debian-lamp-howto

Before you start this step, make sure you have completed the installation of Debian 11.7 and are able to log in both as root and your desired username.

This is a step by step instruction on how to install the infamous LAMP-stack (Linux Apache MariaDB PHP) on Debian 11.7.

https://wiki.debian.org/LaMp - we're following the official Debian guide (mostly)

We will be skipping some steps to focus on a minimal viable solution

Just type in the commands

Commands

Update the system:

```
apt update && apt upgrade
```

Install MariaDB

```
apt install mariadb-server mariadb-client
# mysql_secure_installation #uncomment this line for improved security
```

Create database and user

You should create 1 user that has full access to 1 db with the same name.

Replace `wiki` with your preferred databasename / username

Optionally replace `localhost` with the IP-address of your database client (for example a webserver)

```
CREATE DATABASE wiki;
CREATE USER 'wiki'@'localhost' IDENTIFIED BY 'Yolo123456';
GRANT ALL PRIVILEGES ON wiki.* TO 'wiki'@'localhost';
FLUSH PRIVILEGES;
```

Apache

```
apt install apache2 apache2-doc
systemctl restart apache2
systemctl enable apache2
```

PHP

```
apt install php php-mysql
```

PHPMyAdmin

```
apt install phpmyadmin
echo "Include /etc/phpmyadmin/apache.conf" >> /etc/apache2/apache2.conf # activate it
systemctl restart apache2
```

Congratiulations, it should now be running!

Now all you have to do is copy your files over, good luck!

Copying files to the server

There are basically 2 ways to copy our files over to the server.

1. Using `scp` aka "secure copy" to manually move the files we need from our host-pc to the web-server
2. Installing `git` on the server and manually pulling our repository over

Copying over files using `scp`

`scp` stands for "secure copy" and is a unix-based tool to copy files between UNIX-like and Linux-based systems.

In recent versions of Windows 10, scp now comes as default with windows, you can check to see if you have it by typing `scp` in the command line.

The great thing about scp is that it uses only port 22 aka SSH to copy the files over. Which is great since we already have that service running.

The syntax of `scp` is a mix between `cp` (normal copy which takes in 2 parameters: `source` and `destination`), and `ssh` which uses the `username`@`hostname` syntax.

So lets say that I am on my Windows-system and I want to copy over the folder `wiki`. The command I would use is:

```
scp wiki arvidj@<ip of server>:
```

Notice the `:` in the end.

Now we can verify that the files have been moved over by using `ls` in our home directory on the server.

Now that we have copied our application-files to our home-folder, the full path to our application is `/home/arvidj/wiki`.

All we have to do now is to point the current root of the webserver (`/var/www/html`) to this directory. This can be done by logging in as root `sudo -i` and then making a link from the current root-directory to our newly clone git repo.

```
ln -s /home/arvidj/wiki/ /var/www/html
```

Now after you restart the apache2-service, the files should be served by the web-server. You can test this by accessing the web-server `http://<ip of web-server>/name of project`

Great success!

> Remember that you will have to create the tables in your database or it will not work!

Exporting and importing our database

To get our database up and running with the tables and data we had on our host-pc, we need to do 3 things. Btw this is also how we backup a mysql-server:

1. Export the database to a .sql file
2. Move the .sql file over to our server
3. Import the .sql file on the new server

We can use the tool `mysqldump` to achieve this.

To export:

```
mysqldump -u <username> <databasename> > exported_db_file.sql
```

Now move the file over to the server using your preferred method (hopefully scp)

To import:

```
mariadb -u <username> <databasename> -p < exported_db_file.sql
```

This will re-create all the tables and data that you had in your database in the first place.

You can play around with the `--no-data` argument on `mysqldump` if you only want to export the tables, not the data.