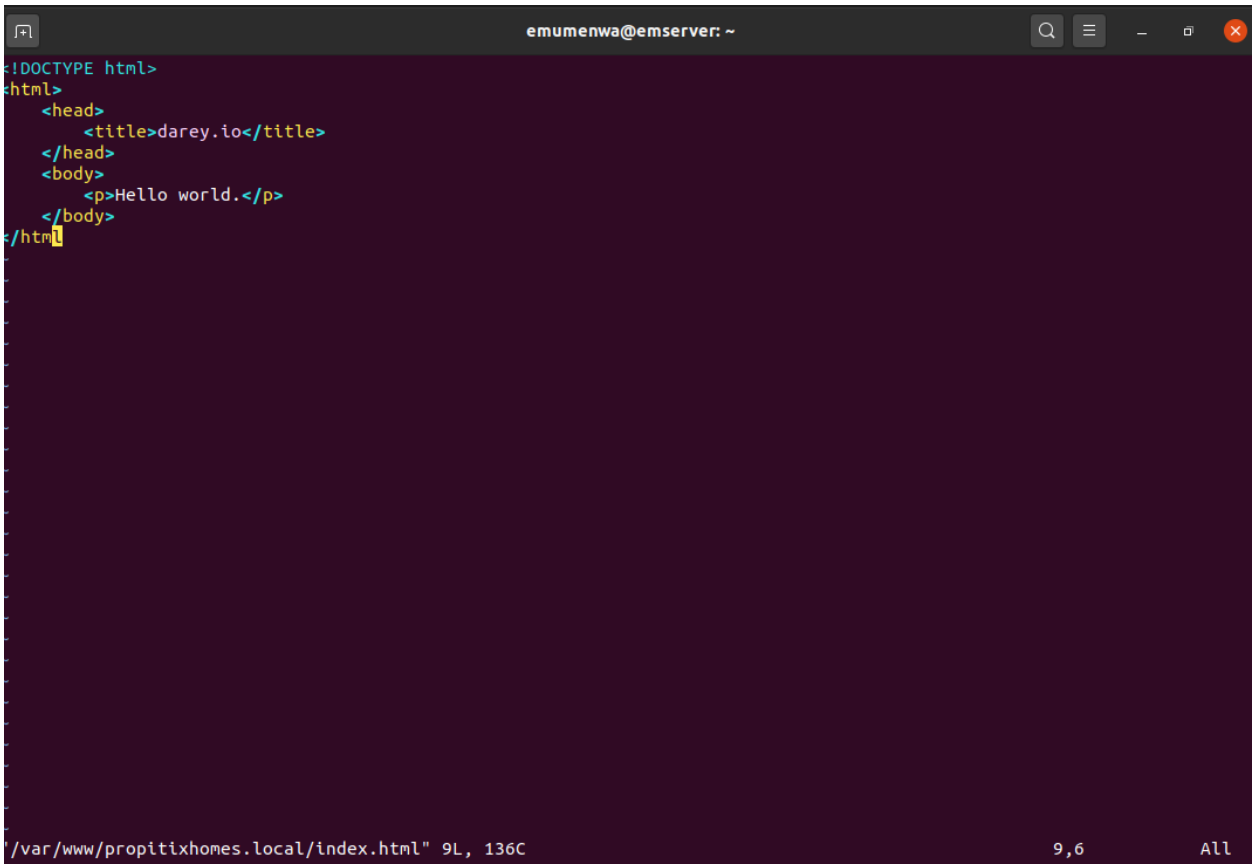
```
emumenwa@emserver:~$ sudo nano /etc/apache2/apache2.conf
emumenwa@emserver:~$
```

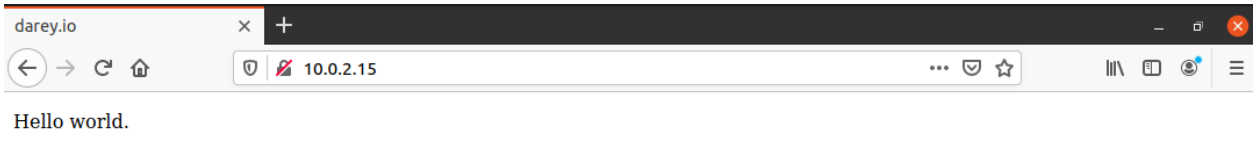


```
emumenwa@emserver:~$ sudo apache2ctl configtest
Syntax OK
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ sudo systemctl reload apache2
emumenwa@emserver:~$
```

