At this time, WordPress is the most popular CMS (content management system) on the internet. It allows you to easily set up flexible blogs and websites on top of a MySQL backend with PHP processing. WordPress has seen incredible adoption and is a great choice for getting a website up and running quickly.

In this guide, we’ll focus on getting a WordPress instance set up with an Apache web server on Ubuntu

**PREREQUISITES**

* Before you begin this guide, there are some important steps that you need to complete on your server.
* We will be proceeding through these steps as a non-root user with sudo privileges, so you will need to have one available.
* Additionally, you’ll need to have a LAMP (Linux, Apache, MySQL, and PHP) stack installed on your instance. If you don’t have these components already installed and configured, you can use this guide to learn how to [install Apache](http://www.hyperdroid.tk/index.php/2017/11/08/how-to-host-your-website-using-ubuntu/) on Ubuntu.
* When you are finished with these steps, you can continue with this guide.

**STEP ONE — CREATE A MYSQL DATABASE AND USER FOR WORDPRESS**

The first step that we will take is a preparatory one. WordPress uses a relational database to manage and store site and user information.

We have MySQL installed, which can provide this functionality, but we need to make a database and a user for WordPress to work with.

To get started, log into the MySQL root (administrative) account by issuing this command:

sudo mysql -u root -p

You will be prompted for the password you set for the MySQL root account when you installed the software. You will then be given a MySQL command prompt.

First, we can create a separate database that WordPress can control. You can call this whatever you would like, but I will be calling it **wordpress**because it is descriptive and simple. Enter this command to create the database:

CREATE DATABASE wordpress;

Every MySQL statement must end in a semi-colon (;), so check to make sure this is present if you are running into any issues.

Next, we are going to create a separate MySQL user account that we will use exclusively to operate on our new database. Creating one-function databases and accounts is a good idea from a management and security standpoint.

I am going to call the new account that I’m making **wordpressuser**and will assign it a password of **password**. You should definitely change the password for your installation and can name the user whatever you’d like. This is the command you need to create the user:

CREATE USER wordpressuser@localhost IDENTIFIED BY 'password';

At this point, you have a database and a user account, each made specifically for WordPress. However, these two components have no relationship yet. The user has no access to the database.

Let’s fix that by granting our user account access to our database with this command:

GRANT ALL PRIVILEGES ON wordpress.\* TO wordpressuser@localhost;

Now the user has access to the database. We need to flush the privileges so that the current instance of MySQL knows about the recent privilege changes we’ve made:

FLUSH PRIVILEGES;

We’re all set now. We can exit out of the MySQL prompt by typing:

exit

You should now be back to your regular command prompt.

**STEP TWO — DOWNLOAD WORDPRESS**

Next, we will download the actual WordPress files from the project’s website.

Luckily, the WordPress team always links the most recent stable version of their software to the same URL, so we can get the most up-to-date version of WordPress by typing this:

cd ~ && wget http://wordpress.org/latest.tar.gz

This will download a compressed file that contains the archived directory contents of the WordPress files to our home directory.

We can extract the files to rebuild the WordPress directory we need by typing:

tar xzvf latest.tar.gz

This will create a directory called wordpress in your home directory.

sudo apt-get update

**STEP THREE — CONFIGURE WORDPRESS**

Most of the configuration that we will be doing will be through a web interface later on. However, we do need to do some work from the command line before we can get this up and running.

Begin by moving into the WordPress directory that you just unpacked:

cd ~/wordpress

A sample configuration file that mostly matches the configuration we need is included by default. However, we need to copy it to the default configuration file location to get WordPress to recognize the file. Do that now by typing:

cp wp-config-sample.php wp-config.php

Next, let’s open the configuration file in a text editor:

nano wp-config.php

The only modifications we need to make are to the parameters that hold our database information.

We will need to find the settings for **DB\_NAME, DB\_USER, and DB\_PASSWORD** in order for WordPress to correctly connect and authenticate to the database we created.

Fill in the values of these parameters with the information for the database you created. It should look like this:

// \*\* MySQL settings - You can get this info from your web host \*\* //

/\*\* The name of the database for WordPress \*/

define('DB\_NAME', 'wordpress');

/\*\* MySQL database username \*/

define('DB\_USER', 'wordpressuser');

/\*\* MySQL database password \*/

define('DB\_PASSWORD', 'password');

define('FS\_METHOD', 'direct');

These are the only values that you need to change.

