# WEB STACK IMPLEMENTATION (LAMP STACK) IN AWS

A technology stack is a set of frameworks and tools used to develop a software product. This set of frameworks and tools are very specifically chosen to work together in creating a well-functioning software. They are acronymns for individual technologies used together for a specific technology product. some examples ara:

- **LAMP** (Linux, Apache, MySQL, PHP or Python, or Perl)
- **LEMP** (Linux, Nginx, MySQL, PHP or Python, or Perl)
- **MERN** (MongoDB, ExpressJS, ReactJS, NodeJS)
- **MEAN** (MongoDB, ExpressJS, AngularJS, NodeJS

## Lamp Stack

This is a software bundle which is composed of Linux operating system, the Apache HTTP server, the MySQLrelational database management system(RDBMS) and the PHP(Peri, Python) programming language. It was one of the first open source software stacks for the web and remains one of the most common ways to deliver web applications.  They serve dynamic web content through CGI scripting and are suitable for building dynamic websites and web applications.
It is stable, simple and powerful.

### The Components of Lamp Stack

LAMP stands for Linux, Apache, MySQL, and PHP. Together, they provide a proven set of software for delivering high-performance web applications. Each component contributes essential capabilities to the stack:

- Linux: The operating system. Linux is a free and open source operating system (OS) that has been around since the mid-1990s. Today, it has an extensive worldwide user base that extends across industries. Linux is popular in part because it offers more flexibility and configuration options than some other operating systems.
- Apache: The web server. The Apache web server processes requests and serves up web assets via HTTP so that the application is accessible to anyone in the public domain over a simple web URL. Developed and maintained by an open community, Apache is a mature, feature-rich server that runs a large share of the websites currently on the internet.
- MySQL: The database. MySQL is an open source relational database management system for storing application data. With My SQL, you can store all your information in a format that is easily queried with the SQL language. SQL is a great choice if you are dealing with a business domain that is well structured, and you want to translate that structure into the backend. MySQL is suitable for running even large and complex sites.
- PHP: The programming language. The PHP open source scripting language works with Apache to help you create dynamic web pages. You cannot use HTML to perform dynamic processes such as pulling data out of a database. To provide this type of functionality, you simply drop PHP code into the parts of a page that you want to be dynamic.

PHP is designed for efficiency. It makes programming easier—and a bit more fun—by allowing you to write new code, hit refresh, and immediately see the resulting changes without the need for compiling. If you prefer, you can swap out PHP in favor of Perl or the increasingly popular Python language.

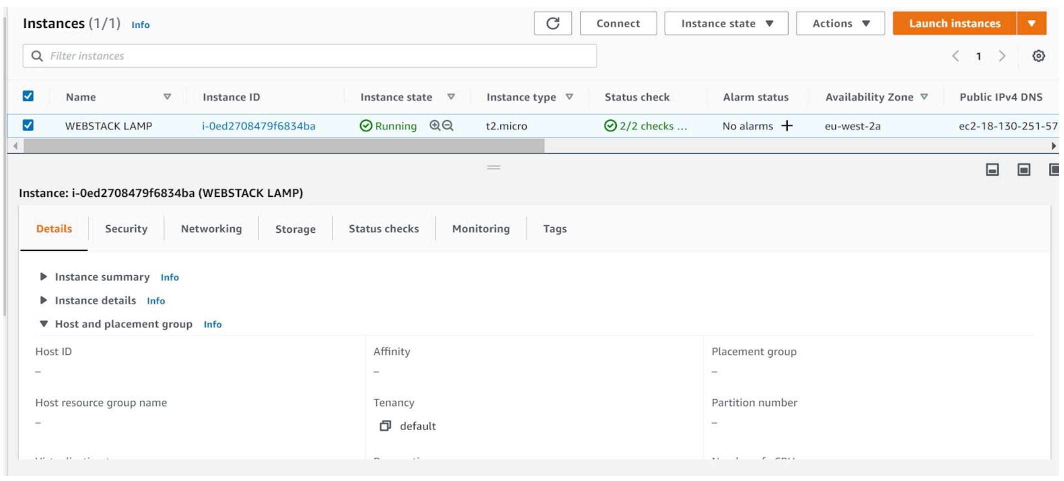
The objective of this project is to deploy the LAMP stack website in AWS cloud.

**STEP 1 - Setting up the Amazon EC2 Ubuntu server.**
There are several types of Linux distribution such as Redhat, Cent0S, Fedora, Ubuntu etc. For the purpose of this project, I will be using Ubuntu on a AWS EC2 server. The **package manager, apt (advanced package tool)** is also used in installation.

Just like Windows, iOS, and Mac OS, Linux is an operating system. Android is powered by the Linux operating system. An operating system is software that manages all of the hardware resources associated with your desktop or laptop. To put it simply, the operating system manages the communication between your software and your hardware. Without the operating system (OS), the software would not function.

TCP: The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layer of the TCP/IP suite. SSL/TLS often runs on top of TCP.
TCP is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established.

