How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu

For Ubuntu 12.04

About LAMP

LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP. Since the virtual private server is already running Ubuntu, the linux part is taken care of. Here is how to install the rest.

Set Up

The steps in this tutorial require the user to have root privileges on your VPS.

Step One – Install Apache

Apache is a free open source software which runs over 50% of the world's web servers.

To install apache, open terminal and type in these commands:

```
sudo apt-get update
sudo apt-get install apache2
```

That's it. To check if Apache is installed, direct your browser to your server's IP address (eg. http://12.34.56.789). The page should display the words "It works!" like this.

How to Find your Server's IP address

You can run the following command to reveal your server's IP address.

```
ifconfig eth0 | grep inet | awk '{ print $2 }'
```

Step Two – Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data

To install MySQL, open terminal and type in these commands:

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
```

During the installation, MySQL will ask you to set a root password. If you miss the chance to set the password while the program is installing, it is very easy to set the password later from within the MySQL shell.

Once you have installed MySQL, we should activate it with this command:

```
sudo mysql_install_db
```

Finish up by running the MySQL set up script:

```
sudo /usr/bin/mysql_secure_installation
```

The prompt will ask you for your current root password. Type it in.

```
Enter current password for root (enter for none):

OK, successfully used password, moving on...
```

Then the prompt will ask you if you want to change the root password. Go ahead and choose N and move on to the next steps.

It’s easiest just to say Yes to all the options. At the end, MySQL will reload and implement the new changes.

```
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? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

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

By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
... Success!
 - Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...
```

Once you're done with that you can finish up by installing PHP.

Step Three – Install PHP

PHP is an open source web scripting language that is widely use to build dynamic webpages.

To install PHP, open terminal and type in this command.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

After you answer yes to the prompt twice, PHP will install itself.

It may also be useful to add php to the directory index, to serve the relevant php index files:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Add index.php to the beginning of index files. The page should now look like this:

```
<IfModule mod_dir.c>

     DirectoryIndex index.php index.html index.cgi index.pl index.php
     index.xhtml index.htm

</IfModule>
```

PHP Modules

PHP also has a variety of useful libraries and modules that you can add onto your virtual server. You can see the libraries that are available.

```
apt-cache search php5-
```

Terminal will then display the list of possible modules. The beginning looks like this:

```
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
php5-cli - command-line interpreter for the php5 scripting language
php5-common - Common files for packages built from the php5 source
php5-curl - CURL module for php5
php5-dbg - Debug symbols for PHP5
php5-dev - Files for PHP5 module development
php5-gd - GD module for php5
php5-gmp - GMP module for php5
php5-ldap - LDAP module for php5
php5-mysql - MySQL module for php5
php5-odbc - ODBC module for php5
php5-pgsql - PostgreSQL module for php5
php5-pspell - pspell module for php5
php5-recode - recode module for php5
php5-snmp - SNMP module for php5
php5-sqlite - SQLite module for php5
php5-tidy - tidy module for php5
php5-xmlrpc - XML-RPC module for php5
php5-xsl - XSL module for php5
php5-adodb - Extension optimising the ADOdb database abstraction library
php5-auth-pam - A PHP5 extension for PAM authentication
[...]
```

Once you decide to install the module, type:

```
sudo apt-get install name of the module
```

You can install multiple libraries at once by separating the name of each module with a space.

Congratulations! You now have LAMP stack on your droplet!

Step Four – RESULTS: See PHP on your Server

Although LAMP is installed, we can still take a look and see the components online by creating a quick php info page

To set this up, first create a new file:

```
sudo nano /var/www/info.php
```

Add in the following line:

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

Then Save and Exit.

Restart apache so that all of the changes take effect:

```
sudo service apache2 restart
```

Finish up by visiting your php info page (make sure you replace the example ip address with your correct one): http://12.34.56.789/info.php

How To Install and Secure phpMyAdmin on Ubuntu 12.04

About phpMyAdmin

phpMyAdmin is an free web software to work with MySQL on the web—it provides a convenient visual front end to the MySQL capabilities.

Setup

The steps in this tutorial require the user to have root privileges on your virtual private server.

Before working with phpMyAdmin you need to have LAMP installed on your server. Once you have the user and required software, you can start installing phpMyAdmin on your VPS!

Install phpMyAdmin

The easiest way to install phpmyadmin is through apt-get:

```
sudo apt-get install phpmyadmin apache2-utils
```

During the installation, phpMyAdmin will walk you through a basic configuration. Once the process starts up, follow these steps:

- Select Apache2 for the server
- Choose YES when asked about whether to Configure the database for phpmyadmin with dbconfig-common
- Enter your MySQL password when prompted
- Enter the password that you want to use to log into phpmyadmin

After the installation has completed, add phpmyadmin to the apache configuration.

```
sudo nano /etc/apache2/apache2.conf
```

Add the phpmyadmin config to the file.

