Install LEMP Stack on Ubuntu 22.04

To follow this tutorial, you need an Ubuntu 22.04 OS running on your local computer or on a remote server.

Step 1: Update Software Packages

```
sudo apt update
```

Step 2: Install Nginx Web Server

```
sudo apt install nginx
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
```

Check Nginx version.

Now type in the public IP address of your Ubuntu 22.04 server in the browser address bar. You should see the “Welcome to Nginx” Web page, which means Nginx Web server is running properly. If you are installing LEMP on your local Ubuntu 22.04 computer, then type `127.0.0.1` or `localhost` in the browser address bar.

If you are using the UFW firewall, then run this command to open TCP port 80.

```
sudo ufw enable
sudo ufw allow http
sudo ufw status
```

Finally, we need to make `www-data` (Nginx user) as the owner of web directory. By default, it’s owned by the ro ot user.

```
sudo chown www-data:www-data /usr/share/nginx/html -R
```

Step 3: Install MariaDB Database Server

Enter the following command to install MariaDB on Ubuntu 22.04.

```
sudo apt install mariadb-server mariadb-client
```

After it’s installed, MariaDB server should be automatically started. Use **systemctl** to check its status.

```
systemctl status mariadb
```

If it’s not running, start it with this command:

```
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

Now run the post-installation security script.

```
sudo mysql_secure_installation
```

enter MariaDB root password, press `Enter` key as the root password isn’t set yet.
Don’t switch to unix_socket authentication because MariaDB is already using unix_socket authentication.
Don’t change the root password
Next, you can press `Enter` to answer all remaining questions

So you can run the following command to log in without providing MariaDB root password.

```
sudo mariadb -u root
exit;
```

Check MariaDB server version information.

```
mariadb --version
```

Step 4: Install PHP8.1

PHP8.1 is included in Ubuntu 22.04 repository and has a minor performance improvement over PHP8.0. Enter the f ollowing command to install PHP8.1 and some common extensions.

```
sudo apt install php8.1 php8.1-fpm php8.1-mysql php-common php8.
1-cli php8.1-common php8.1-opcache php8.1-readline php8.1-mbstri
ng php8.1-xml php8.1-gd php8.1-curl
```

Installing these PHP extensions ensures that your CMS runs smoothly. Now start php8.1-fpm.

```
sudo systemctl start php8.1-fpm
sudo systemctl enable php8.1-fpm
systemctl status php8.1-fpm
```

Step 5: Create an Nginx Server Block

Remove the `default` symlink in `sites-enabled` directory

```
sudo rm /etc/nginx/sites-enabled/default
```

Then use a command-line text editor like Nano to create a new server block file under **/etc/nginx/conf.d/** directory.

```
sudo nano /etc/nginx/conf.d/default.conf
```

```
server {
  listen 80;
  listen [::]:80;
  server_name _;
  root /usr/share/nginx/html/;
  index index.php index.html index.htm index.nginx-debian.html;

  location / {
    try_files $uri $uri/ /index.php;
  }

  location ~ \.php$ {
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_
name;
    include fastcgi_params;
    include snippets/fastcgi-php.conf;
  }

 # A long browser cache lifetime can speed up repeat visits to y
our page
  location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js
|ico|xml)$ {
       access_log        off;
       log_not_found     off;
       expires           360d;
  }

  # disable access to hidden files
  location ~ /\.ht {
      access_log off;
      log_not_found off;
      deny all;
  }
}
```

```
sudo nginx -t
sudo systemctl reload nginx
```

Step 6: Test PHP

To test PHP-FPM with Nginx Web server, we need to create a `info.php` file in the webroot directory.

```
sudo nano /usr/share/nginx/html/info.php
<?php phpinfo(); ?>
```

Now in the browser address bar, enter `server-ip-address/info.php`

Step 7: Improve PHP Performance

The default PHP configurations (`/etc/php/8.1/fpm/php.ini`) are made for servers with very few resources (like a 256MB RAM server). To improve web application performance, you should change some of them.

We can edit the PHP config file (`php.ini`), but it’s a good practice to create a custom PHP config file, so when you upgrade to a new version of PHP8.1, your custom configuration will be preserved.

```
sudo nano /etc/php/8.1/fpm/conf.d/60-custom.ini
```

In this file, add the following lines.

