

LEARNING Ubuntu

A Beginners Guide to Using Linux



NATHAN JAMES NEIL

LEARNING UBUNTU

A BEGINNERS GUIDE TO USING LINUX

NATHAN JAMES NEIL

Copyright © 2016

See More Books by Nathan Neil:

NeilPublishing.com

“In real open source, you have the right to control your own destiny.”

- *Linus Torvalds*

Table of Contents

[Chapter 1](#)

[What is Linux & Ubuntu?](#)

[What Do I Need to Get Started?](#)

[Quick Recap](#)

[Chapter 2](#)

[Setting Up a VPS on DigitalOcean](#)

[Chapter 3](#)

[Getting to the CLI](#)

[Logging In for the First Time](#)

[Navigation](#)

[Quick Recap](#)

[Chapter 4](#)

[Creating a New User](#)

[Quick Recap](#)

[Chapter 5](#)

[System Status at a Glance](#)

[Navigating, Making Directories, and Files](#)

[Creating and Editing Files With Nano](#)

[Editing, Deleting, Moving, and Copying](#)

[Chapter 6](#)

[Updating The Server](#)

[Chapter 7](#)

[Setting Up a Web Server](#)

[Apache2](#)

[The Web Folder](#)

[Installing PHP5](#)

[Installing MySQL](#)

[Installing Postfix](#)

[Installing PHPMYADMIN](#)

[Chapter 8](#)

[Understanding Permissions](#)

[Permission Groups](#)

[Permission Types](#)

[Using Binary References to Set Permissions](#)

[FTP Transfer](#)

[Overview: Ownership of Files and Directories](#)

[Unzipping .Zip Files](#)

[Modifying Apache2 Site Configuration](#)

[Recap on Changing Ownership and Permissions](#)

[Chapter 8](#)

[Creating a Database in MySQL](#)

[Chapter 9](#)

[Bonus Chapter](#)

[Setup WordPress](#)

CHAPTER 1

WHAT IS LINUX & UBUNTU?

You may be familiar with Microsoft Windows and Apple's MacOSX. Both of these are examples of operating systems.

Merriam-Webster defines **operating system** as the main program in a computer that controls the way the computer works and makes it possible for other programs to function.

Essentially, the operating system is the program that lets you interact with a computer, and the computer to understand what you want to do. It goes deeper than that, but I want to keep this topical.

Linux is an open-source operating system. Being open-source is a major difference between Linux and the other popular operating systems. **Open-source** means that the software is provided with the original source code and is made freely available to be redistributed and modified. This is what makes Linux unique. It has a strong community out there modifying the code, participating in forums, and fixing bugs much quicker than the alternatives.

If you are looking to add a new skill to your portfolio, learning how to use Linux, can be very useful. According to ZDNet in 2014, 87% of enterprise businesses added at least one Linux server and 82% planned on adding more in 2015. Linux and its distributions are very popular for database, web, and other types of servers.

A **distro** or **distribution** is a version of Linux that uses the Linux kernel as its foundation. There are hundreds of Linux distros, with most all being free. Ubuntu is a distro of Linux, which is distributed by a company called Canonical. They provide the operating system for free, but charge for commercial support of the product. Nearly a quarter of servers running Linux, are using Ubuntu. This makes it a great choice for your first Linux experience.

WHAT DO I NEED TO GET STARTED?

The tools that you will need, vary on the route that you would like to go. There are four methods, that you could use to experiment with the lessons in this book. One of them, I prefer over the other two options, especially if you are still new to the process. Let's skim over the first three and focus on the forth, which is my preferred method for those of you reading this book.

The first three methods, would require you to manually setup the installation of Ubuntu and configure it. There are a lot of tutorials on YouTube that you can reference if you want to install it your self, but I will caution you that it might not always go the way you anticipated, especially if you are new to the field.

Before we talk about these methods to get started, lets first go over the version of Ubuntu that we will be using for this book. This book, while it can apply to other versions of Ubuntu, is best suited for Ubuntu Server 14.04 LTS. LTS stands for Long Term Support. That means that it has support from Canonical for updates and patches for five years. Many people do not realize that an operating system's support expires.

Once the support expires, there may not be security patches and other upgrades for it available.

In my book, [Securing Business Data: Establishing a Core Value for Data Security](#), we talk about operating system support more in depth. If you work with technology or manage a business, I highly recommend that you read it.

Ubuntu 15.10 is also out now, but it is not an LTS version. I prefer to work with versions that have long term support.

Ubuntu 14.04 LTS is certified for AWS, Microsoft Azure, Joyent, IBM, and the HP Cloud. It is also supported by nearly all VPS providers. VPS stands for Virtual Private Server. A **Virtual Private Server** is a machine sold as a hosting service and is often more affordable than purchasing a server and the bandwidth that it requires. With the popularity of cloud based services, many businesses now run more virtual private servers in order to conserve costs and scale more quickly. With a VPS you pay for what you need to use and can add more resources overtime, if you find you need more processing power or memory.

The one thing that I like about Ubuntu 14.04 LTS is that it has updates to popular packages, many of which we will go over in this book. Ubuntu 14.04 LTS can also be installed on most devices, supporting: x86, x86-64, ARM v7, ARM64, and Power. If you are not sure what those things mean that is fine, but you can be confident in knowing that Ubuntu supports most devices, which is a huge advantage.

For the first three methods to setup Ubuntu on your own, you will need to download the iso file. An **iso** is the file extension used for files that can be burned to a CD or DVD, depending on the size. Essentially it is an image of the disc that you will need to install Ubuntu.

You can download Ubuntu server by going to:

<http://www.ubuntu.com/server>

Once you download the iso file, you will need to burn it to a disc. There are a variety of iso burning tools. In the past, I have used [Ashampoo Burning Studio Free](#), with success.

Method 1: Install Ubuntu on a Spare Machine

If you have a spare computer that you can use to run Ubuntu, that could be the best way to learn how to install and set it up. Since it's a spare machine, there is low risk if you make an error in the installation. What you will need is the spare computer, along with the disc that we burned the Ubuntu image to.

Make sure that the computer is setup to boot from the disc drive and from there you can boot to the installation disc and install Ubuntu. The install is pretty easy, but there is always the possibility of running into a snag. I have found that the Ubuntu forums and YouTube has great resources for troubleshooting common problems.

Once the installation is complete, remove the disc, and the system will boot into Ubuntu.

Method 2: Install Ubuntu using Virtualization Software

If you do not have a spare machine, but have a fair amount of memory on your computer, you can install it using virtualization software. A few examples of this software include the free VirtualBox by Oracle, and the paid version of VMWare.

This method is good for a beginner as it lowers the risk of making an error that may accidentally wipe your machine.

I have used VirtualBox, which is from Oracle and free, but it can be glitchy from time to time. This is an option, but not my preferred choice for beginners reading this book.

To move forward with this and use the free tools, you will need to download and install VirtualBox, which can be found at the following link:

<https://www.virtualbox.org/wiki/Downloads>

You also can configure the virtual machine to boot from the iso file for Ubuntu, rather than burning it to a disc.

Method 3: Dual Boot your Computer

To be honest, I did not want to list this as an option. It is a popular one among those who are skilled and know how to properly set up a machine to run multiple operating systems, but if you are not familiar with this, I would not explore this option.

This one is the most risky, as you could accidentally break the boot cycle of the machine or overwrite your existing operating system if you are not careful.

Preferred Choice for this Book

Method 4: Use a Low Cost VPS

For this book and the nature of it, targeting to educate beginners, I recommend using a VPS. It is more affordable than buying a spare computer if you do not have one, more stable than using virtualization software, and no risk to breaking your primary computer as there is with a dual boot. The main thing that I love about using a VPS is that you can access it from anywhere, since it is hosted in the cloud.

Another advantage is that the VPS provider will install the operating system for you. We will still need to install packages and learn how to use it, but you can setup and deploy an Ubuntu Server very quickly this way.

For creating the tutorials in this book, I will be using DigitalOcean. They have a starter package for \$5 a month, including: 1TB transfer, 512 memory, 1 CPU, and 20 GB of storage. With DigitalOcean, you can get started as quick as 55 seconds.

Another option that I would recommend is Linode. Linode's starter plan has double the resources, but it is also \$10 per month. Since we are just learning and not deploying anything, the starter package from DigitalOcean seems the most practical for our purposes.

There are a lot of other VPS providers out there so feel free to do your research, but I have found these two have the best support and pricing.

With using this option, we do not have to download or burn the Ubuntu iso and we also do not need to do anything special to our computer. Setup is quick and relatively easy. All you need is a credit card and you can cancel at anytime.

If you decide to use methods one, two, or three, your training will start after the next chapter. The next chapter, which is a short one, will walk you through setting up a VPS on DigitalOcean.

