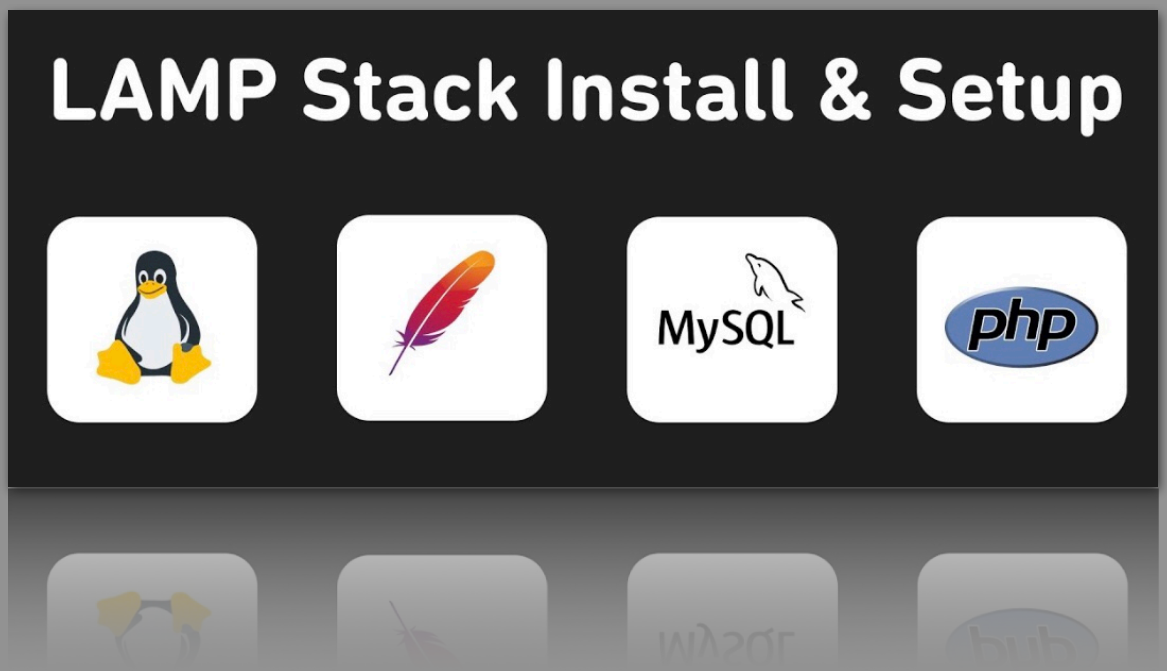
LAMP STACK UBUNTU



A “LAMP” stack is a group of open-source software that is typically installed together in order to enable a server to host dynamic websites and web apps written in PHP. This term is an acronym which represents the Linux operating system, with the Apache web server. The site data is stored in a MySQL database, and dynamic content is processed by PHP.

Step 1 — Installing Apache and Updating the Firewall

```
$ sudo apt update
$ sudo apt install apache2

$ sudo ufw app list

---------------------------------------------------
#Output
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
---------------------------------------------------
```

```
sudo ufw allow 'Apache'
sudo ufw status

-------------------------------------------------

#Output
Status: active

To                         Action      From
 --                         ------          ----
OpenSSH                    ALLOW       Anywhere
Apache                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere
(v6)
Apache (v6)                ALLOW       Anywhere
(v6)
-------------------------------------------------

sudo systemctl status apache2
```

Step 2 — Installing MySQL

$ sudo apt install mysql-server

This script will remove some insecure default settings and lock down access to your database system.

$ sudo mysql_secure_installation

Select Y for yes and 1 for medium password

# Log in to the MySQL console type:
$ sudo mysql

To exit the MySQL console, type:
mysqul> $ exit

Step 3 — Installing PHP

```
$ sudo apt install php libapache2-mod-php
php-mysql
# Version check
$ php -v
```

Step 4 — Creating a Virtual Host

```
sudo mkdir /var/www/stevenagri.com
sudo chown -R $USER:$USER /var/www/
stevenagri.com
sudo chmod -R 755 /var/www/stevenagri.com
sudo nano /var/www/stevenagri.com/
index.html
sudo nano /etc/apache2/sites-available/
stevenagri.com.conf
```

#Output

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName stevenagri.com
    ServerAlias www.stevenagri.com
    DocumentRoot /var/www/stevenagri.com
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log
combined
</VirtualHost>
```

```
sudo a2ensite stevenagri.com.conf
sudo a2dissite 000-default.conf
sudo apache2ctl configtest
sudo systemctl restart apache2
```

Step 5 — Testing PHP Processing

Create a new file named info.php inside your custom web root folder:
$ sudo nano /var/www/your_domain/info.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.

To test this script, go to your web browser and access your server's domain name or IP address, followed by the script name, which in this case is info.php:
http://server_domain_or_IP/info.php

Step 6 How To Secure Apache with Let's Encrypt

Let’s Encrypt is a Certificate Authority (CA) that facilitates obtaining and installing free TLS/SSL certificates, thereby enabling encrypted HTTPS on web servers

We will use Certbot to obtain a free SSL certificate for Apache on Ubuntu 20.04, and make sure this certificate is set up to renew automatically

Step 1 — Installing Certbot

$ sudo apt install certbot python3-certbot-apache

Step 2 — Checking your Apache Virtual Host Configuration

```
$ sudo nano /etc/apache2/sites-available/your_domain.conf
...
ServerName your_domain
ServerAlias www.your_domain
...

# If you already have your ServerName and ServerAlias set up like this, you can exit your text editor.

$ sudo apache2ctl configtest

#You should get a Syntax OK as a response.

$ sudo systemctl reload apache2
```

Step 3 — Allowing HTTPS Through the Firewall

To additionally let in HTTPS traffic, allow the “Apache Full” profile and delete the redundant “Apache” profile:

$ sudo ufw app list

$ sudo ufw allow 'Apache Full'

$ sudo ufw delete allow 'Apache'

Step 4 — Obtaining an SSL Certificate

```
$ sudo certbot --apache

# This script will prompt you to answer a series of questions in order to configure your SSL certificate.

# Add ServerName your_domain
        ServerAlias www.your_domain
Where needed
```

Step 5 — Verifying Certbot Auto-Renewal

#Let's Encrypt's certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process, as well as to ensure that misused certificates or stolen keys will expire sooner rather than later. The certbot package we installed takes care of renewals by including a renew script to /etc/cron.d, which is managed by a systemctl service called certbot.timer.

make sure it's active:

$ sudo systemctl status certbot.timer
To test the renewal process:
$ sudo certbot renew --dry-run