```
; Maximum amount of memory a script may consume. Default is 128M
memory_limit = 512M

; Maximum allowed size for uploaded files. Default is 2M.
upload_max_filesize = 20M

; Maximum size of POST data that PHP will accept. Default is 2M.
post_max_size = 20M

; The OPcache shared memory storage size. Default is 128
opcache.memory_consumption=256

; The amount of memory for interned strings in Mbytes. Default i
s 8.
opcache.interned_strings_buffer=32
```

```
sudo systemctl reload php8.1-fpm
```

Congrats! You have successfully installed Nginx, MariaDB, and PHP8.1 on Ubuntu 22.04. For your server's securit y, you should delete `info.php` file now to prevent hackers from seeing it.

```
sudo rm /usr/share/nginx/html/info.php
```

Troubleshooting Tip

If you encounter errors, you can check the Nginx error log (`/var/log/nginx/error.log`) to find out wh at's wrong.

Nginx Automatic Restart

If for any reason your Nginx process is killed, you need to run the following command to restart it

```
sudo systemctl restart nginx
```

Instead of manually typing this command, we can make Nginx automatically restart by editing the `nginx.service` systemd service unit. To override the default systemd service configuration, we create a separate directory.

```
sudo mkdir -p /etc/systemd/system/nginx.service.d/
```

Then create a file under this directory.

```
sudo nano /etc/systemd/system/nginx.service.d/restart.conf
```

Add the following lines in the file, which will make Nginx automatically restart **5 seconds** after a failure is detected. The default value of `RetartSec` is **100ms**, which is too small. Nginx may complain that "start request repeated too quickly" if `RestartSec` is not big enough.

```
[Service]
Restart=always
RestartSec=5s
```

Save and close the file. Then reload systemd for the changes to take effect.

```
sudo systemctl daemon-reload
```

To check if this would work, kill Nginx with:

```
sudo pkill nginx
```

Then check Nginx status. You will find Nginx automatically restarted.

```
systemctl status nginx
```

MariaDB Automatic Start

By default, MariaDB is configured to automatically restart `on-abort` (`/lib/systemd/system/mariadb.service`). However, if your server runs out of memory (oom) and MariaDB is killed by the oom killer, it won't automatically restart. We can configure it to restart no matter what happens.

Create a directory to store custom configurations.

```
sudo mkdir -p /etc/systemd/system/mariadb.service.d/
```

Create a custom config file.

```
sudo nano /etc/systemd/system/mariadb.service.d/restart.conf
```

Add the following lines in the file.

```
[Service]
Restart=always
RestartSec=5s
```

Save and close the file. Then reload systemd for the changes to take effect.

```
sudo systemctl daemon-reload
```

Create User’s Home Directory

Create a user's home directory under the '/home' directory. You should set the root directory for the website to be '/h ome/username/websitename/public.'

```
adduser dani
usermod –aG sudo dani
rsync –archive –chown=dani:dani ~/.ssh /home/dani
sudo mkdir –p /home/dani/myweb/public
sudo chown –R dani:dani /home/dani/myweb
ls –l /home
ufw allow OpenSSH
ufw status
```

doing ssh

sudo root@localhost

if not connectied

vim /etc/ssh/sshd_config

vim /etc/ssh/sshd_config.d/60-cloudimg-settings.conf

ssh-keygen

systemctl restart sshd

ssh-copy-id root@localhost

passwd root

ssh root@localhost

Set Up vsftpd for a User’s Directory on Ubuntu 22.04

Step 1 — Installing vsftpd

install the `vsftpd` daemon:

```
sudo apt install vsftpd
```

When the installation is complete, copy the configuration file so you can start with a blank configuration, while also saving the original as a backup:

```
sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig

vim /etc/selinux/semanage.conf
```

Step 2 — Opening the Firewall

make adjustments to ensure that FTP traffic is permitted so firewall rules don’t block the tests.

```
sudo ufw status
sudo ufw enable
sudo ufw allow OpenSSH
sudo ufw allow 20/tcp
sudo ufw allow 21/tcp
sudo ufw allow 990/tcp
sudo ufw allow 40000:50000/tcp
sudo ufw status
```

Step 3 — Preparing the User Directory

create an `ftp` directory to serve as the `chroot` and a writable `files` directory to hold the actual files.

```
sudo mkdir /home/dani/ftp
sudo chown nobody:nogroup /home/dani/ftp
sudo chmod a-w /home/dani/ftp
sudo ls -la /home/dani/ftp
sudo mkdir /home/dani/ftp/files
sudo chown dani:dani /home/dani/ftp/files
sudo ls -la /home/dani/ftp
echo "vsftpd test file" | sudo tee /home/dani/ftp/files/test.txt
```

Step 4 — Configuring FTP Access

In this step, you will allow a single user with a local shell account to connect with FTP. The two key settings for this are already set in `vsftpd.conf`.

