

How to Install Nginx, MariaDB & PHP (LEMP) on Debian 11

Learn how to install and configure Nginx, MariaDB, and PHP (LEMP stack) on Debian 11 with our step-by-step guide for a powerful web server environment.

Contents

01	Introduction	3
02	Prerequisites	3
03	1. Install Nginx	3
04	2. Configure the Firewall	4
05	3. Create an Nginx Virtual Host	5
06	4. Install MariaDB	6
07	5. Secure MariaDB Database Server	6
08	6. Install PHP	9
09	7. Install a Let's Encrypt Certificate	10

Introduction

The LEMP stack (Linux, Nginx, MySQL/MariaDB, and PHP) is a free, open-source web application stack used to develop and deploy web applications. The LEMP Stack is like the LAMP stack, but it substitutes Nginx for Apache web server. This guide explains how to install a LEMP stack on Debian 11 and use Certbot to secure it with a Let's Encrypt TLS/SSL certificate.

Prerequisites

- Deploy a [Debian 11](#) cloud server.
- Create a non-root user with sudo access and log in with SSH.
- [Update](#) the Debian Server.

This guide assumes you want to use both the apex domain `example.com` and the `www.example.com` hostname for your server. To follow this guide, you should assign both the apex domain (sometimes referred to as `@`) and the `www` hostname to the server's IP address in your DNS settings.

1. Install Nginx

1. Install the Nginx web server.

```
$ sudo apt-get install nginx -y
```

2. Start the Nginx service.

```
$ sudo systemctl start nginx
```

3. Enable the Nginx service to start at system reboot.

```
$ sudo systemctl enable nginx
```

4. Check the Nginx version to verify the installation.

```
$ sudo nginx -v
```

You should see output like this:

```
$ nginx version: nginx/1.18.0
```

2. Configure the Firewall

1. List the available application profiles.

```
$ sudo ufw app list
```

Among the other entries, you should see the following profiles:

```
Nginx Full  
Nginx HTTP  
Nginx HTTPS
```

* The **Nginx Full** profile opens both HTTPS (443) and HTTP (80) ports.

- The **Nginx HTTP** profile opens the HTTP (80) port only.
- The **Nginx HTTPS** profile opens the HTTPS (443) port only.

2. Allow the **Nginx Full** profile in the firewall. Certbot requires ports 80 and 443 to install a Let's Encrypt TLS/SSL certificate.

```
$ sudo ufw allow 'Nginx Full'
```

3. Check the Firewall status.

```
$ sudo ufw status
```

You should see output like this:

To	Action	From
--	-----	----
22	ALLOW	Anywhere
Nginx Full	ALLOW	Anywhere
22 (v6)	ALLOW	Anywhere (v6)
Nginx Full (v6)	ALLOW	Anywhere (v6)

3. Create an Nginx Virtual Host

1. Remove the default Nginx configuration.

```
$ sudo rm -rf /etc/nginx/sites-enabled/default
$ sudo rm -rf /etc/nginx/sites-available/default
```

2. Create an Nginx virtual host configuration file. Replace `your-domain-name.com` with your domain name.

```
$ sudo nano /etc/nginx/sites-available/your-domain-name.com
```

3. Paste this into the file. Replace `example.com` with your domain name.

```
server {

    listen 80;
    server_name example.com www.example.com;

    root /var/www/html;
    index index.php index.html index.nginx-debian.html;
    access_log /var/log/nginx/example_access.log;
    error_log /var/log/nginx/example_error.log;

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

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_index index.php;
        include fastcgi_params;
    }
}
```

```
fastcgi_pass unix:/run/php/php7.4-fpm.sock;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}

}
```

4. Enable the new Nginx configuration. Replace `example.com` with your domain name.

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
example.com
```

5. Reload the Nginx service.

```
$ sudo systemctl reload nginx
```

4. Install MariaDB

1. Install MariaDB database server.

```
$ sudo apt-get install mariadb-server -y
```

2. Start the MariaDB service.