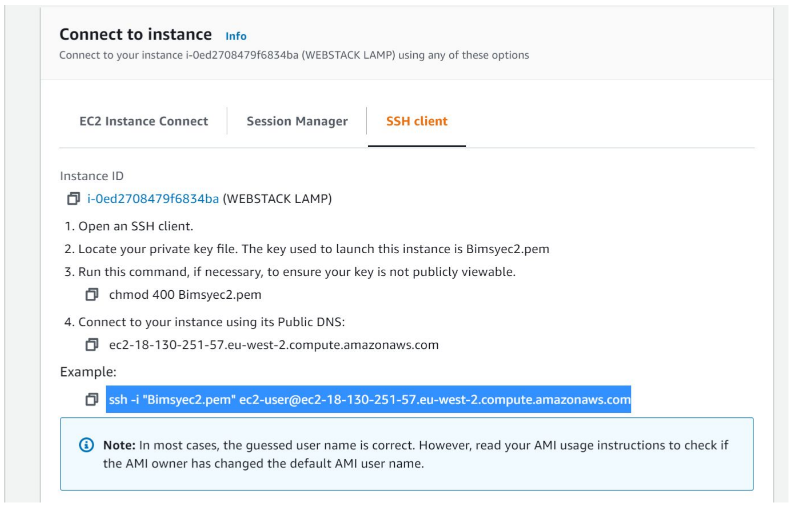
Basically, an Ubuntu instance was launched in AWS. In the process, the security, storage, networking etc were defined. The instance was also connected to the terminal through the 'connect' tab in AWS and then copying the connecting link as in the image below.



| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS |
|---|---|---|---|---|---|---|---|
| WEBSTACK LAMP | i-0ed2708479f6834ba | Running | t2.micro | 2/2 checks ... | No alarms | eu-west-2a | ec2-18-130-251-57 |

Instance: i-0ed2708479f6834ba (WEBSTACK LAMP)

| Host ID | Affinity | Placement group |
|---|---|---|
| – | – | – |
| **Host resource group name** | **Tenancy** | **Partition number** |
| – | default | – |


**Connect to instance** Info
Connect to your instance i-0ed2708479f6834ba (WEBSTACK LAMP) using any of these options

Instance ID
i-0ed2708479f6834ba (WEBSTACK LAMP)

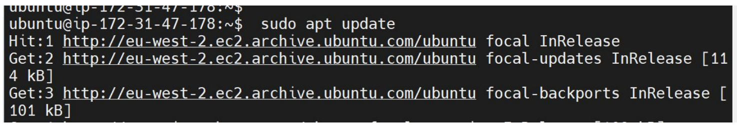
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Bimsyec2.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
```
chmod 400 Bimsyec2.pem
```
4. Connect to your instance using its Public DNS:
```
ec2-18-130-251-57.eu-west-2.compute.amazonaws.com
```
Example:
```
ssh -i "Bimsyec2.pem" ec2-user@ec2-18-130-251-57.eu-west-2.compute.amazonaws.com
```
**Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

We will now proceed with the LAMP stack implementation with the steps highlighted below:

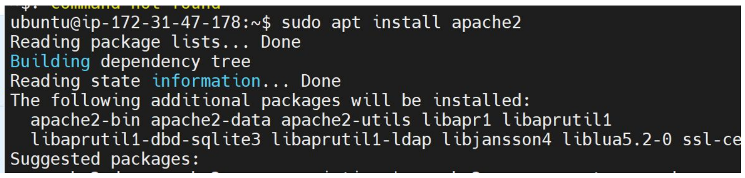
**STEP 2 — Installing Apache and Updating the Firewall**
Apache HTTP Server is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and 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, Microsoft's IIS, 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':

#update a list of packages in package manager
$ sudo apt update
#run apache2 package installation
$ sudo apt install apache2



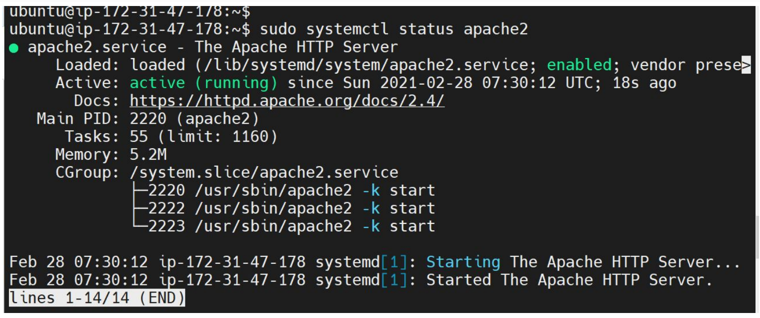
```
ubuntu@ip-172-31-47-178:~$  sudo apt update
Hit:1 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [11
4 kB]
Get:3 http://eu-west-2.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [
101 kB]
```



```
ubuntu@ip-172-31-47-178:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libjansson4 liblua5.2-0 ssl-ce
Suggested packages:
```

