2014

Hands On Labs+

LAMP Stack on Ubuntu 13.10 With PHPMyAdmin Setup

Table of Contents

Introduction

Linux, Apache, MySQL (increasingly MariaDB) and PHP – the magic application "stack" that is most commonly identified with a Linux Server. No other operating system or stack of applications or utilities is more commonly associated with each other and no other group handles as much world wide internet traffic as these four components do.

We will talk about the setup and configuration of this application stack in Ubuntu 13.10, as well as how to manage it locally and remotely. Finally, we will also use what we install to create a PHP based test site and the PhpMyAdmin application to show off our newly configured server.

Goals

This Hands On Lab will show you how to install and configuration the LAMP stack on Ubuntu 13.10. Once we confirm that all the services are working as expected, we will then download and install an application called "PHPMyAdmin" that will showcase all of the work we just completed.

We are going to be using Ubuntu 13.10 for this installation but this process applies equally to Debian in general. However, the process is a bit more complex in RPM based distributions like Red Hat and CentOS. In those cases, each of the components need to be installed separately, we will list those packages below in the prerequisite section.

Packages, Resources and Prerequisites

- Apache 2.2+ Web Server (RPM)
- MySQL (or MariaDB) (RPM)
- PHP 5.x (RPM)
- Lamp Server (Debian)

The resources you will be accessing during the course of this lab are:

- Ubuntu 13.10 Server: Test Client
    - You will use this to install our application stack and then access the installation and test availability.

Prerequisites to this lab:

- A LinuxAcademy.com Lab+ Subscription
- Internet Access and SSH Client
    - You will need to connect to the public IP of the server in order to install and configure LAMP and the PHPMyAdmin
        - SSH client can be Windows (i.e. Putty) or from another Linux system shell

    - For testing, use any browser from a system with internet access
- Login Information (Provided When The Server Starts Up)

Document Conventions

Just a couple of housekeeping items to go over so you can get the most out of this Hands On Lab without worrying about how to interpret the contents of the document.

When we are demonstrating a command that you are going to type in your shell while connected to a server, you will see a box with a dark blue background and some text, like this:

```
linuxacademy@ip-10-0-0-0:~$ sudo apt-get install package
[sudo] password for linuxacademy: <PASSWORD PROVIDED>
```

That breaks down as follows:

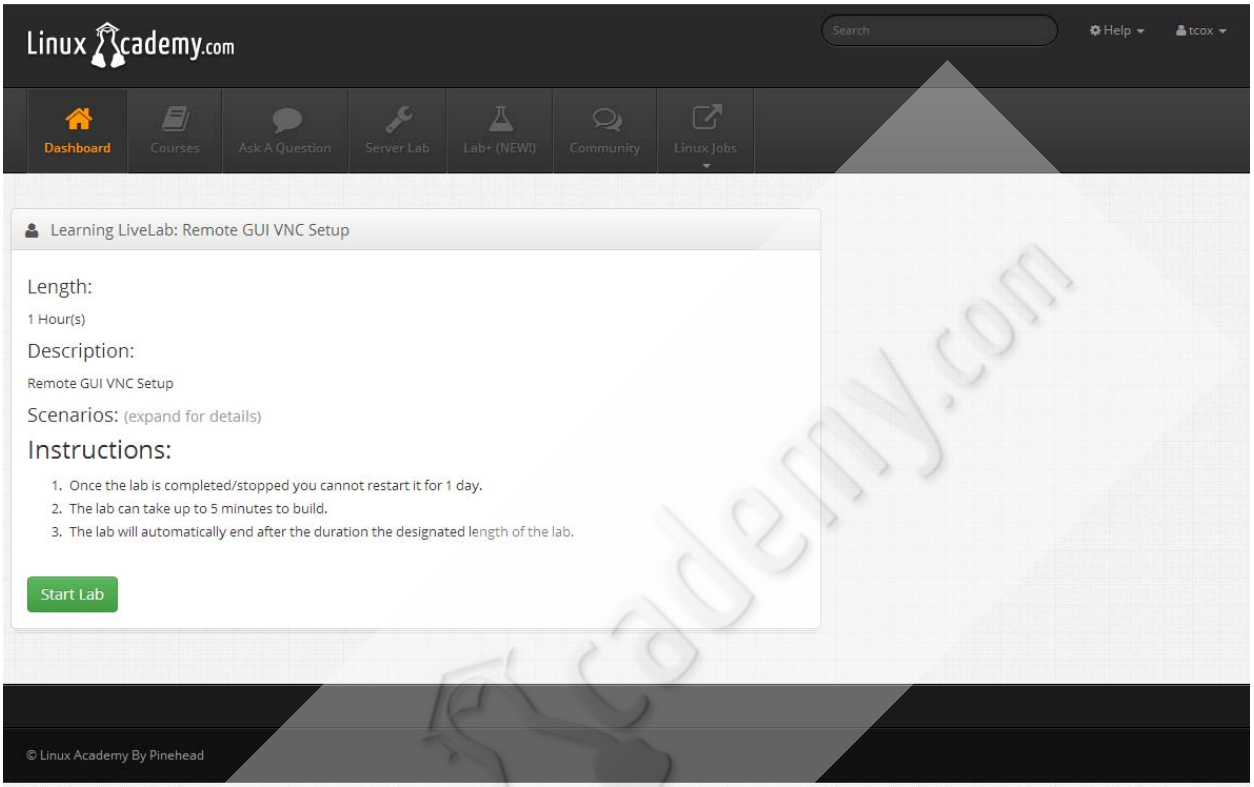
- The white text that looks something like "linuxacademy@ip-10-0-0-0:~$: "should be interpreted as the console prompt your cursor will be sitting at when you are logged into the server. You are not typing that into the console, it is just there. Note that the portion that says "ip-10-0-0-0" will be different for you based on the IP address of your system.
- The bold yellow text in any command example is the actual command you will type in your shell.
- Any lines subsequent to the bold yellow command that you type in is to be considered as the response you can expect from your command entry.

If you do not see the command prompt as the first line in the example, then all the white text is an example of a text, script or configuration file and is intended to be typed in its entirety (in the event you are instructed to create a previously non-existent file) or to compare against the contents of the file on your system.

One final convention, if you see a "~" at the end of a line in a command or text example, that will indicate that the line overflowed the end of line. Meaning you should just keep typing without hitting enter to start a new line until the natural end of the command.

General Process

When you are ready to begin the Hands On Lab, log into your Linux Academy Lab+ subscription and navigate to the "Live Labs" section on the Dashboard. Once you choose the "LAMP Stack" Lab from the list, you will see the screen below:

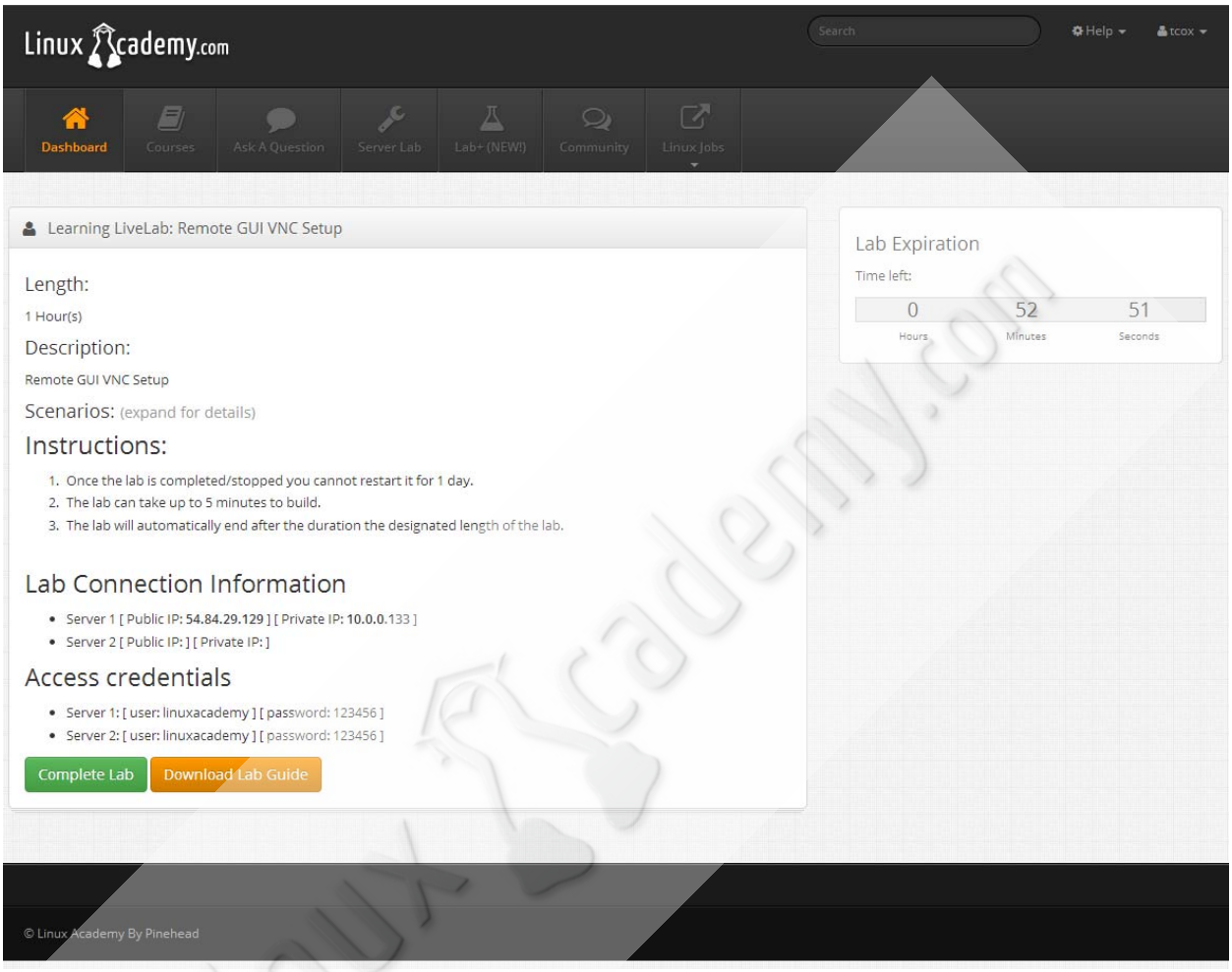


[Linux Academy Live Labs+ - Start Screen]

A few things to note before you start this process:

- When you launch the lab from this screen, it may take up to FIVE MINUTES for your servers to be deployed and be available for your use.
- Do not leave your desktop and come back, once the servers are launched, you will only have THREE HOURS to complete this lab from start to finish. After that point, the servers time out and are deleted permanently. Any and all work that you have done will then be lost.
- You can only use this lab ONCE PER DAY. If you try to use it more than that after completing the lab or the servers timing out, the screen will tell you when it will be available to you again.
- Other than those descriptions, you may retry any of the Labs+ labs as many times as you wish as long as you are a subscriber.

Once you have clicked on the 'Start Lab' button that you see above, a process will launch on our servers that will deploy the two servers we will use in our lab for testing. After a few minutes of processing (and you will see a status message that says "Creating Lab… Please Wait"), you should see a screen that looks like this one:



[Linux Academy Live Labs+ - Start Screen]

You will see all the information you need to access your servers from another system. Specifically, you need:

- The server public IP address
- Access credentials

One thing to note is that, in addition to the IP that you see above, the server will have another IP assigned to it in the 10.0.0.x subnet. This is a private IP address and will not route outside of your private server pool. Your server will have a static private address. We will be using the external IP address to connect over SSH as well as when configuring the LAMP application stack.

Linux Academy Lab Server – Server Installation and Configuration

Connect to your Linux Academy Lab server over SSH using any Windows or Linux SSH client or shell. We will generate all of our configuration and component installations from the command line while connected to the Ubuntu 13.10 Server in this lab.

LAMP Stack Installation

Once we are logged into our Linux Academy Lab Server, we will start by installing the LAMP components. Fortunately, Ubuntu makes this process very easy for us since it packages this stack of applications in a single group install. Open a command prompt and type in the following:

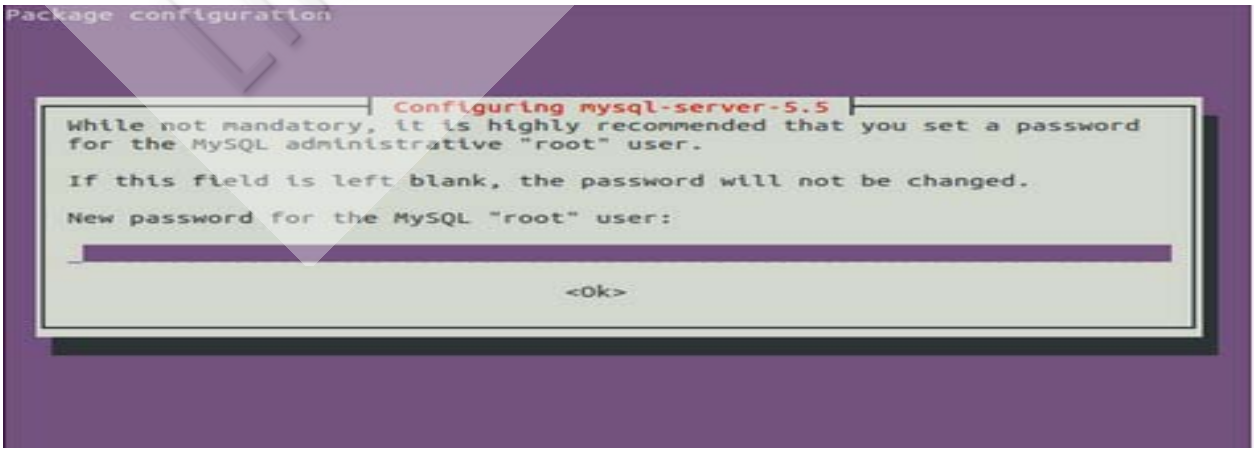
```
sudo apt-get install lamp-server^   ← Note that the ^ is required
```