```
$ sudo systemctl start mariadb
```

3. Enable the MariaDB service to start at system reboot.

```
$ sudo systemctl enable mariadb
```

5. Secure MariaDB Database Server

1. MariaDB provides a security script to secure the database. Run it and answer all the security questions as shown.

```
$ sudo mysql_secure_installation
```

Initially, there is no password for root. Press Enter.

```
Enter current password for root (enter for none):  
OK, successfully used password, moving on...
```

Press Y to Switch to unix_socket authentication.

```
Switch to unix_socket authentication [Y/n] Y  
Enabled successfully!  
Reloading privilege tables..  
... Success!
```

Press Y to change the root password.

```
Change the root password? [Y/n] Y  
New password:  
Re-enter new password:  
Password updated successfully!  
Reloading privilege tables..  
... Success!
```

Press Y to remove anonymous users.

```
Remove anonymous users? [Y/n] Y  
... Success!
```

Press Y to remove remote root login.

```
Disallow root login remotely? [Y/n] Y  
... Success!
```

Press Y to remove test database and access to it.

```
Remove test database and access to it? [Y/n] Y  
- Dropping test database...  
... Success!
```

```
- Removing privileges on test database...  
... Success!
```

Press Y to reload the privilege tables.

```
Reload privilege tables now? [Y/n] Y  
... Success!  
  
Cleaning up...  
  
All done! If you've completed all of the above steps, your MariaDB  
installation should now be secure.  
  
Thanks for using MariaDB!
```

2. Connect to the MariaDB shell and enter your MariaDB root password.

```
$ sudo mysql -u root -p
```

3. Check the MariaDB version to verify the installation.

```
MariaDB [(none)]> SELECT @@version;
```

It should return something like this:

```
+-----+  
| @@version          |  
+-----+  
| 10.5.12-MariaDB-0+deb11u1 |  
+-----+  
1 row in set (0.000 sec)
```

4. Exit MariaDB shell.

```
MariaDB [(none)]> exit
```

6. Install PHP

1. Install PHP-FPM 7.4 and other required packages.

```
$ sudo apt-get install php php-fpm php-curl php-cli php-zip php-mysql php-xml -y
```

2. Check the PHP version to verify the installation.

```
$ php -v
```

It should return something like this:

```
PHP 7.4.28 (cli) (built: Feb 17 2022 16:17:19) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.28, Copyright (c), by Zend Technologies
```

3. Create a PHP test file in your editor.

```
$ sudo nano /var/www/html/phpinfo.php
```

4. Paste this into your `phpinfo.php` file.

```
<?php

phpinfo();

?>
```

5. Save and exit the file.
6. In your browser, navigate to `http://www.example.com/phpinfo.php` to view the PHP test file, which shows the PHP information.

7. Install a Let's Encrypt Certificate

1. Certbot requires Snap. Install `snapd` and enable classic Snap support.

```
$ sudo apt install snapd
```

2. Either log out and back in again, or restart your system, to update Snap's paths.

3. Install the core Snap to get the latest `snapd`.

```
$ sudo snap install core
```

4. Update core Snap.

```
$ sudo snap refresh core
```

5. Verify there are no Certbot packages installed with `apt`.

```
$ sudo apt remove certbot
```

6. Install Certbot with Snap.

```
$ sudo snap install --classic certbot
```

7. Link Certbot to `/usr/bin`.

```
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

8. Request a certificate for your server. Replace the example email and domains your values. The command shown requests a multi-domain ([SAN](#)) certificate for `example.com` and `www.example.com`.

```
$ sudo certbot --nginx --agree-tos --redirect --email your_email@example.com -d example.com -d www.example.com
```

9. Test your SSL configuration on an SSL check website like [SSL Labs](#).

10. Navigate to your website and verify the SSL certificate works as expected.

This completes the initial setup of your LEMP server. The server is ready for you to install applications that work with LEMP, or develop your own.



VULTR