```
Include /etc/phpmyadmin/apache.conf
```

Restart apache:

```
sudo service apache2 restart
```

You can then access phpmyadmin by going to youripaddress/phpmyadmin.

Security

Unfortunately older versions of phpMyAdmin have had serious security vulnerabilities including allowing remote users to eventually exploit root on the underlying virtual private server. One can prevent a majority of these attacks through a simple process: locking down the entire directory with

Apache's native user/password restrictions which will prevent these remote users from even attempting to exploit older versions of phpMyAdmin.

Set Up the .htaccess File

To set this up start off by allowing the .htaccess file to work within the phpmyadmin directory. You can accomplish this in the phpmyadmin configuration file:

```
sudo nano /etc/phpmyadmin/apache.conf
```

Under the directory section, add the line “AllowOverride All” under “Directory Index”, making the section look like this:

```
<Directory /usr/share/phpmyadmin>
      Options FollowSymLinks
      DirectoryIndex index.php
      AllowOverride All
      [...]
```

Configure the .htaccess file

With the .htaccess file allowed, we can proceed to set up a native user whose login would be required to even access the phpmyadmin login page.

Start by creating the .htaccess page in the phpmyadmin directory:

```
sudo nano /usr/share/phpmyadmin/.htaccess
```

Follow up by setting up the user authorization within .htaccess file. Copy and paste the following text in:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/apache2/.phpmyadmin.htpasswd
Require valid-user
```

Below you’ll see a quick explanation of each line

- **AuthType:** This refers to the type of authentication that will be used to the check the passwords. The passwords are checked via HTTP and the keyword Basic should not be changed.
- **AuthName:** This is text that will be displayed at the password prompt. You can put anything here.
- **AuthUserFile:** This line designates the server path to the password file (which we will create in the next step.)
- **Require valid-user:** This line tells the .htaccess file that only users defined in the password file can access the phpMyAdmin login screen.

Create the htpasswd file

Now we will go ahead and create the valid user information.

Start by creating a htpasswd file. Use the htpasswd command, and place the file in a directory of your choice as long as it is not accessible from a browser. Although you can name the password file

whatever you prefer, the convention is to name it .htpasswd.

```
sudo htpasswd -c /etc/apache2/.phpmyadmin.htpasswd username
```

A prompt will ask you to provide and confirm your password.

Once the username and passwords pair are saved you can see that the password is encrypted in the file.

Finish up by restarting apache:

```
sudo service apache2 restart
```

Accessing phpMyAdmin

phpMyAdmin will now be much more secure since only authorized users will be able to reach the login page. Accessing youripaddress/phpmyadmin should display a screen like this.

Fill it in with the username and password that you generated. After you login you can access phpmyadmin with the MySQL username and password.

How To Create a SSL Certificate on Apache for Ubuntu 12.04

About SSL Certificates

A SSL certificate is a way to encrypt a site's information and create a more secure connection. Additionally, the certificate can show the virtual private server's identification information to site visitors. Certificate Authorities can issue SSL certificates that verify the server's details while a self-signed certificate has no 3rd party corroboration.

Set Up

The steps in this tutorial require the user to have root privileges on the VPS. You can see how to set that up here in steps 3 and 4.

Additionally, you need to have apache already installed and running on your virtual server. If this is not the case, you can download it with this command:

```
sudo apt-get install apache2
```

Step One—Activate the SSL Module

The next step is to enable SSL on the droplet.

```
sudo a2enmod ssl
```

Follow up by restarting Apache.

```
sudo service apache2 restart
```

Step Two—Create a New Directory

We need to create a new directory where we will store the server key and certificate

```
sudo mkdir /etc/apache2/ssl
```

Step Three—Create a Self Signed SSL Certificate

When we request a new certificate, we can specify how long the certificate should remain valid by changing the 365 to the number of days we prefer. As it stands this certificate will expire after one year.

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

With this command, we will be both creating the self-signed SSL certificate and the server key that protects it, and placing both of them into the new directory.

This command will prompt terminal to display a lists of fields that need to be filled in.

The most important line is "Common Name". Enter your official domain name here or, if you don't have one yet, your site's IP address.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:New York
Locality Name (eg, city) []:NYC
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Awesome Inc
Organizational Unit Name (eg, section) []:Dept of Merriment
Common Name (e.g. server FQDN or YOUR name) []:example.com
Email Address []:webmaster@awesomeinc.com
```

Step Four—Set Up the Certificate

Now we have all of the required components of the finished certificate.The next thing to do is to set up the virtual hosts to display the new certificate. Open up the SSL config file:

```
nano /etc/apache2/sites-available/default-ssl
```

Within the section that begins with <VirtualHost _default_:443>, quickly make the following changes. Add a line with your server name right below the Server Admin email:

```
ServerName example.com:443
```

Replace example.com with your DNS approved domain name or server IP address (it should be the same as the common name on the certificate). Find the following three lines, and make sure that they match the extensions below:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key
```

Save and Exit out of the file.

Step Five—Activate the New Virtual Host

Before the website that will come on the 443 port can be activated, we need to enable that Virtual Host:

```
sudo a2ensite default-ssl
```

You are all set. Restarting your Apache server will reload it with all of your changes in place.

```
sudo service apache2 reload
```

In your browser, type https://youraddress, and you will be able to see the new certificate.