QUICK RECAP

- 1) An **operating system** is the main program in a computer, which controls the way the computer works, making it possible for other programs to function.
 1. Microsoft Windows, Mac OSX, and Linux are examples of operating systems.
- 2) **Open-source** means that the software is provided with the original source code and is made freely available to be redistributed and modified.
- 3) A **distro** or **distribution** is a version of Linux that uses the Linux kernel as its foundation.
- 4) A **Virtual Private Server** is a machine sold as a hosting service and is often more affordable than purchasing a server and the bandwidth that it requires
- 5) The recommend method for you to experiment with Ubuntu, is by setting up a VPS.
 1. This books instructions focus on using DigitalOcean as your host, but you can choose other providers as well.

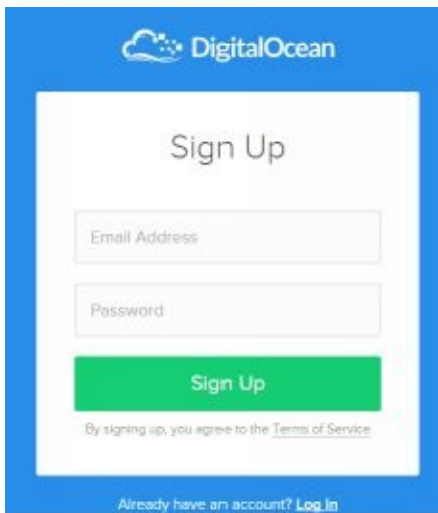
CHAPTER 2

SETTING UP A VPS ON DIGITAL OCEAN

If you have decided to pay the small monthly fee for a VPS, I think you made the right decision. What I like about using a VPS is that it doesn't use my machine's hardware and I can work on it from anywhere that I have an internet connection.

Let's go ahead and get our server setup in a few seconds.

First, go to <https://www.digitalocean.com/> and click on 'Sign Up' in the top right. From there, enter an email address and password that you want to use for your account. Then click 'Sign Up'.

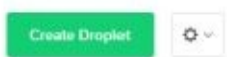
A screenshot of the DigitalOcean 'Sign Up' page. The page has a blue header with the DigitalOcean logo. Below the header is a white box with the title 'Sign Up'. Inside this box are two input fields: 'Email Address' and 'Password'. Below these fields is a green button labeled 'Sign Up'. At the bottom of the white box, there is a small link that says 'By signing up, you agree to the Terms of Service'. Below the white box, on the blue background, is a link that says 'Already have an account? Log In'.

Now we need to confirm our email address. Open your email and find the email from DigitalOcean. It will be titled "DigitalOcean – Please confirm your email address." Open the email and proceed to click on the verification link.

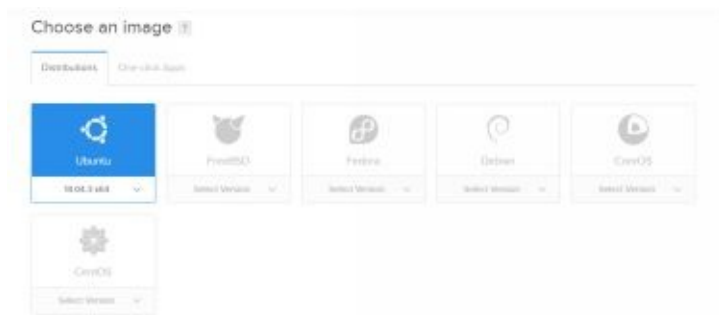
Great, now that our email is confirmed, we need to select a payment option. DigitalOcean gives you the option of using a credit card or PayPal. For me, I use PayPal for convenience.

If you use PayPal, you have to prepay \$5, which is credited to your account. This will cover our first month!

Once you have added your payment method, we can create our VPS, what DigitalOcean calls a Droplet. In the top right, click on the green 'Create Droplet' button.



Next, it will ask you to select a distribution. We want to select Ubuntu 14.04, which is the default.



Once you have selected Ubuntu, scroll down to select the size of server you want to deploy. They have the \$20 per month plan selected as the default. Just click on the \$5 per month plan, if that is the size you want, and then scroll down to select a datacenter region.

I recommend selecting a region based on your location. Since I am in the USA on the east coast, New York is my pick.

Under the additional options, it is up to you if you would like a backup. If we were doing development or anything outside of training and testing, I would recommend selecting the backup option, but since we just want the basics for now to learn, I am going to leave these options unchecked.

Select additional options ?

☐ Private Networking ☐ Backups ☐ IPv6 ☐ User Data

From there, scroll down to the bottom and click the big green ‘Create’ button.

Finalize and create

How many Droplets? Deploy multiple Droplets with the same configuration. <div>— 1 Droplet +</div>	Choose a hostname Give your Droplets an identifying name you will remember them by. Your Droplet name can only contain alphanumeric characters, dashes, and periods. <div>ubuntu-512mb-nyc-2-01</div>
--	--

Create

Then in a matter of a few seconds, your system is setup and ready. DigitalOcean will then email you the IP address, username, and password for your server.

CHAPTER 3

GETTING TO THE CLI

CLI stands for command line interface, where you can interact with the system by entering lines of text. This is how we interact with our server.

If you decided not to use the recommended method of setting up a VPS on DigitalOcean or another provider, then you can see the CLI on your computers display. When you boot your computer, it will bring you to the CLI, where you can then login.

If you are using a VPS, then you will need a way to connect and view the CLI. We can connect to the VPS by using a protocol called SSH. **SSH** stands for Secure Shell, which is an encrypted way that we can remote login to our server.

When we signed up for DigitalOcean, they sent us an email with our IP address, username, and password. If you used a similar service, it would also provide you with this information or allow you to set it manually. While you are likely familiar with user names and passwords, you may not be familiar with IP addresses. An **IP Address**, is a string of numbers that identifies a computer on a network. For our VPS, our IP Address is a public one, which can be accessed from anywhere with an internet connection.

The program I recommend to SSH into your server is called PuTTY. **PuTTY** is a free SSH client that is open source and supported on Windows. You can download it from the following link:

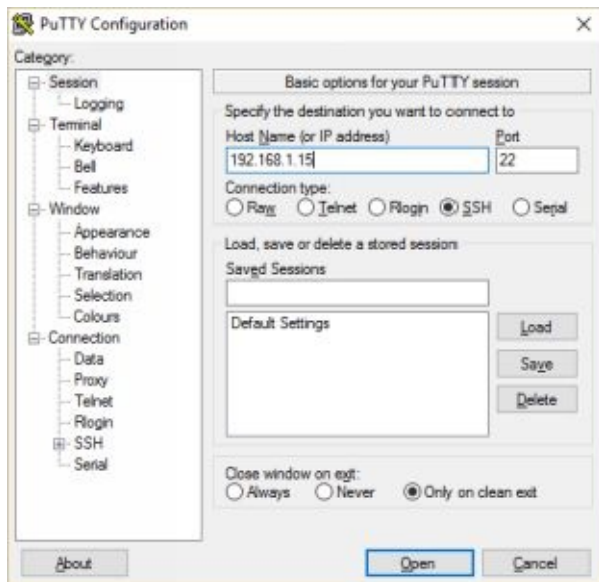
<http://putty.org>

The file you download is an executable (.exe) that does not require an install. Once you download it, you can drag it onto your desktop for quick access.

Connecting to your VPS with this tool is very easy. Under Host Name or IP address, put the IP address that was provided. Select SSH as the connection type and then click open. See figure 1 as an example of what it should look like.

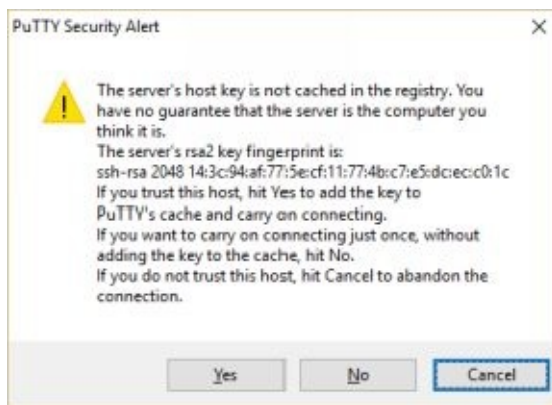
Note: Your IP Address will be different than the one in the figure.

Figure 1: Putty Configuration



When you hit connect, you may get a message like the one in Figure 2. Click Yes.

Figure 2: PuTTY Security Alert



LOGGING IN FOR THE FIRST TIME

Now you will be prompted in for the first time. Enter the username root, then the password that was provided.

If you used DigitalOcean, their system requires a password reset the first time you log in. You will need to reenter the provided password and then the new one you wish to set twice. Make sure you follow secure password guidelines and set a secure password for the root user.