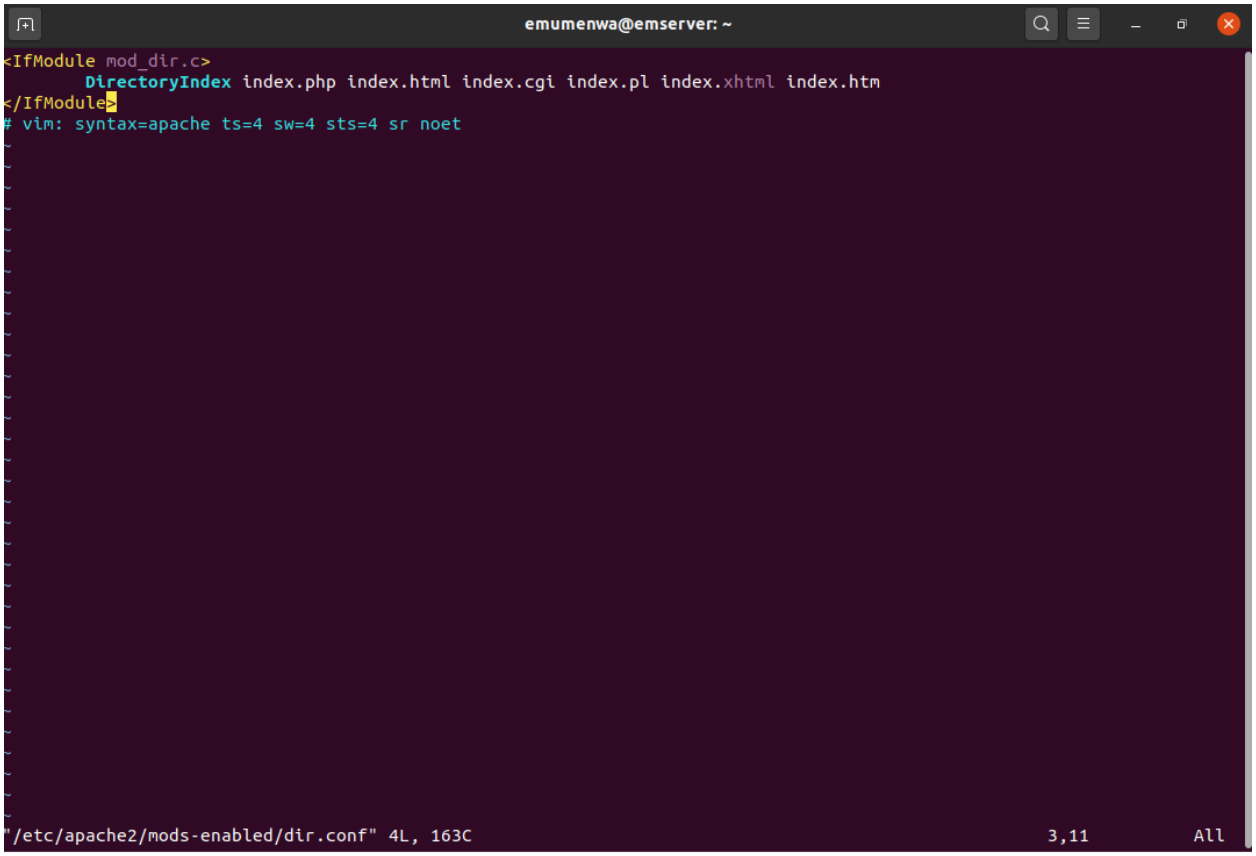
emumenwa@emserver:~$ vi /var/www/propitixhomes.local/index.html





Hello world.

