Project 1: LAMP Stack Implementation

**Step -1- Installing Apache and updating firewall**

Graphical user interface, text, email

Description automatically generated

What exactly is Apache?

[Apache HTTP Server](https://httpd.apache.org) is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules. Most WordPress hosting providers use Apache as their web server software. However, websites and other applications can run on other web server software as well. Such as [Nginx](https://www.nginx.com), [Microsoft’s IIS](https://www.iis.net), etc.

The Apache web server is among the most popular web servers in the world. It’s well documented, has an active community of users, and has been in wide use for much of the history of the web, which makes it a great default choice for hosting a website.

Install Apache using Ubuntu’s package manager [***‘apt’***](https://en.wikipedia.org/wiki/APT_(software)):

#update a list of packages in package manager

sudo apt update

#run apache2 package installation

sudo apt install apache2

Text

Description automatically generated

To verify that apache2 is running as a Service in our OS, use following command

sudo systemctl status apache2

If it is green and running, then you did everything correctly – you have just launched your first Web Server in the Clouds!

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

First, let us try to check how we can access it locally in our Ubuntu shell, run:

curl http://localhost:80

Output: As an output you can see some strangely formatted text

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<!--

Modified from the Debian original for Ubuntu

Last updated: 2016-11-16

See: https://launchpad.net/bugs/1288690

-->

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Apache2 Ubuntu Default Page: It works</title>

<style type="text/css" media="screen">

\* {

margin: 0px 0px 0px 0px;

padding: 0px 0px 0px 0px;

}

body, html {

padding: 3px 3px 3px 3px;

background-color: #D8DBE2;

font-family: Verdana, sans-serif;

font-size: 11pt;

text-align: center;

}

div.main\_page {

position: relative;

display: table;

width: 800px;

margin-bottom: 3px;

margin-left: auto;

margin-right: auto;

padding: 0px 0px 0px 0px;

border-width: 2px;

border-color: #212738;

border-style: solid;

background-color: #FFFFFF;

text-align: center;

}

div.page\_header {

height: 99px;

width: 100%;

background-color: #F5F6F7;

}

div.page\_header span {

margin: 15px 0px 0px 50px;

font-size: 180%;

font-weight: bold;

}

div.page\_header img {

margin: 3px 0px 0px 40px;

border: 0px 0px 0px;

}

div.table\_of\_contents {

clear: left;

min-width: 200px;

margin: 3px 3px 3px 3px;

background-color: #FFFFFF;

text-align: left;

}

div.table\_of\_contents\_item {

clear: left;

width: 100%;

margin: 4px 0px 0px 0px;

background-color: #FFFFFF;

color: #000000;

text-align: left;

}

div.table\_of\_contents\_item a {

margin: 6px 0px 0px 6px;

}

div.content\_section {

margin: 3px 3px 3px 3px;

background-color: #FFFFFF;

text-align: left;

}

div.content\_section\_text {

padding: 4px 8px 4px 8px;

color: #000000;

font-size: 100%;

}

div.content\_section\_text pre {

margin: 8px 0px 8px 0px;

padding: 8px 8px 8px 8px;

border-width: 1px;

border-style: dotted;

border-color: #000000;

background-color: #F5F6F7;

font-style: italic;

}

div.content\_section\_text p {

margin-bottom: 6px;

}

div.content\_section\_text ul, div.content\_section\_text li {

padding: 4px 8px 4px 16px;

}

div.section\_header {

padding: 3px 6px 3px 6px;

background-color: #8E9CB2;

color: #FFFFFF;

font-weight: bold;

font-size: 112%;

text-align: center;

}

div.section\_header\_red {

background-color: #CD214F;

}

div.section\_header\_grey {

background-color: #9F9386;

}

.floating\_element {

position: relative;

float: left;

}

div.table\_of\_contents\_item a,

div.content\_section\_text a {

text-decoration: none;

font-weight: bold;