Note: *Throughout this book, I use tables to explain the different commands that we are going to learn. The first row is the task that we want to do, the second is the command we need to use, and the third is an example of what your CLI will look like. I think this is the best way to depict and explain the commands we are going to learn.*

If you were not prompted to change your password, but want to, you can enter the command, `passwd`. See the table on the next page for more details.

Change My Password

Command passwd

Result System will prompt for new
password

CLI Display root@ubuntu:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated
successfully

NAVIGATION

Do not be intimidated by the white text and black background on the CLI. I know that there is no mouse or graphics, but after a short while, you will be comfortable navigating through the directories on your server. **Directory** is like a folder that you may have used on a Windows or Mac device. As a person who has used this for a long time, I actually prefer this to having a graphical user interface. Before we do anything else, let's learn to explore the server. By using the *pwd* command, we can see the location of the directory that we are currently viewing.

Find Your Current Directory Location

Command	<code>pwd</code>
Result	System will output the directory that you are in.
CLI Display	<code>root@ubuntu:~# pwd</code> <code>/root</code>

Right now we are in a directory named `root`, which is essentially the home directory of the user `root`. **There is big difference between this root home directory and the actual root of the system.**

The path `/` is the root of the system, but right now we are in a subdirectory that is named `root (/root)`.

The **root** of the server is the top directory. Everything is a subdirectory inside of it. From our current location, there are two ways we can go there. Let's start with the easiest command that will always get us to root.

Go To The Root Directory (/)

Command	<code>cd /</code>
Result	We change to the main root directory
CLI Display	<code>root@ubuntu:~# cd /</code> <code>root@ubuntu:/#</code>

Now let's use, *pwd* again to see where we are at.

Find Your Current Directory Location

Command	<code>pwd</code>
Result	System will output the directory that you are in.
CLI Display	<code>root@ubuntu:~# pwd</code> <code>/</code>

If you want to see what is in the folder, we can use the `ls` command.

List Files

Command	<code>ls</code>
Result	Lists the files in the current directory
CLI Display	<code>root@ubuntu:/# ls</code> <code>bin home lib64 opt sbin usr</code> <code>boot initrd.img lost+found proc</code> <code>srv var</code> <code>dev media root sys vmlinuz</code> <code>etc lib mnt run tmp</code>

We can also go into more detail by adding a flag after `ls`.

A **flag** is typed at the prompt as part of our command and adds additional parameters to tell the system what we want.

Try running this command: `ls -la`

In that example, `-la` is the flag. The `l` tells the system that we want to see more details and `a` lists all the files in the directory, even the hidden ones.

Now, let's change to the directory named home.

Change Directory

Command	<code>cd home</code>
Result	System moves us to child folder

home

CLI Display root@ubuntu:/# cd home

root@ubuntu:/home#

Note: It is important to understand the difference between parent and child directories. If we have a directory called documents and another inside of it called pictures. Pictures would be a child directory of documents. In this instance we were in the parent directory / and changed it to the child directory /home. If you were not in the parent folder, you could still navigate to home, but after cd, you would have to list the full path (example: cd /home) instead. If you get an error such as, “No such file or directory”, you may need to enter the full path, since you may not be in the right parent directory.

If we want to go back to the root directory we can use another change directory command, which makes going up a directory very easy.

Move Up A Directory Level

Command	<code>cd ..</code>
Result	System moves us to the parent folder
CLI Display	<code>root@ubuntu:/home# cd ..</code> <code>root@ubuntu:/#</code>

Using *cd* along with the directory names and `..` can allow us to quickly move throughout our server.

QUICK RECAP

- 1) **CLI** stands for command line interface, where you can interact with the system by entering lines of text.
- 2) **SSH** stands for Secure Shell, which is an encrypted way that we can remote login to our server.
- 3) An **IP Address**, is a string of numbers that identifies a computer on a network.
- 4) **PuTTY** is a free SSH client that is open source and supported on Windows.
- 5) The *passwd* command, allows us to change the current users password.
- 6) Entering *pwd*, will provide us with our current location in the server's directories.
- 7) Using *ls*, will list the files and directories in our current directory.
- 8) Adding a flag, such as *-la*, after *ls*, provides us with more details and shows hidden files.
- 9) We can use *cd*, to change directories quickly
 1. We can move up a folder with *cd ..*
 2. Putting a child directory after *cd*, will put us in the directory.
 1. Example: *cd home*
 3. We can always find root by using *cd /*
- 10) The path */* is the root of the system, but */root* is a child or sub directory named root.

CHAPTER 4

CREATING A NEW USER

The user ‘root’ has full access to the system. While we used it to learn some basic navigation, we should not use it as our main username for the system. It defeats the security model that is great and makes Linux systems so secure. By not using root, it also prevents accidental typo errors that could result in disasters. It’s just good practice. What we will do in our next command is create a new user, named milo, and then we will provide him with sudo access.

Sudo before a command, will allow our user, milo, to be able to elevate it to higher security privileges to install software and make changes. Sudo runs the command as a super user, even though the user milo would normally only have more limited access. This will come together soon, when we dive into using it, but first we need to create milo’s account.

Add a New User

Command	<code>adduser milo</code>
Result	System sets up user milo
CLI Display	<code>root@ubuntu:/# adduser milo</code> Adding user ‘milo’ ...

Once you run that command, the system will then have you setup the account. You will be prompted for a password first. Then it will prompt you for user information, such as Full Name that you can skip by pressing ENTER for default settings. If this was a production system, adding the name, room number, work phone, home phone, and other information could be useful, but for right now there is no reason not to skip through it. Once you have reached the end, you will need to enter Y and then press enter.

You can see a screenshot of my process adding milo below, if you need some added help.

```
root@ubuntu-512mb-nyc2-01: /  
root@ubuntu-512mb-nyc2-01:/# adduser milo  
Adding user 'milo' ...  
Adding new group 'milo' (1000) ...  
Adding new user 'milo' (1000) with group 'milo' ...  
Creating home directory '/home/milo' ...  
Copying files from '/etc/skel' ...  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
Changing the user information for milo  
Enter the new value, or press ENTER for the default  
Full Name []:  
Room Number []:  
Work Phone []:  
Home Phone []:  
Other []:  
Is the information correct? [Y/n] Y  
root@ubuntu-512mb-nyc2-01:/#
```

Earlier, we learned how to change our password. If you are root or a sudo user you can change a user's password as well by following the steps below. Since we just set milo's password you may not want to change it, but these are the steps you would take.

Change a User's Password

Command	<code>passwd milo</code>
Result	System will prompt for you to set a new password for user milo.
CLI Display	<code>root@ubuntu:/# passwd milo</code> Enter new UNIX password: Retype new UNIX password: <code>passwd: password updated successfully</code>

Ok, so now that we have milo setup, we still need to grant him access to sudo. We would only do this for users, who may need to increase their security privileges.

First we need to open the sudoer's configuration file to modify it. We can do this by using the visudo command.

Giving a User Sudo Privileges

Command	<code>visudo</code>
Result	System will open the sudoer's file.
CLI Display	<code>root@ubuntu:/# visudo</code> # This file MUST be edited with the 'visudo' command as root.

```
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# Directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults
secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL

...
```

There is a lot of text in this file, but do not be intimidated. In the table above I highlighted: **root ALL=(ALL:ALL) ALL**.

Below that selection we need to add the following:

```
milos ALL=(ALL:ALL) ALL
```

Press the down key until you are in the blank space under root in **User privilege specification**. Once there type *milos*, press tab, type *ALL=(ALL:ALL)*, then press space, and type *ALL*. Now press the control key and X. The system will ask you if you wish to save modified buffer. Press Y and press enter.

Now you can breathe. I know we dived in deep a little fast, but it is important that we start using a user id other than root. It is also important that at the same time we can install software and learn more, so this was needed.

Before we move on, we can learn two quick commands. Right now there is probably a lot

of text on your CLI. If you want to clean that up, we can use the clear command.

Clearing the Screen

Command	clear
Result	System will clear the screen
CLI Display	root@ubuntu:/# clear

Now that we have a clear screen we can close out of PuTTY, before continuing on our new user id, milo.

Logging Out

Command	exit
Result	System will close SSH session and exit PuTTY
CLI Display	root@ubuntu:/# exit

QUICK RECAP

- 1) We should avoid using the 'root' user as a good usage practice.
- 2) **sudo** before a command, allows a user to have super user privileges.
- 3) We can add a user by using the command *adduser*, with the username afterwards.

1. Example: *adduser nathan*

- 4) We can change a user's password by using the *passwd* command before their name.

1. Example: *passwd nathan*

- 5) To have access to use **sudo**, a user must first be added to the configuration file, which can be edited by using the *visudo* command.
- 6) If our CLI gets too cluttered, we can clear it by using the *clear* command.
- 7) To end our session and sign out of PuTTY we can use *exit*.