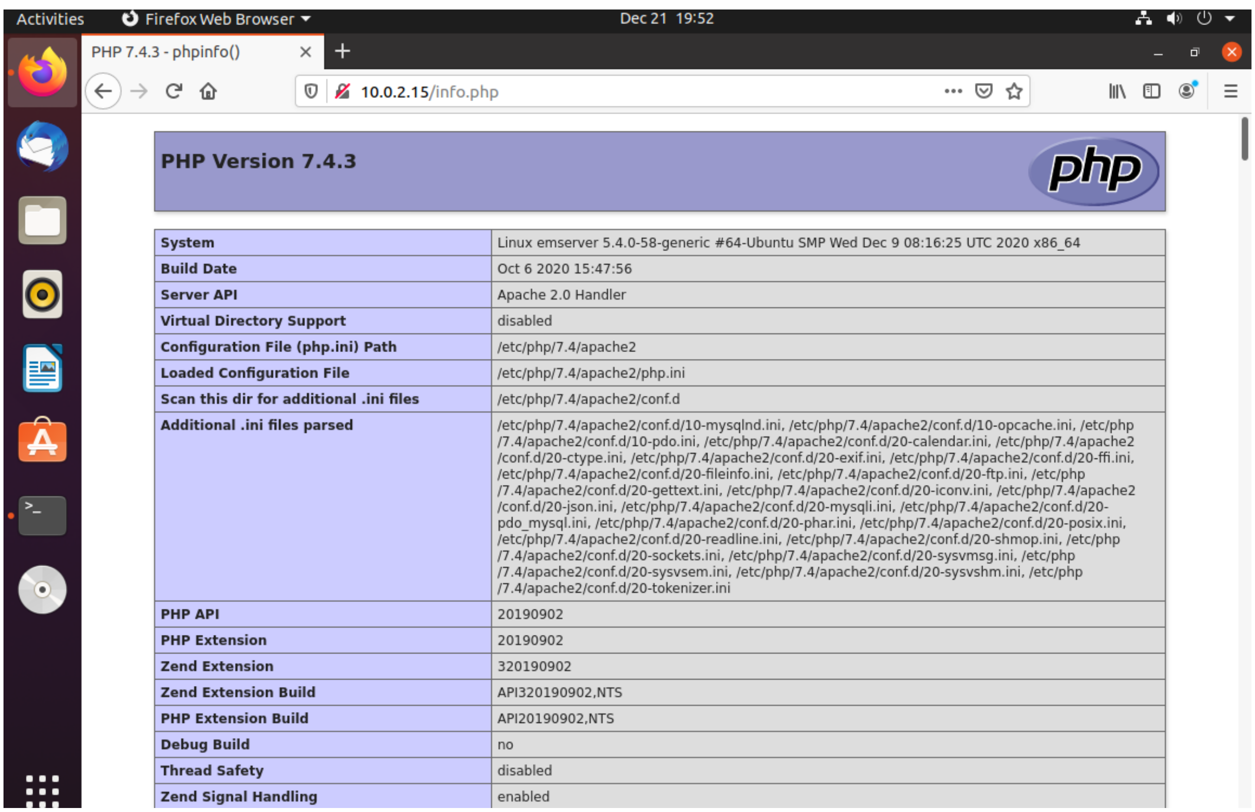
```
emumenwa@emserver:~$ sudo vim /etc/apache2/mods-enabled/dir.conf
[sudo] password for emumenwa:
emumenwa@emserver:~$
```




```
emumenwa@emserver:~$ sudo systemctl reload apache2
emumenwa@emserver:~$
```

```
emumenwa@emserver:~$ vim /var/www/propitixhomes.local/info.php
```


<?php
phpinfo();
"/var/www/propitixhomes.local/info.php" 2L, 17C
2,10
All


Activities
Firefox Web Browser
Dec 21 19:52
PHP 7.4.3 - phpinfo()
10.0.2.15/info.php
PHP Version 7.4.3
php
System
Linux emserver 5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020 x86_64
Build Date
Oct 6 2020 15:47:56
Server API
Apache 2.0 Handler
Virtual Directory Support
disabled
Configuration File (php.ini) Path
/etc/php/7.4/apache2
Loaded Configuration File
/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files
/etc/php/7.4/apache2/conf.d
Additional .ini files parsed
/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini
PHP API
20190902
PHP Extension
20190902
Zend Extension
320190902
Zend Extension Build
API320190902,NTS
PHP Extension Build
API20190902,NTS
Debug Build
no
Thread Safety
disabled
Zend Signal Handling
enabled

```
emumenwa@emserver:~$ sudo rm /var/www/propitixhomes.local/info.php
[sudo] password for emumenwa:
emumenwa@emserver:~$
```