

## PROJECT1

- LAMP STACK IMPLEMENTATION

- **Step 1 — Installing Apache and Updating the Firewall**

- To get the instance ID(EC2) connected to the ubuntu OS terminal(Install Apache using Ubuntu's package manager)
- `ssh -i "Hill_EC2.pem" ubuntu@ec2-13-48-28-237.eu-north-1.compute.amazonaws.com.`  
To update the firewall on the Apache web server

```
ubuntu@ip-172-31-11-37:~$ sudo apt update
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:5 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1031 kB]
Get:6 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [902 kB]
Fetched 2270 kB in 1s (2427 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
32 packages can be upgraded. Run 'apt list --upgradable' to see them.
```



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 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 **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 means 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 split 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 **manual** if the `apache2-doc` package was installed on this server.

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

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` 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.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

### Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public\_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

### Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

## Step 2 — Installing MySQL

I need to install a [Database Management System \(DBMS\)](#) to be able to store and manage data for your site in a [relational database](#). [MySQL](#).

```
ubuntu@ip-172-31-11-37:~$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mysql-server is already the newest version (8.0.32-0ubuntu0.22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
ubuntu@ip-172-31-11-37:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.32-0ubuntu0.22.04.2 (Ubuntu)
```

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

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.
```

**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: always say no as it will still prompt you to add password before you login into my sql server.

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

```
$ sudo mysql -p
```

Notice the **-p** flag in this command, which will prompt you for the password used after changing the **root** user password. The password i used was We!come@\$140

To exit the MySQL console, type:

```
mysql> exit
```

Notice that you need to provide a password to connect as the **root** user.

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](#) 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:

```
ubuntu@ip-172-31-42-227:~$ sudo apt install php libapache2-mod-php php-mysql
```

```
Creating config file /etc/php/7.4/mods-available/json.ini with new version
```

```
Setting up php7.4-readline (7.4.3-4ubuntu2.18) ...
```

```
Creating config file /etc/php/7.4/mods-available/readline.ini with new version
```

```
Setting up php7.4-opcache (7.4.3-4ubuntu2.18) ...
```

```
Creating config file /etc/php/7.4/mods-available/opcache.ini with new version
```

```
Setting up php7.4-cli (7.4.3-4ubuntu2.18) ...
```

```
update-alternatives: using /usr/bin/php7.4 to provide /usr/bin/php (php) in auto mode
```

```
update-alternatives: using /usr/bin/phar7.4 to provide /usr/bin/phar (phar) in auto mode
```

```
update-alternatives: using /usr/bin/phar.phar7.4 to provide /usr/bin/phar.phar (phar.phar) in auto mode
```

Creating config file /etc/php/7.4/cli/php.ini with new version  
Setting up libapache2-mod-php7.4 (7.4.3-4ubuntu2.18) ...

Creating config file /etc/php/7.4/apache2/php.ini with new version  
Module mpm\_event disabled.  
Enabling module mpm\_prefork.  
apache2\_switch\_mpm Switch to prefork  
apache2\_invoke: Enable module php7.4  
Setting up php7.4 (7.4.3-4ubuntu2.18) ...  
Setting up libapache2-mod-php (2:7.4+75) ...  
Setting up php (2:7.4+75) ...  
Processing triggers for man-db (2.9.1-1) ...  
Processing triggers for php7.4-cli (7.4.3-4ubuntu2.18) ...  
Processing triggers for libapache2-mod-php7.4 (7.4.3-4ubuntu2.18) ...  
ubuntu@ip-172-31-42-227:~\$ php -v  
PHP 7.4.3-4ubuntu2.18 (cli) (built: Feb 23 2023 12:43:23) ( NTS )  
Copyright (c) The PHP Group  
Zend Engine v3.4.0, Copyright (c) Zend Technologies  
with Zend OPcache v7.4.3-4ubuntu2.18, Copyright (c), by Zend Technologies

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

- Linux (Ubuntu)
- Apache HTTP Server
- MySQL
- PHP

## 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 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>
```

```
ubuntu@ip-172-31-42-227:~$ sudo mkdir /var/www/projectlamp
ubuntu@ip-172-31-42-227:~$ sudo chown -R $USER:$USER /var/www/projectlamp
ubuntu@ip-172-31-42-227:~$ sudo vi /etc/apache2/sites-available/projectlamp.conf
ubuntu@ip-172-31-42-227:~$
```

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.
ubuntu@ip-172-31-42-227:~$ sudo ls /etc/apache2/sites-available
000-default.conf default-ssl.conf projectlamp.conf
```

```
ubuntu@ip-172-31-42-227:~$
```

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
```

```
ubuntu@ip-172-31-42-227:~$ sudo a2ensite projectlamp
```

Enabling site projectlamp.

To activate the new configuration, you need to run:

```
systemctl reload apache2
```

```
ubuntu@ip-172-31-42-227:~$
```

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
```

```
ubuntu@ip-172-31-42-227:~$ sudo a2dissite 000-default
```

Site 000-default disabled.

To activate the new configuration, you need to run:

```
systemctl reload apache2
```

```
ubuntu@ip-172-31-42-227:~$
```

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

```
sudo apache2ctl configtest
```

```
ubuntu@ip-172-31-42-227:~$ sudo apache2ctl configtest
```

Syntax OK

Finally, reload Apache so these changes take effect:

```
sudo systemctl reload apache2
```


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:

```
ubuntu@ip-172-31-42-227:~$ sudo echo 'Hello LAMP from hostname' $(curl -s  
http://13.48.27.107/latest/meta-data/public-hostname) 'with public IP' $(curl -s  
http://13.48.27.107/latest/meta-data/public-ipv4) > /var/www/projectlamp/index.html
```



ubuntu@ip-172-31-42-227:~\$:

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



## Apache2 Ubuntu Default Page

# ubuntu

**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 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 **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 means 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 split 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 **manual** if the `apache2-doc` package was installed on this server.

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

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

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

You 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
```

Then to clear every thing i(insert), esc, :(shift:), %d then insert again

```
<IfModule mod_dir.c>
```

```
    #Change this:
```

```
    #DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
```

```
    #To this:
```

```
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
```

```
</IfModule>
```

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:

```
Sudo vim /var/www/projectlamp/index.php
```

Insert, esc, :%d which is just to clear things or just insert :q

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

```
<?php
```

```
phpinfo();
```

```
:wq(save apache2 file)
```

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



System	Linux ip-172-31-11-37 5.15.0-1031-aws #35-Ubuntu SMP Fri Feb 10 02:07:18 UTC 2023 x86_64
Build Date	Feb 22 2023 22:56:18
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqld.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-phar.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysmsg.ini, /etc/php/8.1/apache2/conf.d/20-sysvsem.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled

## Configuration

### apache2handler

Apache Version	Apache/2.4.52 (Ubuntu)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	projectlamp:0
User/Group	www-data(33)/33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_auth_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php mod_reqtimeout mod_setenvif mod_status

Directive	Local Value	Master Value
engine	On	On
last_modified	Off	Off
xbithack	Off	Off

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
```

Then reload apache

```
sudo systemctl reload apache2
```