CHAPTER 5

SYSTEM STATUS AT A GLANCE

Log into your server, with the new user that we created in the last chapter. From here, through the rest of the book, we will not be using the root user id.

When you first log in, you see some useful information, which will give you an idea of how much system resources you are using.

On the left you will see the following:

- System Load – Average usage of computational work the system has performed over a period of time.
- Usage of / - Total amount of storage used
- Memory Usage – Percentage of memory being utilized.
- Swap usage – Amount of swap memory being used.
 - This is virtual memory from your hard disk.

On the right we can see the following:

- Processes – Number of system processes running.
- Users logged in – Number of users on the server
- IP Address for eth0 – In this case it is our public IP

As those numbers increase, your systems performance may start to lag. A good systems engineer monitors these statuses and scales more resources before that occurs.

NAVIGATING, MAKING DIRECTORIES, AND FILES

Some of this is going to recap earlier chapters, but this repetition is important to remembering the commands that you will be using frequently in daily usage.

Now that we have logged in, use the *pwd* command to see our current directory.

Find Your Current Directory Location

Command	<code>pwd</code>
Result	System will output the directory that you are in.
CLI	<code>milo@ubuntu:~\$ pwd</code>
Display	<code>/home/milo</code>

Notice that our directory this time is different than it was when we logged in with the root user. Last time we were at `/root`, which was the home directory for root. This time we are at `/home/milo`, which is the home directory for the user milo.

Tip: If you ever need to get back to your home directory use *cd*, to get back to your users home folder.

Now that we are in our home folder, let's make a new directory called 'documents'.

Making a New Directory

Command	<code>mkdir documents</code>
Result	System will create a child directory called documents
CLI	<code>milo@ubuntu:~\$ mkdir</code>
Display	<code>documents</code> <code>milo@ubuntu:~\$</code>

Use the *ls* command to list the contents of our home directory and see if the directory 'documents' is there.

List Files

Command	<code>ls</code>
---------	-----------------

Result	Lists the files in the current directory
CLI Display	milo@ubuntu:~\$ ls documents

From using *ls*, we now see that our directory is empty outside of the directory 'documents'.

Let's use what we have learned from the previous chapters to practice our navigation skills and go into the documents directory.

Try using the command: *cd Documents*

You may notice that it returns, "cd: Documents: No such file or directory." Linux is case sensitive, which means that 'Documents' and 'documents' are two separate things. You will want to keep that in mind as you use Ubuntu, since using the correct case matters.

Now let's do it the right way.

Change to a Child Directory	
Command	cd documents
Result	Changes to the documents directory
CLI Display	milo@ubuntu:~\$ cd documents milo@ubuntu:~/documents\$

Use the *ls* command to list the files in the directory, you'll notice that it is empty.

Now let's go back home by using, *cd*.

Change to User Home Directory	
Command	cd
Result	Changes to user's home directory
CLI Display	milo@ubuntu:~/documents\$ cd milo@ubuntu:~\$

If we use *pwd* we will find that we are back at, /home/milo.

Now, use *cd* with the full path to the documents directory. Remember, that Linux is case sensitive.

Change Directory Using a Full Path

Command `cd /home/milo/documents`

Result Changes to specified directory

CLI `milo@ubuntu:~ $ cd`
`/home/milo/documents`
`milo@ubuntu:~/ documents$`

If we wanted to get the parent directory of this folder we can also use: *cd ..*

Move Up A Directory Level

Command `cd ..`

Result System moves us to the parent folder

CLI Display `milo@ubuntu:~/ documents$ cd ..`
`milo@ubuntu:~$`

You should be getting the hang of it now!

Experiment some more and in the next section we will create some files using the built in text editor, nano.

CREATING AND EDITING FILES WITH NANO

Nano is a useful text editor for the command line interface (CLI) in Linux systems. Since we are using the CLI, nano is keyboard oriented with control keys, rather than using a mouse. By learning some of the control keys, you will be able to quickly edit files from the nano editor. Once you pick this up and learn more about using Linux by the command line you can then move onto more advanced editors like vi and, which are great, but often have a steeper learning curve.

Nano is easy to use and great for beginners. It is often installed by default in Ubuntu and other distros and works very well with sudo from the command line. It's a WYSIWYG editor; "what you see is what you get."

If you built your Ubuntu server with DigitalOcean or another VPS provider it is already installed. In some circumstances if you installed Ubuntu manually, it may not be installed.

We will talk about installing packages to Ubuntu more in a later chapter, but in case you do not have nano, you can install it by using the command: `sudo apt-get install nano`

Again we will discuss installing packages in more detail later, but that command, if you don't already have nano, will install it to your server.

Let's go ahead and create a file using nano.

Creating a file in nano

Command	<code>nano notes.txt</code>
Result	System launches nano for new file notes.txt
CLI Display	<code>milo@ubuntu:~\$ nano notes.txt</code>

Now, you will notice that we are in a simple text editor. Since we do not have a mouse we can get a list of the control keys by pressing control and G at the same time.

See a complete list of nano command keys in the table below:

Nano command keys

^ represents the control 'Ctrl' key.		
^G	(F1)	Display this help text
^X	(F2)	Exit from nano
^O	(F3)	Write the current file to disk
^J	(F4)	Justify the current paragraph

^R (F5) Insert another file into the current one

^W (F6) Search for a string or a regular expression

^Y (F7) Go to previous screen

^V (F8) Go to next screen

Nano command keys (cont.)

^K (F9) Cut the current line and store it in the cutbuffer

^U (F10) Uncut from into the current line

^C (F11) Display the position of the cursor

^T (F12) Invoke the spell checker, if available

M-\ (M-|) Go to the first line of the file

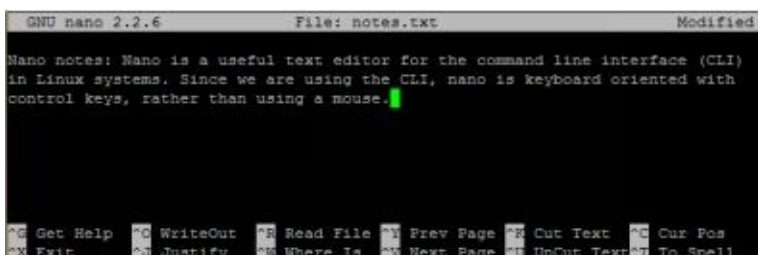
M-/ (M-?) Go to the last line of the file

^Y Prev Page **^P Prev Line** **^X Exit**

^V Next Page **^N Next Line**

In the nano editor, enter the following text:

Nano notes: Nano is a useful text editor for the command line interface (CLI) in Linux systems. Since we are using the CLI, nano is keyboard oriented with control keys, rather than using a mouse.

A screenshot of the GNU nano 2.2.6 text editor. The title bar shows "GNU nano 2.2.6", "File: notes.txt", and "Modified". The main text area contains the text: "Nano notes: Nano is a useful text editor for the command line interface (CLI) in Linux systems. Since we are using the CLI, nano is keyboard oriented with control keys, rather than using a mouse." followed by a green cursor. The bottom status bar shows various keyboard shortcuts: ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Page, ^C Cut Text, ^X Exit, ^J Justify, ^W Where Is, ^N Next Page, ^U UnCut Text, ^S To Spell.

Once you have the text entered, use the exit command keys:

Control + X

You then will be prompted if you want to “Save modified buffer?” Press Y then Enter to

save the file.

Now let's see what is in our folder by using the *ls* command.

List Files

Command	ls
---------	----

Result	Lists the files in the current directory
--------	--

CLI Display	milo@ubuntu:~\$ ls documents notes.txt
-------------	---

EDITING, DELETING, MOVING, AND COPYING

With our notes.txt file created, we can now learn how to move, edit, delete, and copy files.

To get started, let's copy the notes.txt file to a new file called test.txt. To do this we will use the *cp* command, which will allow us to copy one file to another. To use *cp*, we need to also provide the file we want to copy, and also the file we want to copy into. See the table below:

Copy to New File	
Command	<code>cp notes.txt test.txt</code>
Result	System copies notes.txt and creates a new file, test.txt with the contents.
CLI Display	<code>milo@ubuntu:~\$ cp notes.txt test.txt</code> <code>milo@ubuntu:~\$</code>

By using the *ls* command, we can check to see if the file was copied.

List Files	
Command	<code>ls</code>
Result	Lists the files in the current directory
CLI Display	<code>milo@ubuntu:~\$ ls</code> <code>documents notes.txt test.txt</code>

Now that we have two files, we should open test.txt in nano and make a few changes.