**Verify that apache2 is running as a Service in the OS:**

Use this command: $ sudo systemctl status apache2



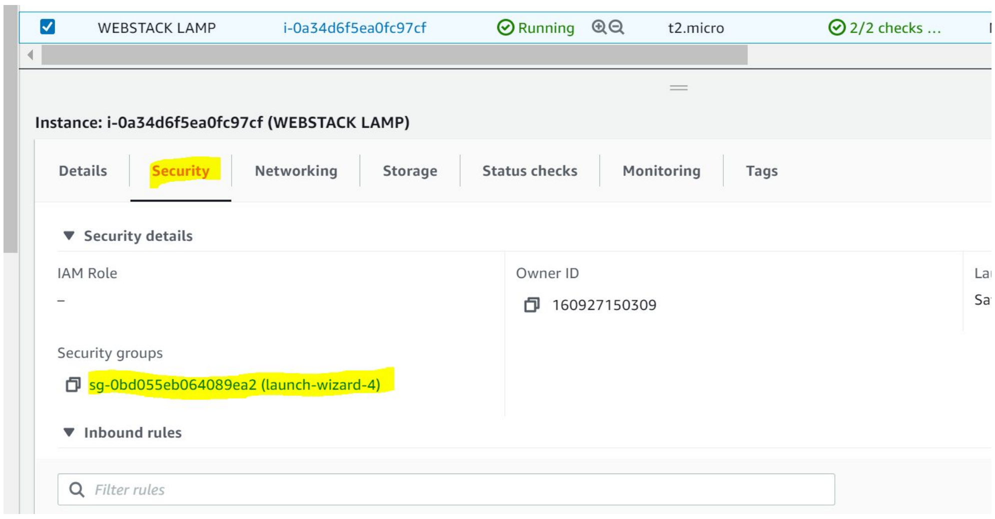
```
ubuntu@ip-172-31-47-178:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese>
     Active: active (running) since Sun 2021-02-28 07:30:12 UTC; 18s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2220 (apache2)
      Tasks: 55 (limit: 1160)
     Memory: 5.2M
     CGroup: /system.slice/apache2.service
             ├─2220 /usr/sbin/apache2 -k start
             ├─2222 /usr/sbin/apache2 -k start
             └─2223 /usr/sbin/apache2 -k start

Feb 28 07:30:12 ip-172-31-47-178 systemd[1]: Starting The Apache HTTP Server...
Feb 28 07:30:12 ip-172-31-47-178 systemd[1]: Started The Apache HTTP Server.
lines 1-14/14 (END)
```

The output above shows that Apache is running ok. This confirms the success of Web Server in the clouds.

**Ensure that the Web server can receive Traffic :**

- Open TCP port 80 which is the default port that web browsers use to access web pages on the Internet. This allows connection through port 80 (http)
- Enable the ufw (uncomplicated firewall)
- And, allow Apache traffic through the firewall.



WEBSTACK LAMP | i-0a34d6f5ea0fc97cf | Running | t2.micro | 2/2 checks ...

Instance: i-0a34d6f5ea0fc97cf (WEBSTACK LAMP)

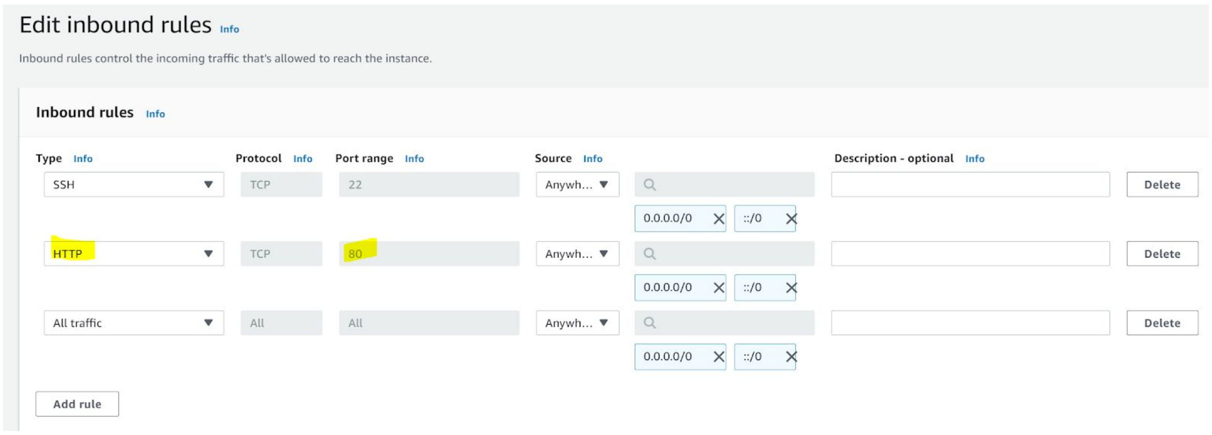
Security details

IAM Role
–

Owner ID
160927150309

Security groups
sg-0bd055eb064089ea2 (launch-wizard-4)

Inbound rules



Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

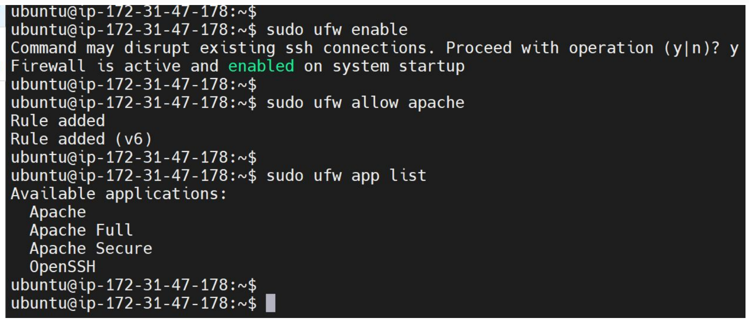
| Type | Protocol | Port range | Source | | Description - optional | |
|---|---|---|---|---|---|---|
| SSH | TCP | 22 | Anywh... | 0.0.0.0/0 ::/0 | | Delete |
| HTTP | TCP | 80 | Anywh... | 0.0.0.0/0 ::/0 | | Delete |
| All traffic | All | All | Anywh... | 0.0.0.0/0 ::/0 | | Delete |

The output above shows that we can access the server locally and from the Internet (Source 0.0.0.0/0 means 'from any IP address').