When you are finished, save and close the file.

**STEP FOUR — COPY FILES TO THE DOCUMENT ROOT**

Now that we have our application configured, we need to copy it into Apache’s document root, where it can be served to visitors of our website.

One of the easiest and most reliable way of transferring files from directory to directory is with the **rsync**command. This preserves permissions and has good data integrity features.

The location of the document root in the Ubuntu is **/var/www/html/**. We can transfer our WordPress files there by typing:

sudo rsync -avP ~/wordpress/ /var/www/html/

This will safely copy all of the contents from the directory you unpacked to the document root.

We should now move into the document root to make some final permissions changes

cd /var/www/html

You will need to change the ownership of our files for increased security.

We want to give user ownership to the regular, non-root user (with sudo privileges) that you plan on using to interact with your site. This can be your regular user if you wish, but some may suggest that you create an additional user for this process. It is up to you which you choose.

For this guide, we will use the account name as **demo**(demo@pc-name). This is the account we are performing all of the actions of this guide as.

The group ownership we will give to our web server process, which is **www-data**. This will allow Apache to interact with the content as necessary.

We can quickly assign these ownership values by typing:

sudo chown -R demo:www-data \*

This will set up the ownership properties that we are looking for.

While we are dealing with ownership and permissions, we should also look into assigning correct ownership on our uploads directory. This will allow us to upload images and other content to our site. Currently, the permissions are too restrictive.

First, let’s manually create the **uploads** directory beneath the **wp-content** directory at our document root. This will be the parent directory of our content:

mkdir /var/www/html/wp-content/uploads

mkdir /var/www/html/wp-content/themes

We have a directory now to house uploaded files, however the permissions are still too restrictive. We need to allow the web server itself to write to this directory. We can do this by assigning group ownership of this directory to our web server, like this:

sudo chmod -R 777 /var/www

This will allow the web server to create files and directories under this directory, which will permit us to upload content to the server.

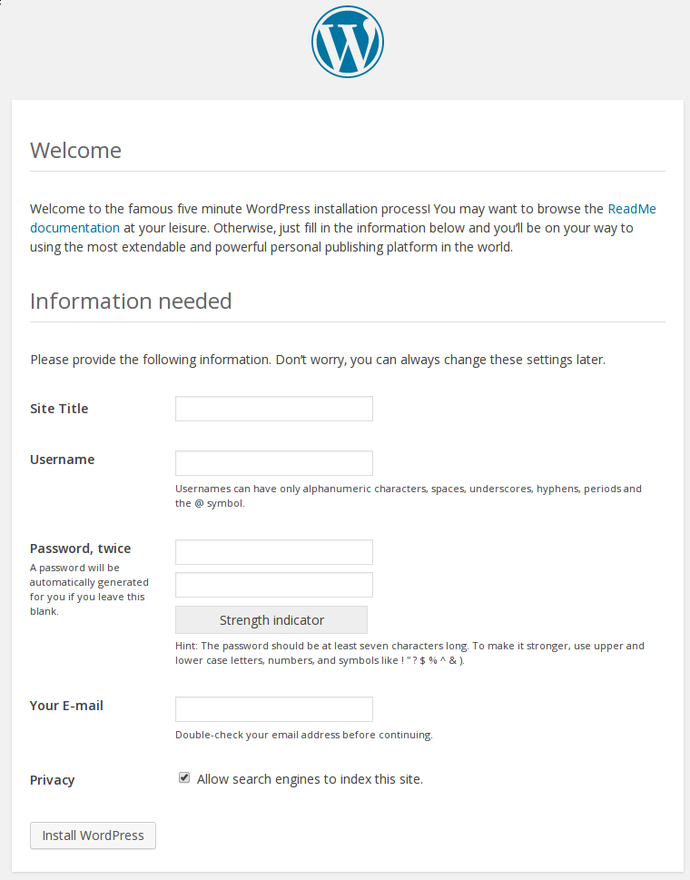
**STEP FIVE — COMPLETE INSTALLATION THROUGH THE WEB INTERFACE**

Now that you have your files in place and your software is configured, you can complete the installation through the web interface.