This will install the following application groups:

- Apache2
- A number of Apache modules
- PHP 5.x
- MySQL Core 5.5 Client
- MySQL Core 5.5 Server
- PHP MySQL Module for Apache
- Supporting Libraries

Secure MySQL Installation

Up to this point, everything has been pretty hands off. However, once the MySQL Server 5.5 installation gets underway, you will be prompted to add a password. This password is intended to secure the MySQL root account (the main administration account that will have access to everything that is in the database itself). The screen you see will look something like this:



Please take the time to pick a secure root password for your database. Realizing that this lab environment is not going to be a permanent site and will be erased once you complete the lab does not make this critical path, but in general practice, picking a complex root password for your server will be one of the primary ways you protect your server and more importantly, the data it contains from unwanted access.

Test the Installation

There are a couple of ways to test locally that your system is listening for incoming HTTP connections right now. The easiest way is to the following test:

```
telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
```

At this point, we can be sure that Apache is listening for incoming HTTP connections over port 80, at least locally, and responding with whatever is defined in our default vhost file (which we will customize shortly).

PHPMyAdmin – MySQL Administration

So we do not get stuck doing all MySQL administration at the command line (and because we are not planning on opening MySQL to the internet in general, only for local connections), let's take advantage of our application stack to install a management interface to our database.

At the command line, execute the following installation command:

```
sudo apt-get install phpmyadmin
```

Again, Ubuntu makes our job a bit easier by having a precompiled installation package available for this valuable tool. If you are running CentOS or RedHat, you can find the package already compiled for you in a third party repository (epel for CentOS for example). If you want to grab the latest version for installation, you can always visit the site (http://www.phpmyadmin.net).

Once the installation is almost complete, phpmyadmin is nice enough to ask you which webserver to install itself into. You will be asked to choose between "lighthttpd" and "apache". For the purposes of this lab, please choose the Apache integration.

You will be prompted for your root MySQL password and then asked to provide a password for PHPMyAdmin to use to register with your database. In this case, pick something easy to remember, in a production environment, be sure it is as secure as your root.

Configure the Virtual Hosts and Modules

So far so good, we just have a few more things to do in order to be ready for our PHPMyAdmin setup. Let's enable the module that allows Apache to serve "user friendly URLs", we will use that in PHPMyAdmin. Drop to a command prompt and do the following:

```
cd /etc/apache2/sites-available
sudo a2enmod rewrite
```

Now we just have to copy our default vhost configuration file, make a couple of quick edits and then activate it. Easy enough, let's execute the following commands:

```
sudo cp 000-default.conf phpmyadmin.conf
rm 000-default.conf
```

Open up our newly copied vhost file for editing using vi and make sure the following changes appear:

```
<VirtualHost *:80>
...          ← Stuff appearing before below can remain unedited

   DirectoryIndex index.php index.html
   DocumentRoot /var/www

...          ← Stuff appearing before below can remain unedited

   <Directory /var/www>
      AllowOverride All
      Options Indexes FollowSymLinks MultiViews
      Require all granted
   </Directory>
</VirtualHost>
```

Now we can activate our vhost and all our changes by enabling it and restarting Apache, complete that by executing:

```
sudo a2dissite 000-default.conf
sudo a2ensite phpmyadmin.conf
sudo server apache2 restart
```

At this point, our vhost and modules are enabled and we can test remote access to PHPMyAdmin and login!