```
sudo vim /etc/vsftpd.conf
```

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
dirmessage_enable=YES
use_localtime=YES
xferlog_enable=YES
connect_from_port_20=YES

chroot_local_user=YES

user_sub_token=$USER
local_root=/home/$USER/ftp
pasv_min_port=40000
pasv_max_port=50000
pasv_min_port=40000
pasv_max_port=50000
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
```

Finally, add your user to /etc/vsftpd.userlist. Use the -a flag to append to the file

```
echo "sammy" | sudo tee -a /etc/vsftpd.userlist
cat /etc/vsftpd.userlist
sudo systemctl restart vsftpd
```

Step 5 — Testing FTP Access

```
ftp -p 203.0.113.0

dani

dani@1
```

Step 6 — Securing Transactions

Since FTP does *not* encrypt any data in transit, including user credentials, you can enable TLS/SSL to provide that encryption. The first step is to create the SSL certificates for use with `vsftpd`.

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

common name : your_server_ip

```
sudo nano /etc/vsftpd.conf
```

```
    #rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
    #rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key

    rsa_cert_file=/etc/ssl/private/vsftpd.pem
    rsa_private_key_file=/etc/ssl/private/vsftpd.pem
    ssl_enable=YES
    allow_anon_ssl=NO
    force_local_data_ssl=YES
    force_local_logins_ssl=YES
    ssl_tlsv1=YES
    ssl_sslv2=NO
    ssl_sslv3=NO
    require_ssl_reuse=NO
    ssl_ciphers=HIGH
```

```
sudo systemctl restart vsftpd
```

Step 7 — Testing TLS with FileZilla

Install phpMyAdmin with Nginx (LEMP) on Ubuntu 22.04 LTS

Step 1: Download phpMyAdmin on Ubuntu 22.04 Server

phpMyAdmin is included in Ubuntu 22.04 software repository, but it's better to install the latest version using the upstream package.

```
wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-
languages.zip
```

```
sudo apt install unzip
unzip phpMyAdmin-latest-all-languages.zip
```

Move phpMyadmin to `/home/dani/myweb/public/` directory.

```
sudo mv phpMyAdmin-5.2.1-all-languages /home/dani/myweb/public/p
ma.myphp.com
```

Then make the web server user (`www-data`) as the owner of this directory.

```
sudo chown -R www-data:www-data /home/dani/myweb/public/pma.myph
p.com
```

Step 2: Create a MariaDB Database and User for phpMyAdmin

Log in to MariaDB console.

```
sudo mysql -u root
```

Create a new database for phpMyAdmin

```
CREATE DATABASE phpmyadmin DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

create the `dani` database user and set a password, and at the same time grant all permission of the new database to the new user so later on phpMyAdmin can write to the database.

```
GRANT ALL ON phpmyadmin.* TO 'dani'@'localhost' IDENTIFIED BY 'd
ani@1';

FLUSH PRIVILEGES;

EXIT;
```

Step 3: Install Required and Recommended PHP Modules.

Run the following command to install PHP modules required or recommended by phpMyAdmin.

```
sudo apt install php-imagick php-phpseclib php-php-gettext php8.
1-common php8.1-mysql php8.1-gd php8.1-imap php8.1-curl php8.1-z
ip php8.1-xml php8.1-mbstring php8.1-bz2 php8.1-intl php8.1-gmp
```

Step 4: Create Nginx Server Block for phpMyAdmin

To be able to access the phpMyAdmin web interface, we need to create a Nginx server block by running the followi
ng command.

```
sudo nano /etc/nginx/conf.d/phpmyadmin.conf
```

```
server {
  listen 80;
  listen [::]:80;
  server_name pma.myphp.com;
  root /home/dani/myweb/public/pma.myphp.com;
  index index.php index.html index.htm index.nginx-debian.html;

  access_log /var/log/nginx/phpmyadmin_access.log;
  error_log /var/log/nginx/phpmyadmin_error.log;

  location / {
    try_files $uri $uri/ /index.php;
  }

  location ~ ^/(doc|sql|setup)/ {
    deny all;
  }

  location ~ \.php$ {
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_
name;
    include fastcgi_params;
    include snippets/fastcgi-php.conf;
  }

  location ~ /\.ht {
    deny all;
  }
}
```

Your phpMyAdmin files are in `/home/dani/myweb/public/pma.myphp.com` directory. Save a nd close the file.

create DNS A record for it.

```
vim /etc/hosts
```

```
C:\Windows\System32\drivers\etc
```

Then test Nginx configurations.