In your web browser, navigate to your server’s domain name or public IP address:

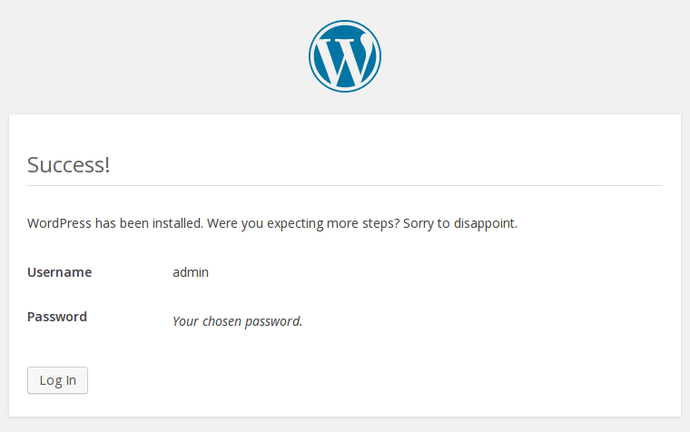
http://server\_domain\_name\_or\_IP

You will see the WordPress initial configuration page, where you will create an initial administrator account:

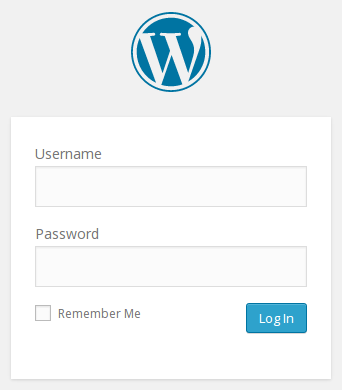


Fill out the information for the site and the administrative account you wish to make. When you are finished, click on the install button at the bottom.

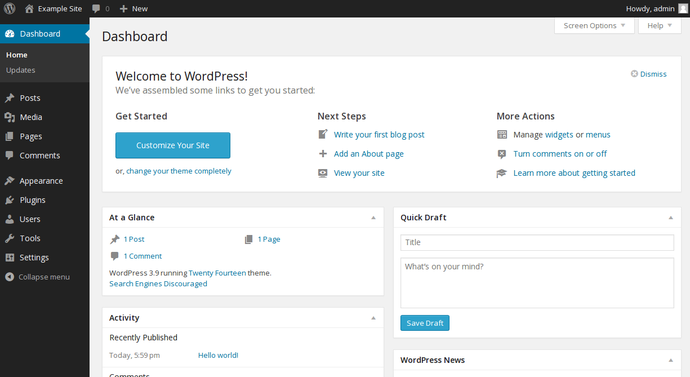
WordPress will confirm the installation, and then ask you to log in with the account you just created:



Hit the button at the bottom and then fill out your account information:



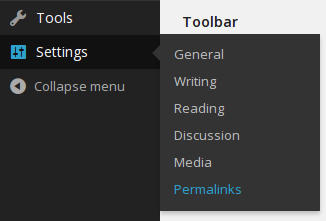
You will be presented with the WordPress interface:



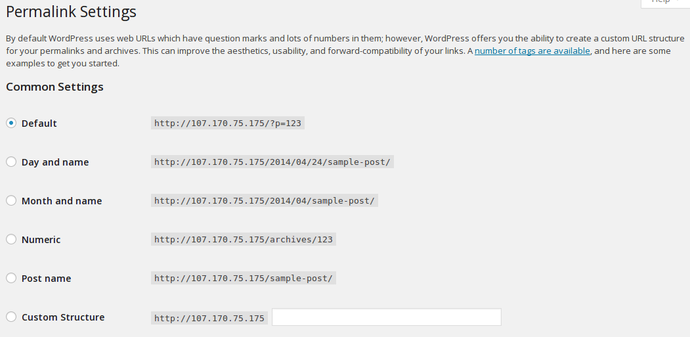
**CHANGING THE PERMALINK SETTINGS IN WORDPRESS**

When you are finished doing the server-side changes, you can easily adjust the permalink settings through the WordPress administration interface.

On the left-hand side, under the **Settings**menu, you can select **Permalinks**:



You can choose any of the preconfigured settings to organize URLs, or you can create your own.



When you have made your selection, click “Save Changes” to generate the rewrite rules. You should see a message like this:

