The No Bullshit Bible : Creating Web 2.0 Startups & Programming

Written by James Gillmore
Edited by Holly Welch

FaceySpacey

www.faceyspacey.com

Linux

TABLE OF CONTENTS

Technical

Chapter 5

5-1 LINUX SERVER SETUP | SETTING UP LAMP & AMAZON WEB SERVICES

Many things in software development rely on winning recipes. This is one of them, and the one that kicks it all off--one that's required if you have any hope of getting anywhere in PHP coding and getting your application production ready. Quick note: if you don't already know the, the acronym, "LAMP," stands for Linux, Apache, MySQL and PHP, i.e. the 4 applications necessary to run the sort of apps these tutorials are written for.

The goal is this: to install LAMP using the Ubuntu flavor of Linux on an Amazon EC2 server. This by the way is the official FaceySpacey recipe. Ubuntu Linux is what we live by, and quickly become the best Linux distribution, especially for newbies like you. Here's the official Amazon EC2 Ubuntu setup guide:

https://help.ubuntu.com/community/EC2StartersGuide

On that page is links to the latest Amazon EC2 AMIs to install. If you don't know what an AMI is, it's basically a template that you generate your Amazon EC2 server from. As of August 2011, here's the latest set of AMIs:

http://uec-images.ubuntu.com/releases/11.04/release/

On that page is a table. You will be clicking the link in the AMI column. Find the row that corresponds to the region closest to you that has a 32 bit architecture and a "root store" of "EBS." Click the link in that row, and it will link to your Amazon account and begin the process of creating an EC2 instance based on that AMI/template.

If you haven't signed up for Amazon AWS yet, do so here:

http://aws.amazon.com/

Lately, they've been making you signup for each service you will use. Signup for EC2 to begin with.

Ok, so back to creating your instance. Go through the steps, and choose the following options when presented with them:

```
    1. set the instance type to m1 small
    2. create a new keypair and download the corresponding files to your
       computer
    3. for everything use the defaults--at least for now
    4. set the instance type to m1 small
    5. create a new keypair and download the corresponding files to your
       computer
    6. for everything use the defaults--at least for now
```

After you've created your instance, go to the Security Groups link in the left column. Click the "default" security group which was assigned to your EC2 instance. Set the rules so HTTP traffic is allowed from all IP addresses, and allow SSH traffic from the ip address of your computer. Find your IP address here:http://whatismyipaddress.com/ .

The next thing you will do is set your server up so you can login to it without using the keys provided by the Amazon EC2 web interface. Though experts will not recommend accessing your server via "root access," I'm going to teach you how to do it since if you're new to this you're most likely dependent on root access for much of what you do.

But let's first login once with the keys provided by the EC2 web interface, since that's the only way to initially get in. Click the "Instances" link in the left column. Then right-click your instance row and click "connect." There will be a set of directions there to follow. Basically it boils down to executing a command (from Putty on windows), or the Terminal on Macs and linux, and making sure the private key you created when you first created your instance is associated with the connection command. One thing to note is that you will connect as the “ubuntu” user, rather than the root user. That’s how Ubuntu AMIs are setup.

Once you’re logged in as the Ubuntu user, setup the password for the root user with this command:

```
# sudo passwd root
```

After following the steps to change the root password the above command will lead to, navigate to /etc/ssh/sshd_config and uncomment this line: PasswordAuthentication yes

Then restart your SSH server with this command:

```
# sudo /etc/init.d/ssh restart
```

Now you can connect to the server remotely via the root and the password you set. Do that, i.e. login via SSH from the terminal with with user “root” and the password you set.

Next, now that we’re basically in control of your fresh EC2/Ubuntu Linux server, it’s time to install the applications you will need. First you need to update your server so that it gets the latest patches and updates from the internet, i.e. like your Mac or Windows computer regularly--and automatically--updates itself. Run this command:

```
# apt-get update
```

Now execute the following commands one after another. Wait for each to complete before doing the next one:

```
# apt-get install lamp-server^
# apt-get install curl libcurl3 libcurl3-dev php5-curl php5-mcrypt

# apt-get install mercurial

# apt-get install phpmyadmin

# apt-get install chkconfig

# apt-get install subversion

# apt-get install unzip

# apt-get install php-pear

# apt-get install php5-dev

# apt-get install apache2-dev
```

We installed a few tools we commend there such as “mercurial,” which is a version control application--we’ll talk about that stuff in other articles. Just know this is our recipe to get everything done efficiently all at once, and that’s why installing such apps are in this article.