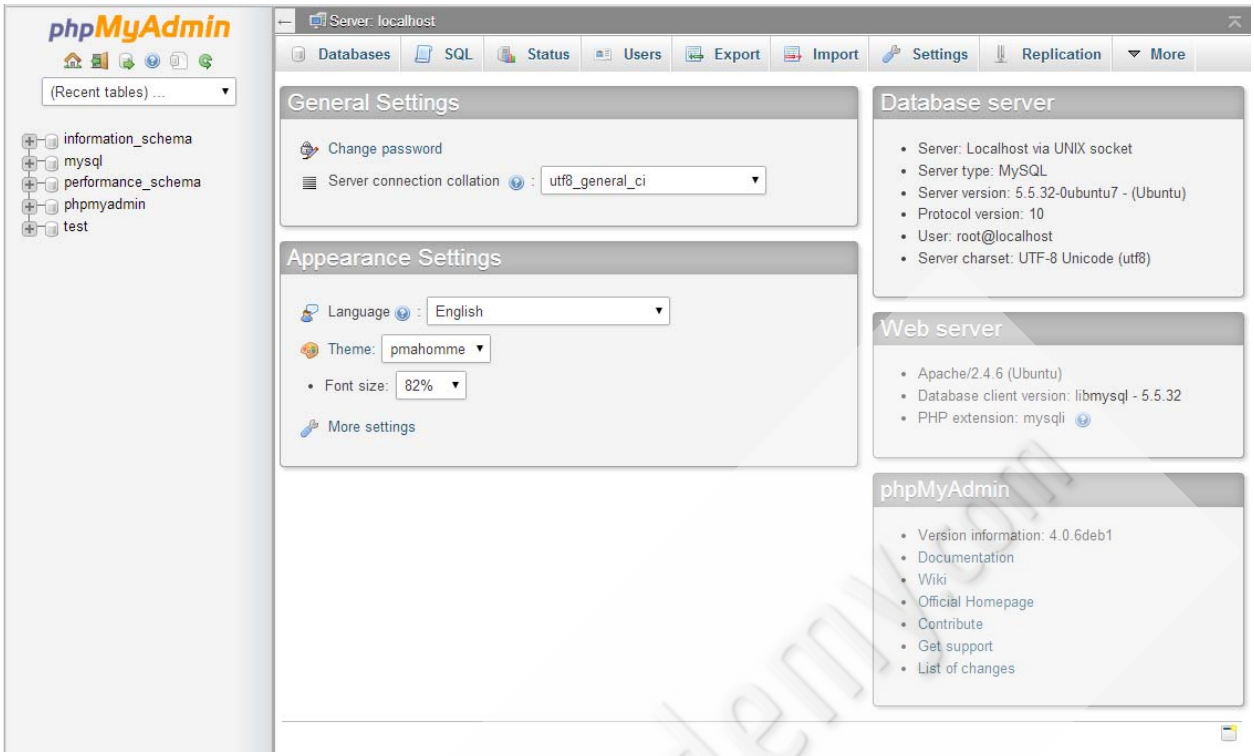
Test the Connection

Finally, all we need to do is open a browser and navigate to the external IP address of the site and confirm it comes up. If you go to just the site IP itself, you will get the default “It Works” Apache site.

We want a little more than that though. We need to verify that PHP, MySQL and Apache are all playing nice together. So we need to connect to PHPMyAdmin and be able to login with the username and password that we created during the MySQL installation.

Open a browser and connect to the URL http://YOURSERVERIP/phpmyadmin and if all works as it is supposed to, you will see:

Let’s login. Remember the ‘root’ account username and password that you just created, let’s enter that and you will then see:



That's it! If you can connect remotely over your browser to PHPMyAdmin and log in, you have configured Apache, MySQL and PHP to work together. In this case, we are using a tool to manage our web based MySQL Database but we can now configure any PHP based site that needs MySQL access for data storage as well – congratulations!

Appendix A: /etc/apache2/phpmyadmin.conf example

```
<VirtualHost *:80>
	# The ServerName directive sets the request scheme, hostname and port
	# that
	# the server uses to identify itself. This is used when creating
	# redirection URLs. In the context of virtual hosts, the ServerName
	# specifies what hostname must appear in the request's Host: header to
	# match this virtual host. For the default virtual host (this file)
	# this
	# value is not decisive as it is used as a last resort host regardless.
	# However, you must set it for any further virtual host explicitly.
	#ServerName www.example.com

	ServerAdmin webmaster@localhost
	DocumentRoot /var/www

	DirectoryIndex index.php index.html

	# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
	# error, crit, alert, emerg.
	# It is also possible to configure the loglevel for particular
	# modules, e.g.
	#LogLevel info ssl:warn

	ErrorLog ${APACHE_LOG_DIR}/error.log
	CustomLog ${APACHE_LOG_DIR}/access.log combined

	# For most configuration files from conf-available/, which are
	# enabled or disabled at a global level, it is possible to
	# include a line for only one particular virtual host. For example the
	# following line enables the CGI configuration for this host only
	# after it has been globally disabled with "a2disconf".
	#Include conf-available/serve-cgi-bin.conf

	<Directory /var/www>
		AllowOverride All
		Options Indexes FollowSymLinks MultiViews
		Require all granted
	</Directory>
</VirtualHost>
```