Open a File in Nano	
Command	<code>nano test.txt</code>
Result	System will open test.txt in nano.
CLI Display	<code>milo@ubuntu:~\$ nano test.txt</code>

When you open the file, you will notice that it has the same text as our notes.txt file. To the beginning of this document, let's add a line: *This document is a test for learning Linux*. Once that is added. Save the file as we did earlier.

With the *cp* command, the file is copied. When we move a file, it is like using cut in Windows as the original is deleted.

We will now move test.txt into the documents directory. By using the *mv* command we can move files. After *mv*, we put the name of the file we want to move and then the location we want to move it to.

Move a File

Command	<code>mv test.txt documents/test.txt</code>
Result	System will open test.txt in nano.
CLI Display	<code>milo@ubuntu:~\$ mv test.txt documents/test.txt milo@ubuntu:~\$</code>

If you change your directory to documents and use *ls* you will find the test.txt file is now located there.

Change Directory to Documents and List Files

Command	<code>cd documents; ls</code>
Result	System will change to documents and list files.
CLI Display	<code>milo@ubuntu:~\$ cd documents; ls milo@ubuntu:~/documents\$ test.txt</code>

You may notice above that I used a semi-colon after *cd documents*. This allows us to combine two commands. In this example the semi-colon allowed us to change directory and then list the files in the new directory.

Now we can also use *mv* to rename a file. Using the command below, we can rename the file and list the files in the directory.

Using mv to Change File Name

Command	<code>mv test.txt newtest.txt; ls</code>
Result	System will change the file name to newtest.txt and list the files in the directory.
CLI Display	<code>milo@ubuntu:~/documents\$ mv test.txt newtest.txt; ls newtest.txt</code>

Hopefully you are not too attached to this file as in the next table, we will learn how to delete a file, using *rm*.

Using rm to Delete a File

Command	<code>rm newtest.txt</code>
Result	System will delete newtest.txt
CLI Display	<code>milo@ubuntu:~/documents\$ rm newtest.txt milo@ubuntu:~/documents\$</code>

If want you can use *ls*, but you will find that the directory is empty.

Now let's go back to our parent directory.

Move to Parent Directory

Command	<code>cd ..</code>
Result	System will move up to parent directory
CLI	<code>milo@ubuntu:~/documents\$ cd ..</code>
Display	<code>milo@ubuntu:~ \$</code>

When deleting files or directories, be very careful as one misstep could delete the entire system. Avoiding using the root user helps with this.

Sometimes you may want to delete a directory, but to delete a directory we must add a flag.

If we use *rm documents*, we will get an error that it is a directory.

To delete a directory, we must use the command in the table below.

Delete a Directory

Command	<code>rm -r documents</code>
Result	System will delete the directory named documents.
CLI	<code>milo@ubuntu:~\$ rm -r documents</code>
Display	<code>milo@ubuntu:~\$</code>

To delete a directory, we have to use the *-r* flag.

CHAPTER 6

UPDATING THE SERVER

In the last few chapters, we got a foundation in navigation and managing files. Another thing that is vital to a strong foundation is knowing how to update your server.

As we install updates and packages, we will be using a utility called **apt-get**. Apt-get is a powerful package management program. APT stands for advanced packaging tool and is how we will update, add, and remove packages.

For us to update the server, we will need to give milo super user privileges. In the first apt-get command below, we will update the list of packages and get information on the newest versions.

Update Package Information

Command	<code>sudo apt-get update</code>
Result	System will increase the user's rights, search for an updated list of packages, and look for newer versions.
CLI Display	<code>milo@ubuntu:~\$ sudo apt-get update</code> <code>[sudo] password for milo:</code>

After you run that, your CLI will list a long list of web addresses as it checks the various package lists for updates. When it is done it will state, "Reading package lists... Done."

With the new information of updates that are available, we can now upgrade our server to have the latest versions of the current packages. Using the command below will fetch the new versions and install them onto the server.

Upgrade Server Packages

Command	<code>sudo apt-get upgrade</code>
Result	System will increase the user's rights, search for an updated list of packages, and look for newer versions.
CLI	<code>milo@ubuntu:~\$ sudo apt-get</code>

```
Display      upgrade
[sudo] password for milo:
Do you want to continue? [Y/n] y
milo@ubuntu:~$
```

After entering, Y to continue, the server will setup and install all of the newest versions of packages to our server.

Keeping your server with the latest updates installed is a vital foundation to a core level of security. There is much more you will need to learn to secure your system, but this is a basic step to get started.

CHAPTER 7

SETTING UP A WEB SERVER

APACHE2

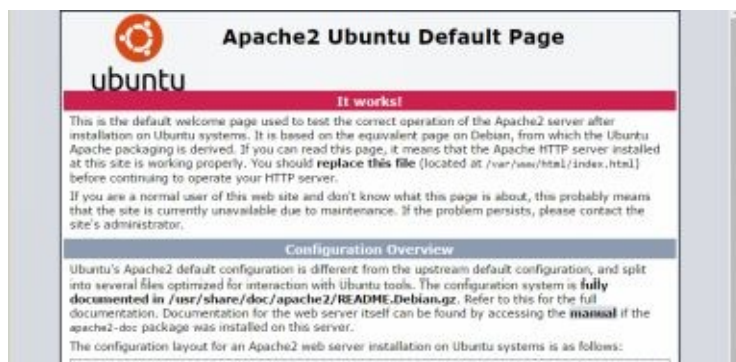
A common configuration for Linux servers is to set them up to be web servers. 80% of servers that host websites are running Linux. To establish a good foundation, learning how to install these features are important.

To get started in setting this server up as a web server, we need to install Apache2. Apache2 is the most commonly used web server on Linux systems. Web servers are used to serve up web pages requested by client computers. To run website software like WordPress we will need more packages than this, but installing Apache2 is the best starting place.

Installing Apache2

Command	<code>sudo apt-get install apache2</code>
Result	System will install apache2 and dependent packages to the server.
CLI Display	<pre>milo@ubuntu:~\$ sudo apt-get install apache2 [sudo] password for milo: Do you want to continue? [Y/n] y milo@ubuntu:~\$</pre>

Now if we open our web browser and type in our IP address, we will see the Apache2 landing page.



Stopping and Restarting Apache2

There may be an instance when you want to stop or restart the Apache2 webserver.

You can do so by using the following commands:

Stop Apache2: `sudo service apache2 stop`

Start Apache2: `sudo service apache2 start`

Restart Apache2: `sudo service apache2 restart`

THE WEB FOLDER

You may be wondering, where are my website files located? On most Ubuntu installs you can get to the web directory by changing directory to `/var/www/html`.

Go To The HTML Directory

Command `cd /var/www/html; ls`

Result System will change to the html directory and list the files.

CLI milo@ubuntu:~\$ cd /var/www/html
Display index.html

If you want to, you can open the default `index.html` file in nano.

Open index.html in nano

Command `nano index.html`

Result System will load `index.html` into nano.

Note: To save and make changes, you may need to open the file using `sudo`.

Now let's change back to our home directory by using `cd`.

INSTALLING PHP5

PHP is a popular web server-side scripting language. It is very popular and used by most content management systems, such as WordPress.

Installing PHP5

Command `sudo apt-get install php5`

Result System will install php5 and

dependencies.

CLI	milo@ubuntu:~\$ sudo apt-get
Display	install php5
	[sudo] password for milo:
	Do you want to continue? [Y/n] y
	milo@ubuntu:~\$

INSTALLING MySQL

Along with Apache2 and PHP, your server will need MySQL for many web applications that need to store data. This includes the widely popular WordPress CMS.

Installing MySQL Server

Command `sudo apt-get install mysql-server`

Result System will install MySQL Server.

Once you run this command, you will need to enter your password in addition to agreeing to continue as we did before. After that is done the install will prompt you for a password for the 'root' user. It is highly recommended you use a different password for MySQL.

Enter your password, then tab to ok and press enter. After that you will need to reenter the password again.

Now, we need to install another package to allow PHP to talk with MySQL.

Installing PHP5 MySQL Dependency

Command `sudo apt-get install php5-mysql`

Result System will install php5 dependency for MySQL.

CLI Display
`milo@ubuntu:~$ sudo apt-get install php5-mysql`
`[sudo] password for milo:`
`Do you want to continue? [Y/n] y`
`milo@ubuntu:~$`

Next we need to install a package to allow apache2 to talk and authenticate to MySQL.

To do so, use the following command:

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

In some circumstances you may use contact forms on your site for people to reach you. In order to send mail from the server we need to install another package: postfix.

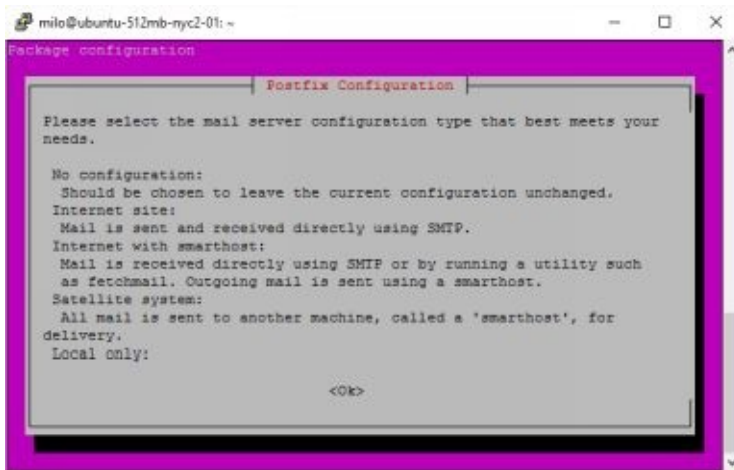
INSTALLING POSTFIX

Postfix is a mail server that started out of IBM research labs as an alternative to Sendmail. Postfix aims to be fast, easy to use (great for beginners), and secure. This is my recommendation if you want to be able to have mail services running on your server. The installation is easy.

Installing Postfix

Command	<code>sudo apt-get install postfix</code>
Result	System will install phpmyadmin.
CLI Display	<code>milo@ubuntu:~\$ sudo apt-get install postfix</code> [sudo] password for milo: Do you want to continue? [Y/n] y

After you approve the install, you will see the window below:



Tap to hit enter on the OK button and then select Internet Site in the next prompt. Once that is done press tab to ok and then enter. The next step asks for your fully qualified domain (FQDN). If you do not have one yet just tab to ok and use the default.

Now it is installed! We won't cover the other various configuration settings in this book, but you can find various helpful resources online.

INSTALLING PHPMYADMIN

A great way to test to see if we configured all of this properly, is by installing phpmyadmin. Phpmyadmin is a great tool to provide a web interface for managing your MySQL server, its tables and databases.

Installing phpmyadmin

Command `sudo apt-get install phpmyadmin`

Result System will install phpmyadmin.

CLI Display `milo@ubuntu:~$ sudo apt-get
install phpmyadmin`

`[sudo] password for milo:`

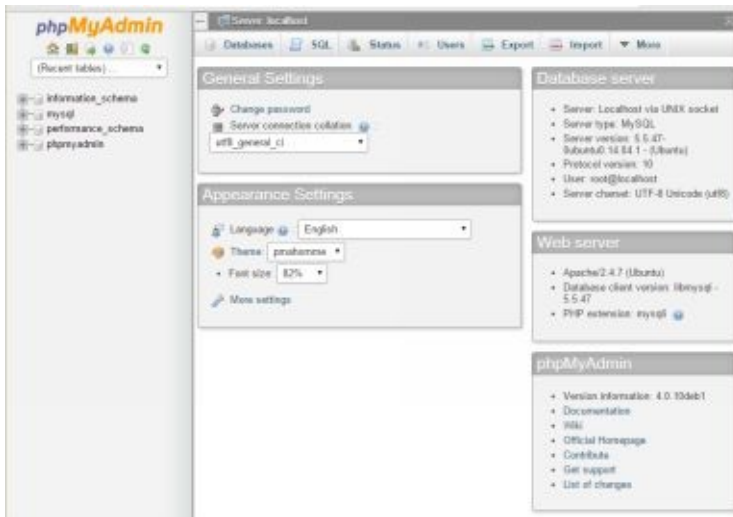
`Do you want to continue? [Y/n] y`



During the phpmyadmin installation, it will prompt you to select which web server you would like to configure. Next to apache2, press the spacebar, then tab to ok, and press enter. Shortly after, the installation will ask you if you want to configure database for phpmyadmin with dbconfig-common. Make sure that yes is selected and press enter. Afterwards you will be prompted to enter the password for the administrative user. Complete the prompts and continue.

Now that phpmyadmin is installed, we can see if it works by going to `http://myipaddress/phpmyadmin` and entering our credentials.

If your installation was successful, you should see a screen like the one below:



We will get back to more of this later, but there are a few more things that we should learn first.

By now we know some of the basics of installing, updating, navigating, and setting up a VPS.

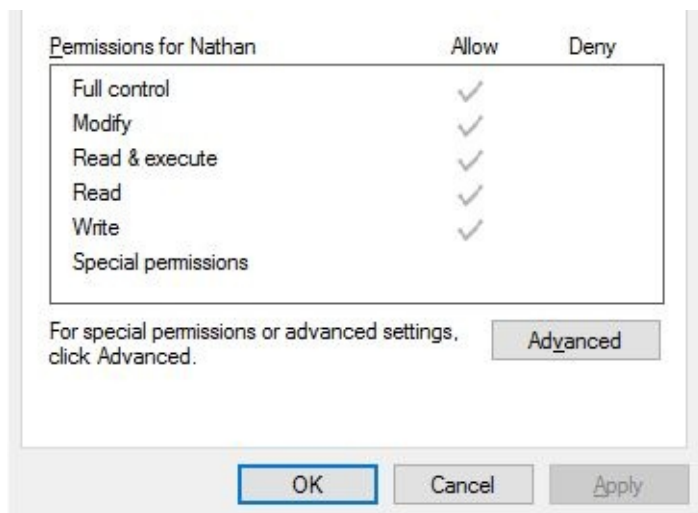
Next we will take a look at understanding, Linux file and directory permissions.

CHAPTER 8

UNDERSTANDING PERMISSIONS

To understand how to allocate permissions to users and groups, we must first understand how the Linux permissions work. It is much easier than determining Window's Server permissions, once you establish an understanding for it.

On a Window's Server there are a variety of permission types to allocate. See the figure below:



Ubuntu and other Linux systems have three basic permission groups and three basic permission types.

PERMISSION GROUPS

Owner – The owner permissions apply only to the owner of the file or directory.

Group – The group permissions apply only to the group that has been assigned to the file or directory.

All Users – The all user's permissions apply to all users on the system. This is the permission group you will want to be the most careful of.

PERMISSION TYPES

Read – The read permission refers to a user's capability to read the contents of the file, but does not allow them to write or save changes.

Write – The write permissions refer to a user's capability to write or modify a file or directory.

Execute – The execute permission affects a user's capability to execute a file or view the contents of a directory.

We can view the existing permissions and check them by using the CLI and our familiar *ls* command, with the *-l* flag.

List Files with Details

Command	<code>ls -l</code>
Result	Lists the files in the current directory with details
CLI Display	<code>milo@ubuntu:~\$ ls -l</code> <code>-rw-rw-r-- 1 milo milo 196 Feb 13 20:48 notes.txt</code>

Dissecting the output:

```
-rw-rw-r— 1 milo milo 196 Feb 13 20:48 notes.txt
```

The first section that is displayed, `-rw-rw-r—`, shows the permissions for the file.

When we are viewing the permissions by using the `ls -l` command, we can understand them by using the following key.

Understanding File Permissions	
Read	r
Write	w
Execute	x

As we view the permission for our `notes.txt` file, we can make a few observations based on the output: `-rw-rw-r—`.

The first three characters `-rw`, tells us that the user `milo` has read and write permissions for the file `notes.txt`. The second set of characters shows us that the group `milo`, also has read and write permissions. The last, shows us the all users permissions. From looking at the output, all users can read the file, but not write to it.

USING BINARY REFERENCES TO SET PERMISSIONS

Noe what we understand the basics of permission groups and permission types, setting permissions using binary references, should be fairly simple. It is important to understand that when you set file permissions this way, it is done by three integers. An **integer** is a whole number (Example: 1, 2, and 4).

To set permissions, we must determine the binary representation of the `rwX` string. **RWX String** is the binary representation that allows us to set read, write, and execute permissions.

We must use the table below to calculate the permissions we want for a file or directory.

File Permissions: Binary Representation	
Read	4
Write	2
Execute	1

We get the permission number by adding the integers together. We must do this for each of the three permission groups.

For example:

Read Access = 4

Read and Write Access = 6

Read, Write, and Execute Permission = 7

Each permission group needs a number. The number we make for setting permissions will be 3 digits. Remember, the order is user, group, and all users.

Exercise 1: What does a permission of 777 mean?

This means that the user, group, and all users have read, write, and execute access. Using 777 as a permission setting is not good for establishing strong security.

Exercise 2: What does a permission of 655 mean?

This means that the user has permission to read and write. The group has permission to read and execute. All users can also read and execute.

Exercise 3: What does a permission of 444 mean?

This means that both the user, group, and all users have only read access to a file.

Exercise 4: What does a permission of 755 mean?

This means that the user has read, write, and execute permission. The group and all users can only read and execute. This is often the permission setting recommended for the web directory for those using WordPress.