Ok, so next go back to the EC2 web panel. Go to the “Elastic IPs” link in the left column. Click the “Allocate New Address” button and create a new IP address. Then right-click the row that appears corresponding to your new IP address, and choose the “Associate” option from the menu that appears, and associate the IP address with the EC2 instance you previously created.

Now copy/paste that IP address into your web browser and visit that URL. You should see a message saying that your server is setup. Next go back to the terminal where you’re connected to your server, and navigate to /var/www. In that directory will be a file called index.html. Rename it to index.php like this

```
# mv index.html index.php
```

Now in that file, replace all the code there with:

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

Now you should be able to revisit that IP address URL and see a bunch of info about your server in pretty blue tables. That's all the information about your server, the settings, and what is installed on it. This also means that PHP is working on your server, as well as of course Apache, which is your "web server" application that works in conjunction with PHP to display web pages to remote web browsers trying to connect to your server.

What next? Basically from here, you will create your applications in directories such as/var/www/application-name . After that, you will need to point your domains from godaddy (or wherever your domain names are hosted) to your server, and then on your server, you will associate an incoming domain name with directory path such as the aforementioned /var/www/application-name. We'll describe this in the next tutorial.

5-2 LINUX SERVER SETUP | CONNECTING YOUR DOMAIN NAME TO YOUR APPLICATION DIRECTORY

Ok, so this is a turning point for all those that have never setup code to be accessed through the internet. The end goal is simply so you can access www.yoursite.com in a web browser and see the result of your code displayed in the web browser. The first time I executed this, it made me really happy, and you'll be sure to have the same reaction.

The high level explanation of what's happening is simply that you matchup a domain name with a directory on your server. You tell your server that any traffic coming for www.yoursite.com should connect to the code located at /var/www/yoursite.com.

Navigate to */etc/apache2/sites-available*. Open up the file there called "*default*". Also note that if you don't have a tool like SCP on windows you'll have to edit these files at the command line using an application that comes with Linux called "*vim*". So for example to access that file would type the following:

```
# cd /etc/apache2/sites-available
# vim default
```

Vim is a text editor for use over SSH in the terminal. Google "Vim tutorial" to learn how to use it, as it's out of the scope of this tutorial. It will be a little abnormal at first, but it won't be hard to get the hang of.

Also, what I love about running Ubuntu Linux on my own desktop is that you can connect to servers and access their file system the same way you access your local file system. You can easily drag and drop files back and forth between your server and your desktop computer without having to do so in an FTP/sFTP program. I highly recommend anyone serious about getting their startup going that they install Ubuntu on their home computer. It's really not that difficult to install. Just go here and follow their directions: http://www.ubuntu.com/ . Once you have it installed, just go to *Places > Connect* to Server and then connect to your EC2 instance via your root ssh credentials. That's one reason why I violate the security precautions of using the Amazon-provided SSH keys...The other thing you can do is open up files on your server in text editors and IDEs just as you would files on your own computer.

Anyway, you've gotten to the point where you can edit the text files on your server. You're now going to setup some Apache configurations to make your site accessible from yoursite.com . Open up that “default” file and add the following configuration lines to the bottom of the file (these are the bare minimum you need--there’s a lot more you can do to configure it, but to convey the main concept here, we’ll only show the minimum):

```
<VirtualHost *:80>
        ServerName yoursite.com
        ServerAlias *.yoursite.com
        DocumentRoot /var/www/yoursite.com
</VirtualHost>
```

Note that /var/www/yoursite.com is actually the name of a folder, i.e. “*yoursite.com*” there is a name of a folder. You don’t have to put the .com, but I like to in order to be very consistent and connote that this folder contains a full-fledged website.

Now in that folder put a file similar to that index.php file made in the previous Setup tutorial. Now go towww.yoursite.com or yoursite.com and you should see the same result, i.e. the information about your server. If you do, you’re money and good to go.

If you haven’t setup your domain on Godaddy to point to your server, do so now. I won’t go into deep detail about it because you can google something like “godaddy dns tutorial” to figure it out very easily. But basically, you’re going to navigate to My Domains, and then to the yoursite.com domain, and then to “Manage DNS Settings” and then enter the IP address of your server near the top of the page, and save the settings. I won’t go into much about the technical terminology about how you’re essentially making your “A record” (i.e. a DNS record) point out your ip address. Just make it happen. It’s no big deal.

You may need to restart your server, so at the command line type:

```
# /etc/init.d/apache2 restart
```

It also may take some time for Godaddy to successfully point your domain at your server, but I've found it almost always to be instantaneous.

5-3 LINUX SERVER SETUP | INSTALLING Yii & YOUR APPLICATION

This tutorial is all about getting Yii to work on your server. We're going to optimize your server to best perform for Yii, install the Yii Framework library code, and generate the beginning of your application so it's connected to all the framework code.