```
sudo nginx -t
sudo systemctl reload nginx
```

Now you should be able to access phpMyAdmin web interface via

```
pma.myphp.com
```

Installing SSL Certificate

To secure the phpMyadmin web interface, we can install a free Let’s Encrypt TLS certificate. Install the Let’s Encrypt client from Ubuntu 22.04 software repository like below:

```
sudo apt install certbot python3-certbot-nginx
```

`Python3-certbot-nginx` is the Nginx plugin for Certbot. Now run the following command to obtain and install TLS certificate.

```
sudo certbot --nginx --agree-tos --redirect --hsts --staple-ocsp
-d pma.myphp.com --email danielvadasserikkara@gmail.com
```

Install WordPress on Ubuntu 22.04 with Nginx, MariaDB, PHP8.1

Step 1: Download WordPress

go to [wordpress.org download page]() and download the zip archive. You can acquire the direct download link by right-clicking the download button and select `copy link location`.

```
wget https://wordpress.org/latest.zip
```

extract the zip archive

```
sudo apt install unzip
sudo mkdir -p /usr/share/nginx
sudo unzip latest.zip -d /usr/share/nginx/
```

The archive will be extracted to `/home/dani/myweb/public/` directory. A new directory named `wordpress` will be created (/usr/share/nginx/wordpress). Now we can rename it like below, so it's easy for us to identify each directory. Replace `daniwpress.com` with your real domain name.

```
sudo mv /usr/share/nginx/wordpress /home/dani/myweb/public/daniwpress.com
```

Step 2: Create a Database and User for WordPress Site

```
sudo mariadb -u root
```

```
create database wordpress;
create user daniel@localhost identified by 'dani@11';
grant all privileges on wordpress.* to daniel@localhost;
flush privileges;
exit;
```

Step 3: Configure WordPress

Go to your WordPress directory.

```
cd /home/dani/myweb/public/daniwpress.com/
```

Copy the sample configuration file and rename it to `wp-config.php`.

```
sudo cp wp-config-sample.php wp-config.php
```

Now edit the new config file

```
sudo vim wp-config.php
```

```
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'daniel');

/** MySQL database password */
define('DB_PASSWORD', 'dani@11');
```

Then scroll down to find the following line.

```
>>   $table_prefix = 'wp_';
```

It's highly recommended to change it to something else to improve security. Use random characters

```
>>   $table_prefix = '9OzB3g_';
```

only the file owner can read this file.

```
sudo chmod 640 wp-config.php
```

We also need to set the Nginx user (`www-data`) as the owner of the WordPress site directory

```
sudo chown www-data:www-data /home/dani/myweb/public/daniwpress.
com/ -R
```

Step 4: Create an Nginx Server Block for WordPress

We will create the server block file `/etc/nginx/conf.d/`directory, The file name must end with `.conf`

```
sudo vim /etc/nginx/conf.d/daniwpress.com.conf
```

```
server {
  listen 80;
  listen [::]:80;
  server_name www.daniwpress.com daniwpress.com;
  root /home/dani/myweb/public/daniwpress.com/;
  index index.php index.html index.htm index.nginx-debian.html;

  error_log /var/log/nginx/wordpress.error;
  access_log /var/log/nginx/wordpress.access;

  location / {
    try_files $uri $uri/ /index.php;
  }

   location ~ ^/wp-json/ {
     rewrite ^/wp-json/(.*?)$ /?rest_route=/$1 last;
   }

  location ~* /wp-sitemap.*\.xml {
    try_files $uri $uri/ /index.php$is_args$args;
  }

  error_page 404 /404.html;
  error_page 500 502 503 504 /50x.html;

  client_max_body_size 20M;

  location = /50x.html {
    root /usr/share/nginx/html;
  }

  location ~ \.php$ {
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_
name;
    include fastcgi_params;
    include snippets/fastcgi-php.conf;
    fastcgi_buffers 1024 4k;
    fastcgi_buffer_size 128k;
  }

  #enable gzip compression
  gzip on;
  gzip_vary on;
  gzip_min_length 1000;
  gzip_comp_level 5;
  gzip_types application/json text/css application/x-javascript
application/javascript image/svg+xml;
  gzip_proxied any;
  # A long browser cache lifetime can speed up repeat visits to
your page
  location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js
|ico|xml)$ {
       access_log        off;
       log_not_found     off;
       expires           360d;
  }

  # disable access to hidden files
  location ~ /\.ht {
      access_log off;
      log_not_found off;
      deny all;
  }
}
```

Then test Nginx configurations.

```
sudo nginx -t

sudo systemctl reload nginx
```

Enter your domain name in the browser address bar.

www.daniwpress.com