Before we go over the steps of applying permissions we should first learn how to transfer files. That process will open the door for applications in which we will learn to set permissions.

FTP TRANSFER

Assuming that you want to install WordPress, an easy way to do that is by using FTP. **FTP** stands for file transfer protocol and is used to transfer files between a client and server.

To connect and send files to our server we will use a program called FileZilla. FileZilla offers both a FTP client and server. For us to upload files, we need the client. You can download FileZilla from: <https://filezilla-project.org/>

Our DigitalOcean VPS already has FTP capability preloaded for SFTP, which is what we will be using. **SFTP**, uses SSH to transfer files using secure shell. There are other options available that you could install to your server for FTP, if you decide not to use the one that I recommend.

One that I have used and would recommend is VSFTPD, but we are not going to walk through the setup of that in this beginner's book.

Let's make a file on our computer to transfer. Open notepad and create a file named *ftptest.txt*. You can type whatever you like within the file, but we will use this to test file transfer.

Once you have that file created and have the FileZilla Client installed we can continue by launching the FileZilla Client.

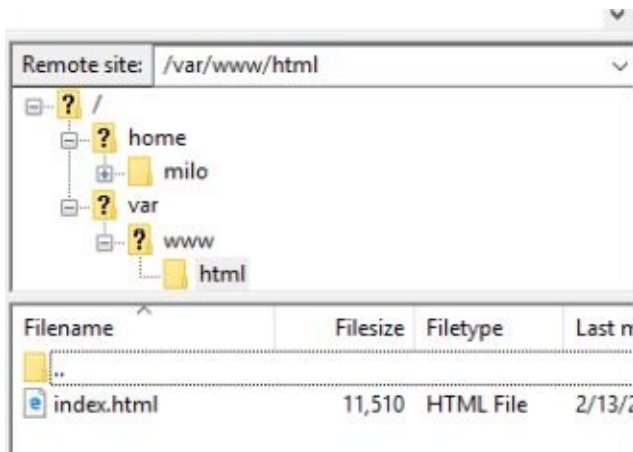
To connect some information will need to be entered in the top of FileZilla: Host, Username, Password, and Port.

In the host field we will need to enter our IP address. In the username field and password field we will enter the user milo and his password. The port number for SFTP is 22. Once that is entered we can click Quickconnect.



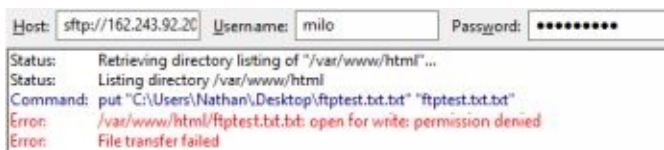
Once you connect, you will see the home directory for milo on the right of FileZilla. On the left you will see your local computers file structure. Navigate to the folder on your PC that you made the *ftptest.txt* file and drag it over to milo's home directory. In a few seconds you will see that the transfer was successful. Now if we log into our server with PuTTY and use the *ls* command, we will see the file.

If you wish to use your VPS for a web server, we will need to put files into the `/var/www/html` directory. In FileZilla, change the remote site directory to: `/var/www/html`. See below for reference:



Now let's try to upload our *ftptest.txt* file to this directory.

Your upload will fail as seen in the image below.



Why is that? Well, we do not have the permissions for milo to upload to that directory.

OVERVIEW: OWNERSHIP OF FILES AND DIRECTORIES

In the next few pages in this book we will allow milo to upload the files for WordPress and also learn more about changing file or directory ownership. We will do this by changing permissions and then modifying them to the recommended permission settings for WordPress.

Log into your VPS and when you are ready, run the command in the table below.

Changing the Owner of a Directory

Command `sudo chown milo -R /var/www`

Result System will make milo the owner of the directory, along with all subdirectories.

CLI Display `milo@ubuntu:~$ sudo chown milo -R /var/www`

[sudo] password for milo:

milo@ubuntu:~\$

To review the command above we must note that we need to use *sudo*. This elevates the permissions for milo. Without the use of *sudo*, we will get errors and the command will

fail.

After *sudo*, we use *chown* which allows us to change the owner of the directory. The command *chown*, requires a user (milo) and a directory (/var/www). The flag *-R* will make milo the owner of all the subdirectories as well. By using this, milo also owns the html directory.

While we do not currently have an example to show the changing of what user owns a file, you can follow the table below. It is much similar to changing the owner of a directory. Experiment with this to practice.

Changing the Owner of a File

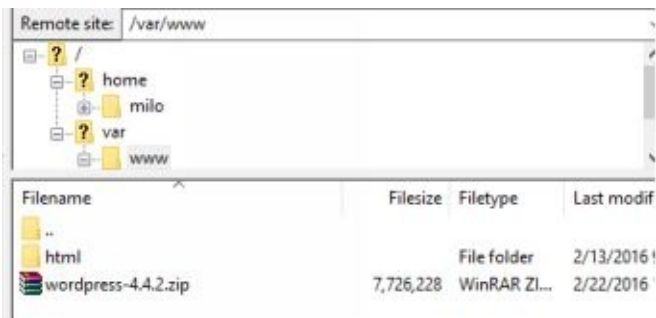
Command	<code>sudo chown milo /home/milo/test.txt</code>
Result	System will make milo the owner of the file.

For the next steps in this example guide, we will need to download WordPress. You can download this at: <https://wordpress.org/download/>

Once you have downloaded the .zip file with the WordPress files, then upload it to the /var/www/ directory using FileZilla. Remember, you can browse to it or drag and drop it into the interface.

When you finish your upload, it should look like the image below.

Important Note: WordPress updates constantly so your version may be different.



If you have this, then you can log into your VPS. We can now install a package to unzip the WordPress files if one is not already included.

UNZIPPING .ZIP FILES

Install the Unzip Package

Command `sudo apt-get install unzip`

Result System will install the unzip package

The DigitalOcean VPS does not come with this, so we will need to install unzip. The installation process should begin to look familiar. Again, we use *sudo* to give milo extra access to install packages and make system changes.

Once the installation is complete, change to the `/var/www` directory.

Changing to `/var/www` Directory

Command `cd /var/www/`

Result System will move to the `www` child directory under `var`.

By now a lot of this should be getting easy with navigating and installing software.

Once we are in the `/var/www` directory, use the *ls* command. By using this we can see our WordPress zip file. Now let's use unzip to extract those files.

Unzipping a Zip File

Command `unzip wordpress-4.4.2.zip`

Result System will unzip the file `wordpress-4.4.2.zip`

Important Note Recap: WordPress updates constantly so your version may be different. Your `.zip` file name may vary.

Run *ls* again and you will notice that we now have a WordPress folder.

In the next several pages, by changing the Apache configuration file and continue the discussion on permissions for this practical use.

MODIFYING APACHE2 SITE CONFIGURATION

Since WordPress's zip file creates a new directory in the `www` directory, we need to tell Apache to point to the new directory. You can host a variety of sites on your server by creating configuration files in the `sites-available` directory, which is located at `/etc/apache2/sites-available`.

Let's change to this directory using `cd` and see what is in there using the `ls` command.

Moving to Site-Available and Listing Files

Command	<code>cd /etc/apache2/sites-available; ls</code>
Result	System will move <code>mil0</code> to <code>sites-available</code> and list the site configuration files.
CLI Display	<code>mil0@ubuntu:~\$ cd /etc/apache2/sites-available; ls</code> <code>000-default.conf default-ssl.conf</code>

If you host multiple sites, you will need multiple configuration files. This book will explain how to enable and disable these configurations, but to understand them more in-depth, I would recommend finding a detailed guide on Apache configurations. That's a book in itself, but I will try to cover the basics.

For what we are doing, we will want to modify the `000-default.conf` file. Just as a point of reference, this is the default Apache site configuration file. Remember that to use `nano` to edit this file `mil0` will need super user access by using `sudo`.

Using nano to Modify 000-default.conf

Command sudo nano 000-default.conf

Result System will launch nano to edit the
000-default.conf file.

Below are the contents of the *000-default.conf* file.

000-default.conf

```
<VirtualHost *:80>
```

```
# The ServerName directive sets the request scheme,  
hostname and port t$
```

```
# 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/html
```

```
# Available loglevels: trace8, ..., trace1, debug, info, notice,  
# 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

</VirtualHost>
```

As we look at this file it may appear overwhelming. What is important to know is that lines that start with the pound/hash sign (#) are comments used to help users understand how to use it.

If we omit the comments in this file, it is much easier to digest. Leave the comments in your file, you may need them for reference, but below you can see the substance of this configuration file without them.

000-default.conf

```
<VirtualHost *:80>

#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

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

</VirtualHost>
```

This is the basic framework in the configuration that outlines: domain name, server admin, document root location, and the location for error logs.

