## Initial server setup (Ubuntu 14.04)

Pitarn Harnpanichpun

email: [pitarnh@gmail.com](mailto:pitarnh@gmail.com)

Step 1: (Root Login)

Connect to your server via putty(windows) or terminal(Linux and Mac)

$ ssh root@YOUR\_SERVER\_IP\_ADDRESS

Step 2: Create a new user

Once you are logged in as root, we're prepared to add the new user account that we will use to log in from now on.

This example creates a new user called "demo", but you should replace it with a user name that you like:

# adduser demo

Step 3: Root privileges

To avoid having to log out of our normal user and log back in as the root account, we can set up what is known as "super user" or root privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word sudo before each command.

To add these privileges to our new user, we need to add the new user to the "sudo" group. By default, on Ubuntu 14.04, users who belong to the "sudo" group are allowed to use the sudo command.

As root, run this command to add your new user to the sudo group (substitute the highlighted word with your new user):

# gpasswd -a demo sudo

For more information: https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-14-04

## Install LAMP (Linux, Apache, MySQL, PHP) (Ubuntu 14.04)

Step 1: Install Apache

The Apache web server is currently the most popular web server in the world, which makes it a great default choice for hosting a website.

We can install Apache easily using Ubuntu's package manager, apt. A package manager allows us to install most software pain-free from a repository maintained by Ubuntu.

For our purposes, we can get started by typing these commands:

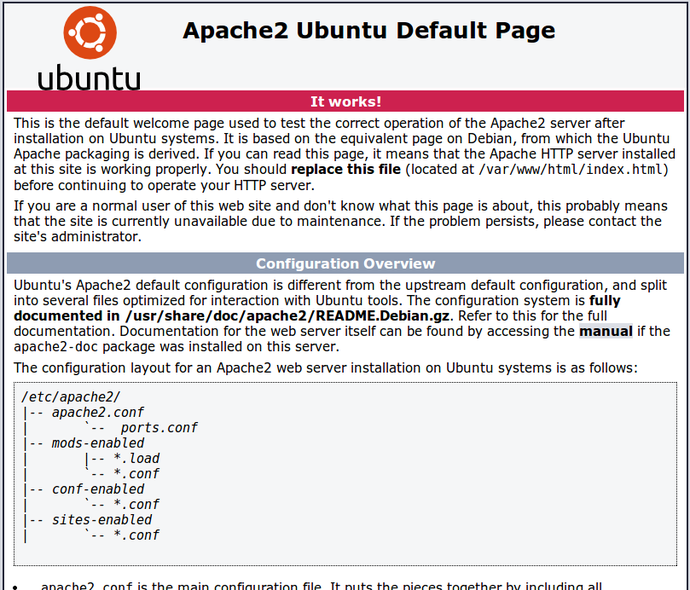
sudo apt-get update

sudo apt-get install apache2

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser:

<http://YOUR_SERVER_IP_ADDRESS>

You will see the default Ubuntu 14.04 Apache web page, which is there for informational and testing purposes. It should look something like this:



Step 2: Install MySQL

Now that we have our web server up and running, it is time to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where our site can store information.

Again, we can use apt to acquire and install our software. This time, we'll also install some other "helper" packages that will assist us in getting our components to communicate with each other:

sudo apt-get install mysql-server php5-mysql

**Note**: In this case, you do not have to run sudo apt-get update prior to the command. This is because we recently ran it in the commands above to install Apache. The package index on our computer should already be up-to-date.

When the installation is complete, we need to run some additional commands to get our MySQL environment set up securely.

First, we need to tell MySQL to create its database directory structure where it will store its information. You can do this by typing:

sudo mysql\_install\_db

Afterwards, we want to run a simple security script that will remove some dangerous defaults and lock down access to our database system a little bit. Start the interactive script by running:

sudo mysql\_secure\_installation

Step 3: install PHP

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display.

We can once again leverage the apt system to install our components. We're going to include some helper packages as well:

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

This should install PHP without any problems. We'll test this in a moment.

In most cases, we'll want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called index.html. We want to tell our web server to prefer PHP files, so we'll make Apache look for an index.php file first.

To do this, type this command to open the dir.conf file in a text editor with root privileges:

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

It will look like this:

<IfModule mod\_dir.c>

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

</IfModule>

We want to move the PHP index file highlighted above to the first position after the DirectoryIndexspecification, like this:

<IfModule mod\_dir.c>

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

</IfModule>

After this, we need to restart the Apache web server in order for our changes to be recognized. You can do this by typing this:

sudo service apache2 restart

Install PHP Modules

To enhance the functionality of PHP, we can optionally install some additional modules.

To see the available options for PHP modules and libraries, you can type this into your system:

apt-cache search php5-

The results are all optional components that you can install. It will give you a short description for each

If you decide you would like to install a package, you can do so by using the apt-get install command like we have been doing for our other software.

Ex: If we decided that php5-cli is something that we need, we could type:

sudo apt-get install php5-cli

At this point, your LAMP stack is installed and configured.

For more information: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-14-04

About the gallery in “your space” menu

1. You should upload the quote images to server and make sure it can access from url (test this by type the url in browser and see that the image is loaded).
2. Create JSON file in the format like glide.json in gallery\_resource folder
3. Upload JSON file and all of images to the server via ftp(you can use filezilla)

For more information: <http://www.androidhive.info/2016/04/android-glide-image-library-building-image-gallery-app/>