Enable the firewall on the server: $ sudo ufw enable

Allow Apache traffic: $ sudo ufw allow apache

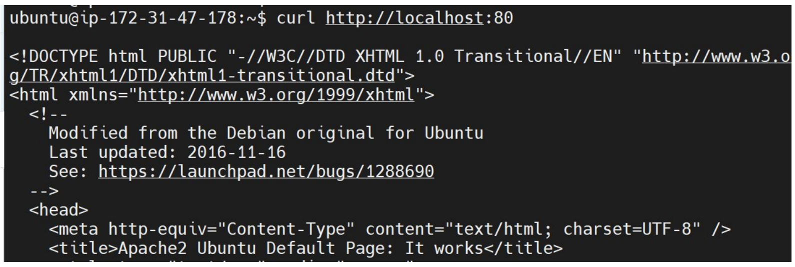
Check all the firewall set up: $ sudo ufw app list



```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo ufw allow apache
Rule added
Rule added (v6)
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ 
```

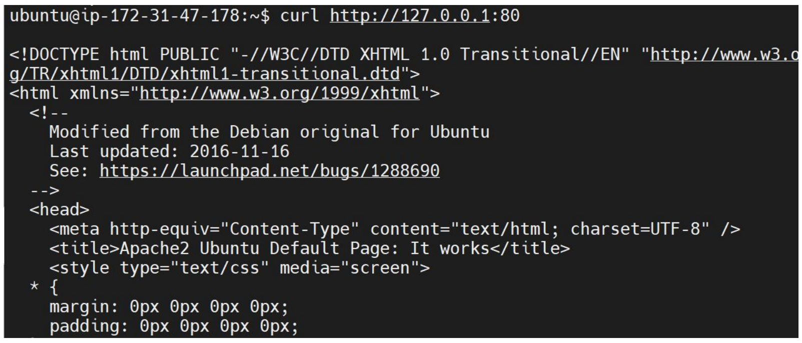
Check if we can access it locally via Ubuntu shell:

$ curl http://localhost:80
or
$ curl http://127.0.0.1:80



```
ubuntu@ip-172-31-47-178:~$ curl http://localhost:80

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.o
g/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>
```



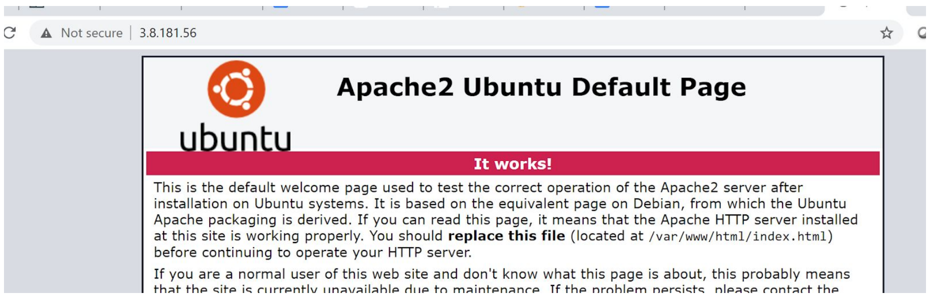
```
ubuntu@ip-172-31-47-178:~$ curl http://127.0.0.1:80

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.o
g/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;
```

These 2 commands above actually do pretty much the same - they use 'curl' command to request our Apache HTTP Server on port 80 (actually you can even try to not specify any port - it will work anyway). The difference is that: in the first case we try to access our server via DNS name and in the second one - by IP address (in this case IP address 127.0.0.1 corresponds to DNS name 'localhost' and the process of converting a DNS name to IP address is called "resolution"). We will touch DNS in further lectures and projects.
As an output you can see some strangely formatted test, do not worry, we just made sure that our Apache web service responds to 'curl' command with some payload.

**Test Apache HTTP:** In order to see if the server can respond to requests from the Internet. Open a web browser of your choice and try to access following url

http://<Public-IP-Address>:80
http://3.8.181.56:80



Not secure | 3.8.181.56

Apache2 Ubuntu Default Page

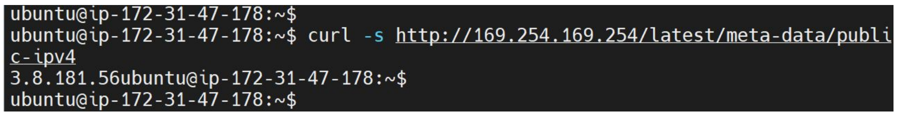
It works!

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

The above output is the apache ubuntu default page and it shows that the web server is now correctly installed and accessible through the firewall.

Another way to retrieve an 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


```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ curl -s http://169.254.169.254/latest/meta-data/publi
c-ipv4
3.8.181.56ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
```

The URL in the browser shall also work if you do not specify port number since all web browsers use port 80 by default.

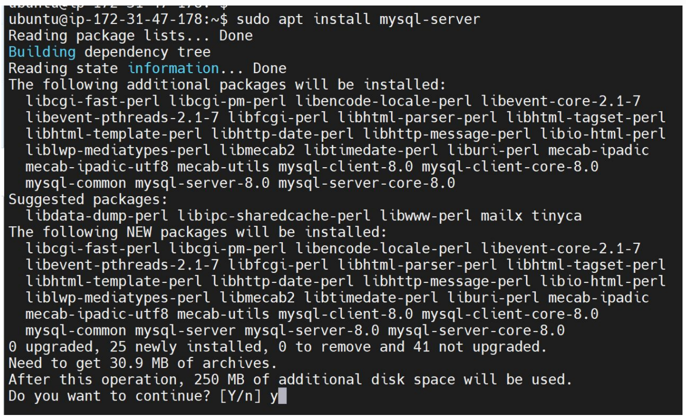
In fact, it is the same content that you previously got by 'curl' command, but represented in nice HTML formatting by your web browser.

**STEP 3 - Installing MySQL**

As defined above, The database. MySQL is a popular open source relational database management system used within the PHP environment for storing application data. The Database Management System (DBMS) is used to store and manage data for your site in a relational database.