}

div.table\_of\_contents\_item a:link,

div.table\_of\_contents\_item a:visited,

div.table\_of\_contents\_item a:active {

color: #000000;

}

div.table\_of\_contents\_item a:hover {

background-color: #000000;

color: #FFFFFF;

}

div.content\_section\_text a:link,

div.content\_section\_text a:visited,

div.content\_section\_text a:active {

background-color: #DCDFE6;

color: #000000;

}

div.content\_section\_text a:hover {

background-color: #000000;

color: #DCDFE6;

}

div.validator {

}

</style>

</head>

<body>

<div class="main\_page">

<div class="page\_header floating\_element">

<img src="/icons/ubuntu-logo.png" alt="Ubuntu Logo" class="floating\_element"/>

<span class="floating\_element">

Apache2 Ubuntu Default Page

</span>

</div>

<!-- <div class="table\_of\_contents floating\_element">

<div class="section\_header section\_header\_grey">

TABLE OF CONTENTS

</div>

<div class="table\_of\_contents\_item floating\_element">

<a href="#about">About</a>

</div>

<div class="table\_of\_contents\_item floating\_element">

<a href="#changes">Changes</a>

</div>

<div class="table\_of\_contents\_item floating\_element">

<a href="#scope">Scope</a>

</div>

<div class="table\_of\_contents\_item floating\_element">

<a href="#files">Config files</a>

</div>

</div>

-->

<div class="content\_section floating\_element">

<div class="section\_header section\_header\_red">

<div id="about"></div>

It works!

</div>

<div class="content\_section\_text">

<p>

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 Ubuntu 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 <b>replace this file</b> (located at

<tt>/var/www/html/index.html</tt>) before continuing to operate your HTTP server.

</p>

<p>

If you are a normal user of this web site and don't know what this page is

about, this probably means that the site is currently unavailable due to

maintenance.

If the problem persists, please contact the site's administrator.

</p>

</div>

<div class="section\_header">

<div id="changes"></div>

Configuration Overview

</div>

<div class="content\_section\_text">

<p>

Ubuntu's Apache2 default configuration is different from the

upstream default configuration, and split into several files optimized for

interaction with Ubuntu tools. The configuration system is

<b>fully documented in

/usr/share/doc/apache2/README.Debian.gz</b>. Refer to this for the full

documentation. Documentation for the web server itself can be

found by accessing the <a href="/manual">manual</a> if the <tt>apache2-doc</tt>

package was installed on this server.

</p>

<p>

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

</p>

<pre>

/etc/apache2/

|-- apache2.conf

| `-- ports.conf

|-- mods-enabled

| |-- \*.load

| `-- \*.conf

|-- conf-enabled

| `-- \*.conf

|-- sites-enabled

| `-- \*.conf

</pre>

<ul>

<li>

<tt>apache2.conf</tt> is the main configuration

file. It puts the pieces together by including all remaining configuration

files when starting up the web server.

</li>

<li>

<tt>ports.conf</tt> is always included from the

main configuration file. It is used to determine the listening ports for

incoming connections, and this file can be customized anytime.

</li>

<li>

Configuration files in the <tt>mods-enabled/</tt>,

<tt>conf-enabled/</tt> and <tt>sites-enabled/</tt> directories contain

particular configuration snippets which manage modules, global configuration

fragments, or virtual host configurations, respectively.

</li>

<li>

They are activated by symlinking available

configuration files from their respective

\*-available/ counterparts. These should be managed

by using our helpers

<tt>

a2enmod,

a2dismod,

</tt>

<tt>

a2ensite,

a2dissite,

</tt>

and

<tt>

a2enconf,

a2disconf

</tt>. See their respective man pages for detailed information.

</li>

<li>

The binary is called apache2. Due to the use of

environment variables, in the default configuration, apache2 needs to be

started/stopped with <tt>/etc/init.d/apache2</tt> or <tt>apache2ctl</tt>.