I left the one comment “#ServerName www.example.com”. We will get back to this in a later chapter. It is commented out because we do not have a domain name pointing to this server yet.

What we need to update is the DocumentRoot line and the ServerAdmin line. You may enter your email address in the ServerAdmin line.

Change the DocumentRoot to the following: /var/www/wordpress

When you are done, the configuration (excluding comments) should look like the one on the next page.

000-default.conf (Updated)

```
<VirtualHost *:80>

#ServerName www.example.com

ServerAdmin info@nathanneil.com
```



```
DocumentRoot /var/www/wordpress
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
CustomLog ${APACHE_LOG_DIR}/access.log combined
```

```
</VirtualHost>
```

Once the changes have been made, use control and x to exit. Hit Y and Enter to save the changes.

Before we move back into talking about changing permissions, we should consider how to enable and disable site configurations.

Disable a Site Configuration

Command	<code>sudo a2dissite 000-default.conf</code>
---------	--

Result	System will disable the site configuration for 000-default.
--------	---

CLI Display	<code>milo@ubuntu:/var/apache2/sites-available\$ sudo a2dissite 000-default.conf</code>
-------------	---

Site 000-default disabled

To activate the new configuration, you will need to run service apache2 reload.

`milo@ubuntu:/var/apache2/sites-available$`

When you make changes to Apache2 configuration files, you will almost always need to restart apache. You can do so by using the following command: *sudo service apache2 restart*

When that is complete the new settings will take hold. Now since we want to use the 000-default.conf site configuration, we need to re-enable it. Follow the table below.

Enable a Site Configuration

Command	<code>sudo a2ensite 000-default.conf</code>
Result	System will enable the site configuration for 000-default.
CLI Display	<pre>milo@ubuntu:/var/apache2/sites-available\$ sudo a2ensite 000-default.conf</pre> <p>Enabling site 000-default.</p> <p>To activate the new configuration, you will need to run service apache2 reload.</p> <pre>milo@ubuntu:/var/apache2/sites-available\$</pre>

Once again, we need to restart apache with the following command: *sudo service apache2 restart*

Now if we load our IP address into the browser, we will see the WordPress installation landing page. Do not click “Let’s Go!” yet as we still have some permissions to update.



RECAP ON CHANGING OWNERSHIP AND PERMISSIONS

Now before we can continue, we need to make some ownership and permissions changes. This is a great way to pull the whole chapter together.

Once you are logged into the server, we must first change the owner of the folder to a non-privileged user, who is already built into the Apache2 install. This user is *www-data*.

Changing the Owner of a Directory

Command `sudo chown www-data -R /var/www`

Result System will make milo the owner of the directory, along with all subdirectories.

CLI
Display `milo@ubuntu:~$ sudo chown www-data -R /var/www`

`[sudo] password for milo:`

`milo@ubuntu:~$`

Now we need to set the permissions for the directories.

WordPress recommends using 755 as the permission binary.

This means that the user (*www-data*) has read, write, and execute permission. The group and all users can only read and execute. This allows *www-data* to make changes to the files in the directories, but other users can only read and execute. These permissions are needed for a user to load the website.

Changing Permissions of a Directory

Command `sudo chmod 755 -R
/var/www/wordpress`

Result System will make change permissions for /var/www and all subdirectories.

CLI Display `milo@ubuntu:~$ sudo chown www-data -R /var/www/wordpress`

`[sudo] password for milo:
milo@ubuntu:~$`

Remember using the `-R` flag makes the settings apply to all subdirectories as well. If we do not use this, we will not be giving the permissions to other subdirectories in the wordpress parent.

Now that the permissions are set, we can move forward in creating a database in MySQL.

CHAPTER 8

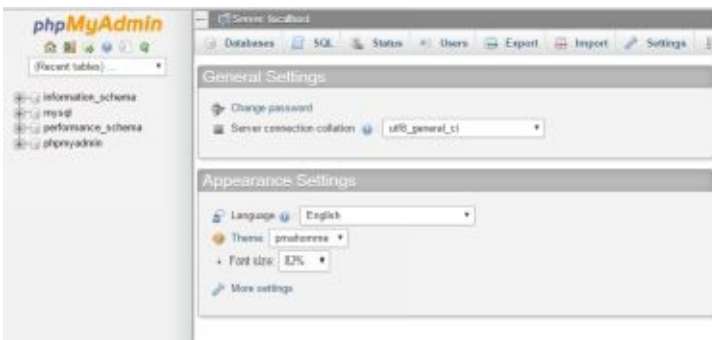
CREATING A DATABASE IN MySQL

There are a multitude of books on MySQL. This chapter is going to be very short, but will cover how to make a database for installing almost any popular content management system or other web software that needs a database to use.

I typically use the CLI for managing MySQL, but since this is geared towards beginners, it may be easier for us to use phpMyAdmin to create the database.

From the web browser, enter your IP address/phpmyadmin (Example: <http://162.243.92.xxx/phpmyadmin>). Replace 162.243.92.xxx with your IP address. Then login to phpMyAdmin.

You should see a home dashboard like the one below:



What we are going to do is create a special user for WordPress to use and a database that it is limited to. This is a good security practice.

In the top left, click on Users.



Now click on Add User.

From the new window enter a user name for the new user. I am going to use: *website*. Leave host listed as any host. After that skip down and click the “Generate” button to

create a secure password. Be sure to note this password as we will need it for the WordPress installation.

Under the generate button is a field to “Create database with same name and grant all permissions.” Check that box. Leave “Global Permissions” all unchecked, since we only want WordPress to access this one database.

Scroll down and in the bottom right click the “Go” button. You will have a popup that the user has been added and on the right side of your screen you will see that the new database for the user has been added.

Again, this is just the basics, but it is an easy and secure way for beginners to add a database.

If you wish to learn more about MySQL and how it works from the CLI, there are a multitude of books out there to serve that need as you grow.

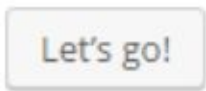
CHAPTER 9

BONUS CHAPTER

SETUP WORDPRESS

There is still a lot that you need to learn to establish the fundamentals of using Ubuntu, but I think this book is good in the sense that it covers some basics, along with a practical use. With the proper permissions in place and a database setup, we can proceed to setup WordPress.

In your web browser enter the IP address of your server. You should then see the “Welcome to WordPress” starting page. At the bottom of the page click on the “Let’s Go!” button!



Now it is going to ask you for the following information. If you followed my example earlier, it should look similar to this:

Database Name: website

User Name: website

Password: [Your randomly generated password]

Database Host: localhost

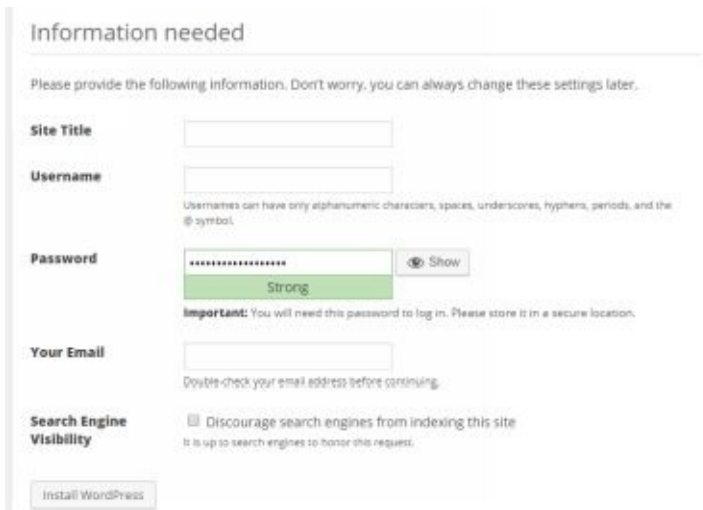
Table Prefix: wp_

Once that information is entered you will be prompted to run the install.



Go ahead and click “Run the install.”

You will then be prompted for the information you want to use for your WordPress Site.



The screenshot shows the 'Information needed' step of the WordPress installation process. It includes fields for Site Title, Username, Password, and Your Email. The Password field shows a strength indicator as 'Strong'. There is a checkbox for 'Search Engine Visibility' which is currently unchecked. At the bottom is an 'Install WordPress' button.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title

Username

Username can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password

Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email

Double-check your email address before continuing.

Search Engine Visibility ☐ Discourage search engines from indexing this site
It is up to search engines to honor this request.

Provide the requested information and click “Install WordPress.”

Congratulations! In just under 70 pages we have established some Ubuntu basics and used that information to build a basic webserver.

There much more to learn of course, but this is just the beginning. This guide should have you comfortable with navigating and installing packages on Ubuntu.

If you have any questions for further learning, do not hesitate to email me at info@nathanneil.com.

Thank you and good luck as you continue learning!