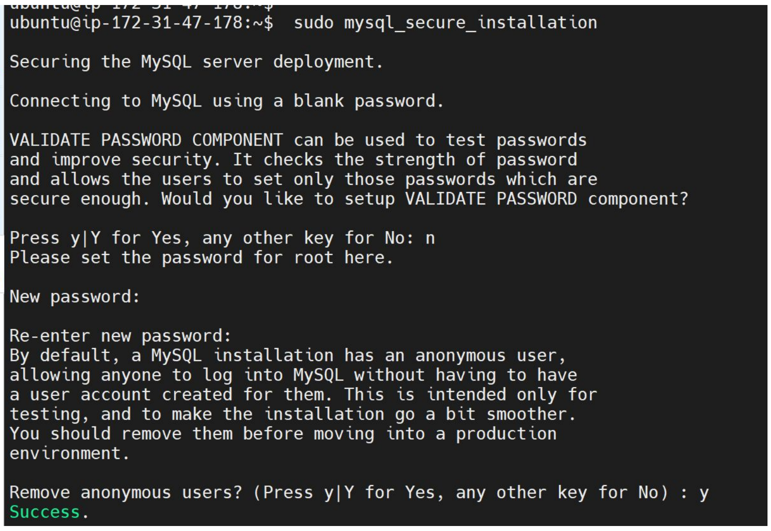
Acquire and install the Mysql software:
$ sudo apt install mysql-server

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


```
ubuntu@ip-172-31-47-178:~$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0
  mysql-common mysql-server-8.0 mysql-server-core-8.0
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca
The following NEW packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0
  mysql-common mysql-server mysql-server-8.0 mysql-server-core-8.0
0 upgraded, 25 newly installed, 0 to remove and 41 not upgraded.
Need to get 30.9 MB of archives.
After this operation, 250 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

When the installation is finished, it's recommended to 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. Start the interactive script by running:

$ sudo mysql_secure_installation.



```
ubuntu@ip-172-31-47-178:~$  sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT 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 component?

Press y|Y for Yes, any other key for No: n
Please set the password for root here.

New password:

Re-enter new password:
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.
```

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.

**If you answer "yes" to the validation question**, 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.

```
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:

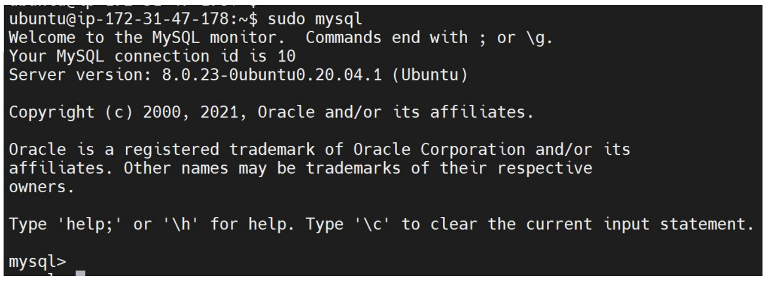
```
Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
```

For the rest of the questions, press `Y` and hit the `ENTER` key at each prompt. This will 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.

**Test MySQL:**

Log in to the MySQL console by typing:

$ sudo mysql



```
ubuntu@ip-172-31-47-178:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.23-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, 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>
```

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. You should see output like this:

To exit the MySQL console, type: mysql> exit

Notice that you didn't need to provide a password to connect as the **root** user, even though you have defined one when running the `mysql_secure_installation` script. That is because the default authentication method for the administrative MySQL user is `unix_socket` instead of `password.` Even though this might look like a security concern at first, it makes the database server more secure because the only users allowed to log in as the **root** MySQL user are the system users with sudo privileges connecting from the console or through an application running with the same privileges. In practical terms, that means you won't be able to use the administrative database **root** user to connect from your PHP application. Setting a password for the **root** MySQL account works as a safeguard, in case the default authentication method is changed from `unix_socket` to `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**: 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

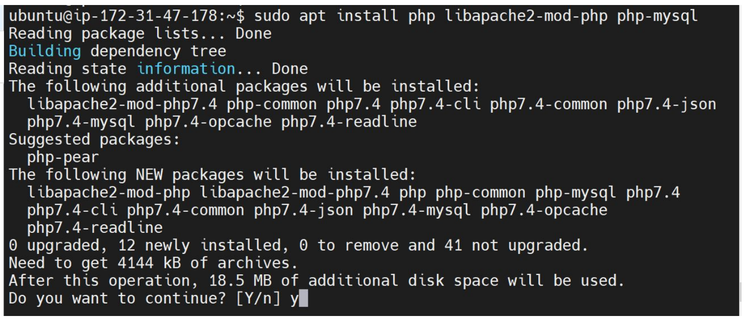
PHP applications on MySQL 8, you'll need to make sure they're configured to use `mysql_native_password` instead.

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

**STEP 4 - Installing PHP**

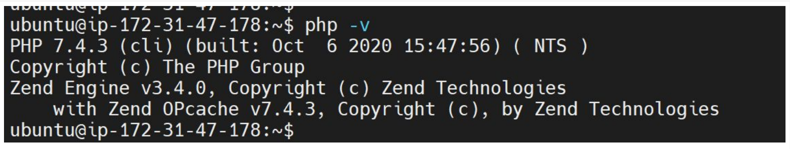
Apache has been installed to serve the web content and MySQL installed to store and manage the data. PHP is the component of our setup that will process code to display dynamic content to the end user. In addition to the `php` package, you'll need `php-mysql`, a PHP module that allows PHP to communicate 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:

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



```
ubuntu@ip-172-31-47-178:~$ sudo apt install php libapache2-mod-php php-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.4 php-common php7.4 php7.4-cli php7.4-common php7.4-json
  php7.4-mysql php7.4-opcache php7.4-readline
Suggested packages:
  php-pear
The following NEW packages will be installed:
  libapache2-mod-php libapache2-mod-php7.4 php php-common php-mysql php7.4
  php7.4-cli php7.4-common php7.4-json php7.4-mysql php7.4-opcache
  php7.4-readline
0 upgraded, 12 newly installed, 0 to remove and 41 not upgraded.
Need to get 4144 kB of archives.
After this operation, 18.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

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



```
ubuntu@ip-172-31-47-178:~$ php -v
PHP 7.4.3 (cli) (built: Oct  6 2020 15:47:56) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
ubuntu@ip-172-31-47-178:~$
```

LAMP stack is completely installed and fully operational.

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

To test the LAMP setup with a PHP script, it's best to set up a proper Apache Virtual Host to hold the 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.

**STEP 5: Creating and configuring a Virtual Host** for the website using Apache.

- Set up a domain called `projectlamp`

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 it is and will add our own directory next to the default one.

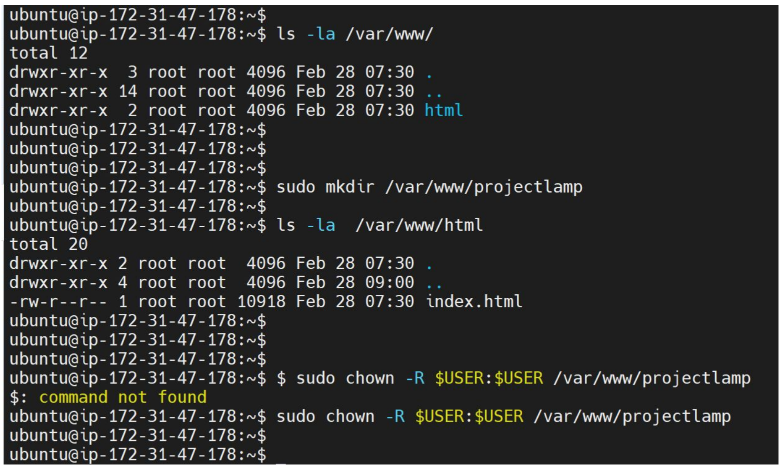
- View the /var/www/ directory: ls -la /var/www/
- View the /var/www/ directory: ls -la /var/www/html

- Create the directory for projectlamp: $ sudo mkdir /var/www/projectlamp

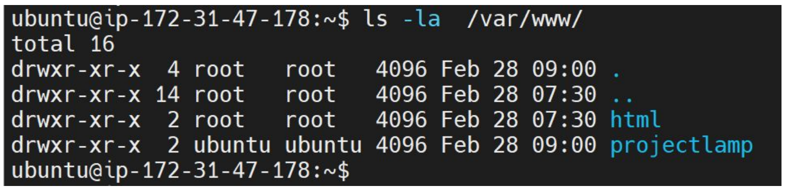
- Assign ownership of the directory with the $USER environment variable, which will reference the current system user:

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

Sample Output



```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ ls -la /var/www/
total 12
drwxr-xr-x  3 root root 4096 Feb 28 07:30 .
drwxr-xr-x 14 root root 4096 Feb 28 07:30 ..
drwxr-xr-x  2 root root 4096 Feb 28 07:30 html
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo mkdir /var/www/projectlamp
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ ls -la  /var/www/html
total 20
drwxr-xr-x 2 root root  4096 Feb 28 07:30 .
drwxr-xr-x 4 root root  4096 Feb 28 09:00 ..
-rw-r--r-- 1 root root 10918 Feb 28 07:30 index.html
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ $ sudo chown -R $USER:$USER /var/www/projectlamp
$: command not found
ubuntu@ip-172-31-47-178:~$ sudo chown -R $USER:$USER /var/www/projectlamp
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ _
```



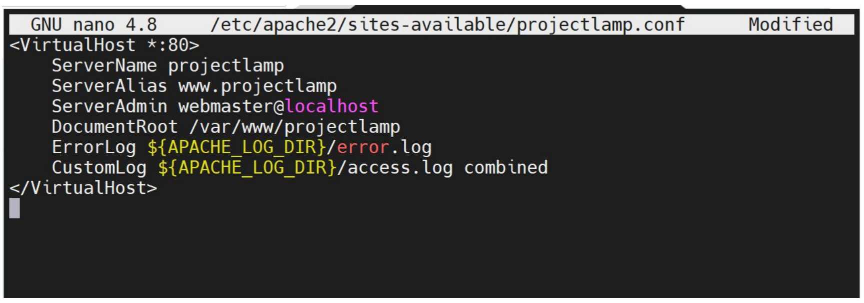
```
ubuntu@ip-172-31-47-178:~$ ls -la  /var/www/
total 16
drwxr-xr-x  4 root   root   4096 Feb 28 09:00 .
drwxr-xr-x 14 root   root   4096 Feb 28 07:30 ..
drwxr-xr-x  2 root   root   4096 Feb 28 07:30 html
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb 28 09:00 projectlamp
ubuntu@ip-172-31-47-178:~$
```

Then, create and open a new configuration file in Apache's `sites-available` directory using the preferred command-line editor. Here, we'll be using

$ 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>
```