<b>Calling <tt>/usr/bin/apache2</tt> directly will not work</b> with the

default configuration.

</li>

</ul>

</div>

<div class="section\_header">

<div id="docroot"></div>

Document Roots

</div>

<div class="content\_section\_text">

<p>

By default, Ubuntu does not allow access through the web browser to

<em>any</em> file apart of those located in <tt>/var/www</tt>,

<a href="http://httpd.apache.org/docs/2.4/mod/mod\_userdir.html" rel="nofollow">public\_html</a>

directories (when enabled) and <tt>/usr/share</tt> (for web

applications). If your site is using a web document root

located elsewhere (such as in <tt>/srv</tt>) you may need to whitelist your

document root directory in <tt>/etc/apache2/apache2.conf</tt>.

</p>

<p>

The default Ubuntu document root is <tt>/var/www/html</tt>. You

can make your own virtual hosts under /var/www. This is different

to previous releases which provides better security out of the box.

</p>

</div>

<div class="section\_header">

<div id="bugs"></div>

Reporting Problems

</div>

<div class="content\_section\_text">

<p>

Please use the <tt>ubuntu-bug</tt> tool to report bugs in the

Apache2 package with Ubuntu. However, check <a

href="https://bugs.launchpad.net/ubuntu/+source/apache2"

rel="nofollow">existing bug reports</a> before reporting a new bug.

</p>

<p>

Please report bugs specific to modules (such as PHP and others)

to respective packages, not to the web server itself.

</p>

</div>

</div>

</div>

<div class="validator">

</div>

</body>

</html>

Graphical user interface, text, application

Description automatically generated

Another way to retrieve your Public IP address, other than to check it in AWS Web console, is to use following command:

curl -s http://169.254.169.254/latest/meta-data/public-ipv4

**Step -2- Installing MySQL**

Now that you have a web server up and running, you need to install a [Database Management System (DBMS)](https://en.wikipedia.org/wiki/Database#Database_management_system) to be able to store and manage data for your site in a [relational database](https://en.wikipedia.org/wiki/Relational_database). [MySQL](https://www.mysql.com) is a popular relational database management system used within PHP environments, so we will use it in our project.

Again, use ‘apt’ to acquire and install this software:

$ sudo apt install mysql-server

When prompted, confirm installation by typing Y, and then ENTER.

When the installation is finished, log in to the MySQL console by typing:

$ sudo mysql

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of sudo when running this command.

Text

Description automatically generated

It’s recommended that you run a security script that comes pre-installed with MySQL. This script will remove some insecure default settings and lock down access to your database system. Before running the script you will set a password for the **root** user, using mysql\_native\_password as default authentication method. We’re defining this user’s password as PassWord.1.

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'PassWord.1';



Exit the MySQL shell with:

mysql> exit

Start the interactive script by running:

$ sudo mysql\_secure\_installation

This will ask if you want to configure the VALIDATE PASSWORD PLUGIN.

**Note**: Enabling this feature is something of a judgment call. If enabled, passwords which don’t match the specified criteria will be rejected by MySQL with an error. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer Y for yes, or anything else to continue without enabling.

VALIDATE PASSWORD PLUGIN can be used to test passwords

and improve security. It checks the strength of password

and allows the users to set only those passwords which are

secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:

If you answer “yes”, you’ll be asked to select a level of password validation. Keep in mind that if you enter 2 for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words e.g PassWord.1.

There are three levels of password validation policy:

LOW Length >= 8

MEDIUM Length >= 8, numeric, mixed case, and special characters

STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1

Regardless of whether you chose to set up the VALIDATE PASSWORD PLUGIN, your server will next ask you to select and confirm a password for the MySQL **root** user. This is not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Even though the default authentication method for the MySQL root user dispenses the use of a password, **even when one is set**, you should define a strong password here as an additional safety measure. We’ll talk about this in a moment.

If you enabled password validation, you’ll be shown the password strength for the root password you just entered and your server will ask if you want to continue with that password. If you are happy with your current password, enter Y for “yes” at the prompt:

Estimated strength of the password: 100

Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y

For the rest of the questions, press Y and hit the ENTER key at each prompt. This will prompt you to change the root password, remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

When you’re finished, test if you’re able to log in to the MySQL console by typing:

$ sudo mysql -p

Text, letter

Description automatically generated

Text

Description automatically generated

Notice the -p flag in this command, which will prompt you for the password used after changing the **root** user password.

For increased security, it’s best to have dedicated user accounts with less expansive privileges set up for every database, especially if you plan on having multiple databases hosted on your server.

**Note**: At the time of this writing, the native MySQL PHP library mysqlnd doesn’t support caching\_sha2\_authentication, the default authentication method for MySQL 8. For that reason, when creating database users for PHP applications on MySQL 8, you’ll need to make sure they’re configured to use mysql\_native\_password instead.

Your MySQL server is now installed and secured. Next, we will install PHP, the final component in the LAMP stack.

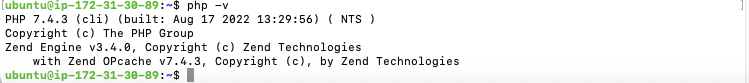
**Step -3- Installing PHP**

You have Apache installed to serve your content and MySQL installed to store and manage your data. [PHP](https://www.php.net) is the component of our setup that will process code to display dynamic content to the end user. In addition to the php package, you’ll need php-mysql, a PHP module that allows PHP to communicate with MySQL-based databases. You’ll also need libapache2-mod-php to enable Apache to handle PHP files. Core PHP packages will automatically be installed as dependencies.

To install these 3 packages at once, run:

sudo apt install php libapache2-mod-php php-mysql

Once the installation is finished, you can run the following command to confirm your PHP version:

php -v

At this point, your LAMP stack is completely installed and fully operational.

* **L**inux (Ubuntu)
* **A**pache HTTP Server
* **M**ySQL
* **P**HP

To test your setup with a PHP script, it’s best to set up a proper [Apache Virtual Host](https://httpd.apache.org/docs/2.4/vhosts/) to hold your website’s files and folders. Virtual host allows you to have multiple websites located on a single machine and users of the websites will not even notice it.

A picture containing graphical user interface

Description automatically generated

**Step -4- Creating a Virtual Host For Your Website Using Apache**

In this project, you will set up a domain called projectlamp, but you can replace this with any domain of your choice.

Apache on Ubuntu 20.04 has one server block enabled by default that is configured to serve documents from the **/var/www/html** directory.  
We will leave this configuration as is and will add our own directory next to the default one.

Create the directory for projectlamp using **‘mkdir’** command as follows:

sudo mkdir /var/www/projectlamp

Next, assign ownership of the directory with your current system user:

sudo chown -R $USER:$USER /var/www/projectlamp

Then, create and open a new configuration file in Apache’s sites-available directory using your preferred command-line editor. Here, we’ll be using vi or vim (They are the same by the way):

sudo vi /etc/apache2/sites-available/projectlamp.conf

This will create a new blank file. Paste in the following bare-bones configuration by hitting on i on the keyboard to enter the insert mode, and paste the text:

<VirtualHost \*:80>

ServerName projectlamp

ServerAlias www.projectlamp

ServerAdmin webmaster@localhost

DocumentRoot /var/www/projectlamp

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

To save and close the file, simply follow the steps below:

1. Hit the esc button on the keyboard
2. Type :
3. Type wq. **w** for write and **q** for quit
4. Hit ENTER to save the file

You can use the **ls** command to show the new file in the **sites-available** directory

sudo ls /etc/apache2/sites-available

You will see something like this;



With this VirtualHost configuration, we’re telling Apache to serve projectlamp using **/var/www/projectlamp** as its web root directory. If you would like to test Apache without a domain name, you can remove or comment out the options ServerName and ServerAlias by adding a **#** character in the beginning of each option’s lines. Adding the **#** character there will tell the program to skip processing the instructions on those lines.

You can now use ***a2ensite*** command to enable the new virtual host:

sudo a2ensite projectlamp

You might want to disable the default website that comes installed with Apache. This is required if you’re not using a custom domain name, because in this case Apache’s default configuration would overwrite your virtual host. To disable Apache’s default website use ***a2dissite*** command , type:

sudo a2dissite 000-default

To make sure your configuration file doesn’t contain syntax errors, run:

sudo apache2ctl configtest

Finally, reload Apache so these changes take effect:

sudo systemctl reload apache2

Text

Description automatically generated

Your new website is now active, but the web root **/var/www/projectlamp** is still empty. Create an index.html file in that location so that we can test that the virtual host works as expected:

sudo echo 'Hello LAMP from hostname' $(curl -s http://169.254.169.254/latest/meta-data/public-hostname) 'with public IP' $(curl -s http://169.254.169.254/latest/meta-data/public-ipv4) > /var/www/projectlamp/index.html

Now go to your browser and try to open your website URL using IP address:

http://<Public-IP-Address>:80

Graphical user interface, text, application

Description automatically generated

If you see the text from ***‘echo’*** command you wrote to index.html file, then it means your Apache virtual host is working as expected.  
In the output you will see your server’s public hostname (DNS name) and public IP address. You can also access your website in your browser by public DNS name, not only by IP – try it out, the result must be the same (port is optional)

http://<Public-DNS-Name>:80

Graphical user interface, text, application

Description automatically generatedYou can leave this file in place as a temporary landing page for your application until you set up an index.php file to replace it. Once you do that, remember to remove or rename the index.html file from your document root, as it would take precedence over an index.php file by default.

**Step -5- Enable PHP On The Website**

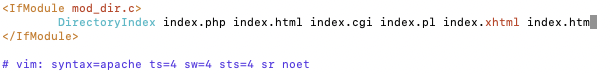
With the default **DirectoryIndex** settings on Apache, a file named index.html will always take precedence over an index.php file. This is useful for setting up maintenance pages in PHP applications, by creating a temporary index.html file containing an informative message to visitors. Because this page will take precedence over the index.php page, it will then become the landing page for the application. Once maintenance is over, the index.html is renamed or removed from the document root, bringing back the regular application page.

In case you want to change this behavior, you’ll need to edit the **/etc/apache2/mods-enabled/dir.conf** file and change the order in which the **index.php** file is listed within the **DirectoryIndex** directive:

sudo vim /etc/apache2/mods-enabled/dir.conf

Text

Description automatically generated with medium confidence

Changed to:

After saving and closing the file, you will need to reload Apache so the changes take effect:

sudo systemctl reload apache2

Finally, we will create a PHP script to test that PHP is correctly installed and configured on your server.

Now that you have a custom location to host your website’s files and folders, we’ll create a PHP test script to confirm that Apache is able to handle and process requests for PHP files.

Create a new file named index.php inside your custom web root folder:

vim /var/www/projectlamp/index.php

This will open a blank file. Add the following text, which is valid PHP code, inside the file:

<?php

phpinfo();

When you are finished, save and close the file, refresh the page and you will see a page similar to this:

Graphical user interface, application

Description automatically generated

This page provides information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If you can see this page in your browser, then your PHP installation is working as expected.

After checking the relevant information about your PHP server through that page, it’s best to remove the file you created as it contains sensitive information about your PHP environment -and your Ubuntu server. You can use rm to do so:

sudo rm /var/www/projectlamp/index.php

You can always recreate this page if you need to access the information again later.

Credit: [This guide was inspired by Digital Ocean](https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04#step-3-%E2%80%94-installing-php)

Congratulations! You have finished your very first **REAL LIFE PROJECT** by deploying a LAMP stack website in AWS Cloud!