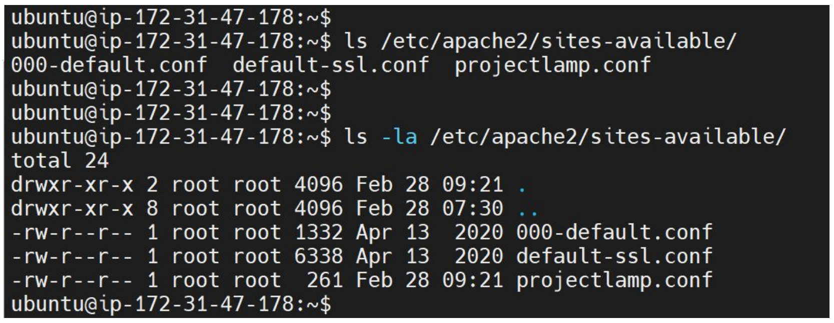
```
  GNU nano 4.8      /etc/apache2/sites-available/projectlamp.conf       Modified
<VirtualHost *:80>
    ServerName projectlamp
    ServerAlias www.projectlamp
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/projectlamp
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log 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

View the new the new file created

$ sudo ls /etc/apache2/sites-available



```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ ls /etc/apache2/sites-available/
000-default.conf  default-ssl.conf  projectlamp.conf
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ ls -la /etc/apache2/sites-available/
total 24
drwxr-xr-x 2 root root 4096 Feb 28 09:21 .
drwxr-xr-x 8 root root 4096 Feb 28 07:30 ..
-rw-r--r-- 1 root root 1332 Apr 13  2020 000-default.conf
-rw-r--r-- 1 root root 6338 Apr 13  2020 default-ssl.conf
-rw-r--r-- 1 root root  261 Feb 28 09:21 projectlamp.conf
ubuntu@ip-172-31-47-178:~$
```

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.

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:

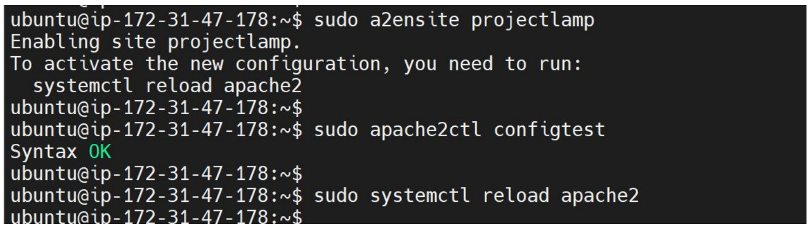
$ sudo a2dissite 000-default

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

$ sudo apache2ctl configtest

Finally, reload Apache so these changes take effect:

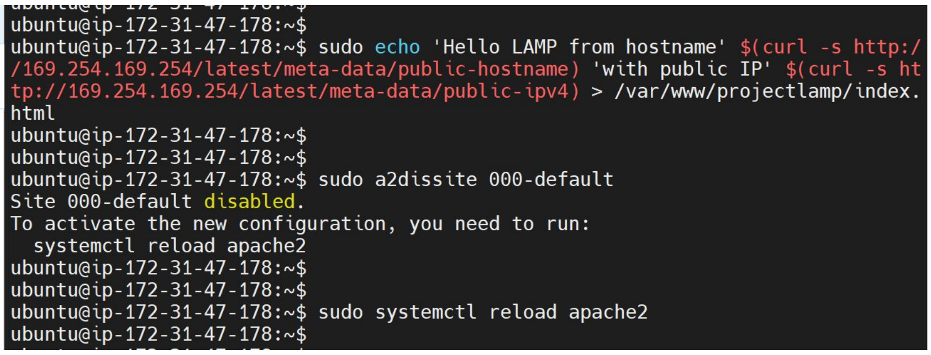
$ sudo systemctl reload apache2



```
ubuntu@ip-172-31-47-178:~$ sudo a2ensite projectlamp
Enabling site projectlamp.
To activate the new configuration, you need to run:
  systemctl reload apache2
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo apache2ctl configtest
Syntax OK
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo systemctl reload apache2
ubuntu@ip-172-31-47-178:~$
```

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
```

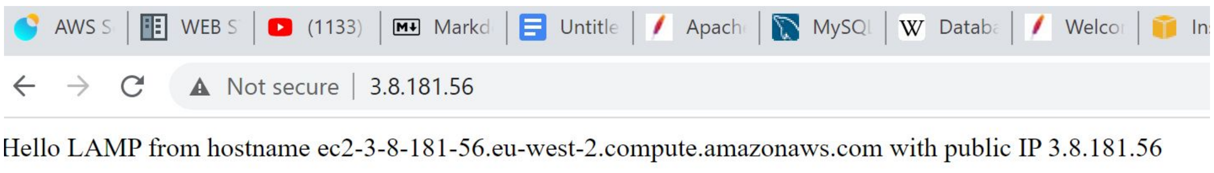


```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo echo 'Hello LAMP from hostname' $(curl -s http:/
/169.254.169.254/latest/meta-data/public-hostname) 'with public IP' $(curl -s ht
tp://169.254.169.254/latest/meta-data/public-ipv4) > /var/www/projectlamp/index.
html
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo a2dissite 000-default
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo systemctl reload apache2
ubuntu@ip-172-31-47-178:~$
```

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

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

`http://3.8.181.56:80`

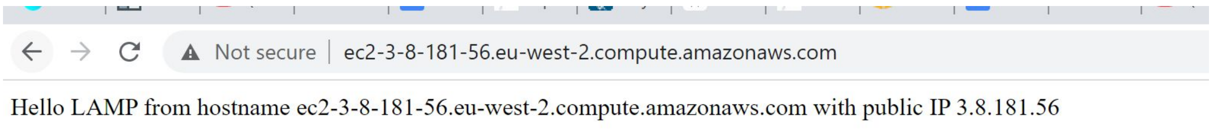


Hello LAMP from hostname ec2-3-8-181-56.eu-west-2.compute.amazonaws.com with public IP 3.8.181.56

If you see the text from ***'echo'*** command you wrote to the 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`

`http://ec2-3-8-181-56.eu-west-2.compute.amazonaws.com:80`


Not secure | ec2-3-8-181-56.eu-west-2.compute.amazonaws.com

Hello LAMP from hostname ec2-3-8-181-56.eu-west-2.compute.amazonaws.com with public IP 3.8.181.56

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.

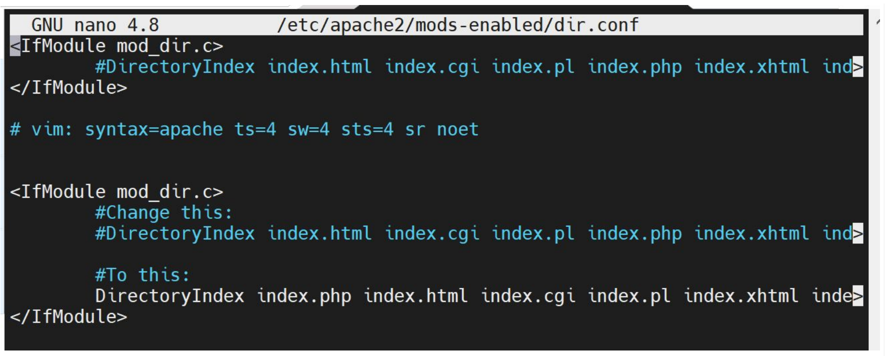
**STEP 5: Enable PHP on the website**

With the default Dir**ectoryIndex** 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

```
<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>
```



```
  GNU nano 4.8            /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
        #DirectoryIndex index.html index.cgi index.pl index.php index.xhtml ind>
</IfModule>

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


<IfModule mod_dir.c>
        #Change this:
        #DirectoryIndex index.html index.cgi index.pl index.php index.xhtml ind>

        #To this:
        DirectoryIndex index.php index.html index.cgi index.pl index.xhtml inde>
</IfModule>
```

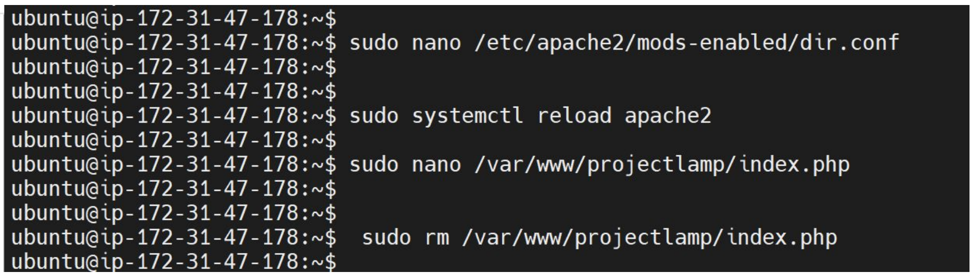
After saving and closing the file, you will need to reload Apache so the changes take effect:

```
$ sudo systemctl reload apache2
```

Finally, create a PHP script to test that PHP is correctly installed and configured on the server. THis is will also confirm that Apache is able to handle and process requests for PHP files using the custom location.

- Create a new file named `index.php` inside the custom web root folder(/var/www/projectlamp) :

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

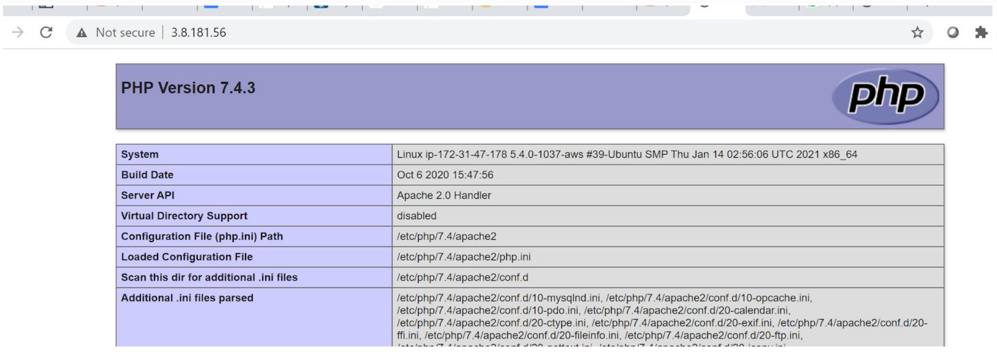


```
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo nano /etc/apache2/mods-enabled/dir.conf
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo systemctl reload apache2
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$ sudo nano /var/www/projectlamp/index.php
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$
ubuntu@ip-172-31-47-178:~$  sudo rm /var/www/projectlamp/index.php
ubuntu@ip-172-31-47-178:~$
```

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

```
<?php
phpinfo();
```

Save and close the file.
Refresh the page and the output below should come up.



Not secure | 3.8.181.56

PHP Version 7.4.3

| | |
|---|---|
| System | Linux ip-172-31-47-178 5.4.0-1037-aws #39-Ubuntu SMP Thu Jan 14 02:56:06 UTC 2021 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, |

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

Credits

Darey.io

Apache: https://httpd.apache.org/

DNS: https://en.wikipedia.org/wiki/Domain_Name_System

MySQL: https://www.mysql.com/

Relational Database: https://en.wikipedia.org/wiki/Relational_database

DBMS: https://en.wikipedia.org/wiki/Database#Database_management_system

Apache Virtual Host: https://httpd.apache.org/docs/2.4/vhosts/