First run the following commands to install a tool called "APC" which is a caching application that makes PHP execute faster:

```
# pecl install apc
```

Now add the following lines at the bottom of this file /etc/php5/apache2/php.ini :

```
[apc]
extension=apc.so
apc.enabled=1apc.shm
size=512M
```

Then restart Apache with the **`/etc/init.d/apache2`** restart command

Also, in this file */etc/php5/apache2/php.ini* you were just in, you might as well change your time zone to UTC now so you're running a common & consistent time zone for all the developers that may be working on your app. Google "UTC" if you're not sure what that's about. Search that file for "date.timezone" and change that line to:

```
date.timezone = UTC
```

Also, lower your error reporting in that file by ammending the following line to look like this (obviously search for the "error_reporting" directive similar to how was done above):

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_NOTICE
```

Next, for the URL rewriting rules generated by your upcoming Yii application to work, execute the following command:

```
# a2enmod rewrite
```

That’s a simple command that installed a “module” to your Apache web server application. This application is called “Mod Rewrite” and it’s basically the most common Apache module. What it does is allow you to do things like hide file names such as “index.php” from your URLs, and generally aid in beautifying your URLs. Yii has a bunch of tools to automatically beautify your URLs, but it needs mod rewrite installed to do so. Specifically, yii will generate a file called .htaccess in your root application directory that will have these rules. It may be a hidden file, so you will have to unhide it to read its contents if you would like. Check that file out after you install Yii and your base application code in the following steps.

Now navigate to your /var folder and execute the following command

```
# cd /var
# svn co http://yii.googlecode.com/svn/trunk yii
```

That will install Yii here: /var/yii . That’s just the place I like to put it on all my servers so that I always know where it is. Remember this is the FaceySpacey formula. We do things in a consistent way so we know what to expect from server to server.

Also note that we can run the “svn” command because we installed the “subversion” application in first Setup tutorial. Subversion is another version control system. It’s one of the most popular ones, and it just so happens that Yii uses it. We don’t recommend it for development, but you need it to install yii like we just did. The benefit of using

SVN to install yii is that you can run the following command to update your framework any time it’s improved:

```
# cd /var/yii
# svn update
```

In that command you navigate to your Yii code repository, and update its code.

Now, it’s time to generate the beginnings of your application. The idea is that Yii provides a command you can execute from the command line that will generate the folder structure for your application and connect it to the framework code. Execute the following from the command line:

```
# /var/yii/framework/yiic webapp /var/www/yoursite.com
```

To learn more about what’s generated, go here:http://www.yiiframework.com/doc/guide/1.1/en/quickstart.first-app

But generally, here’s the folder structure that will be generated:

```
testdrive/
  index.php                 Web application entry script file
  index-test.php            entry script file for the functional tests
  assets/                   containing published resource files
  css/                      containing CSS files
  images/                   containing image files
  themes/                   containing application themes
  protected/                containing protected application files
     yiic                   yiic command line script for Unix/Linux
     yiic.bat               yiic command line script for Windows
     yiic.php               yiic command line PHP script
     commands/              containing customized 'yiic' commands
        shell/              containing customized 'yiic shell' commands
     components/            containing reusable user components
        Controller.php      the base class for all controller classes
        UserIdentity.php    the 'UserIdentity' class used for authentication
config/                 containing configuration files
        console.php          the console application configuration
        main.php             the Web application configuration
        test.php             the configuration for the functional tests
     controllers/            containing controller class files
```

```
         SiteController.php  the default controller class
      data/                  containing the sample database
         schema.mysql.sql    the DB schema for the sample MySQL database
         schema.sqlite.sql   the DB schema for the sample SQLite database
         testdrive.db        the sample SQLite database file
      extensions/            containing third-party extensions
      messages/              containing translated messages
      models/                containing model class files
         LoginForm.php       the form model for 'login' action
         ContactForm.php     the form model for 'contact' action
      runtime/               containing temporarily generated files
      tests/                 containing test scripts
      views/                 containing controller view and layout files
         layouts/            containing layout view files
            main.php         the base layout shared by all pages
            column1.php      the layout for pages using a single column
            column2.php      the layout for pages using two columns
         site/               containing view files for the 'site' controller
            pages/           containing "static" pages
               about.php     the view for the "about" page
            contact.php      the view for 'contact' action
            error.php        the view for 'error' action (displaying external errors)
            index.php        the view for 'index' action
            login.php        the view for 'login' action
```

You should now be able to visit yoursite.com and see the basic Yii web app that was generated. That is, if you previously connected your domain to your server, and accordingly the path to /var/www/yoursite.com.

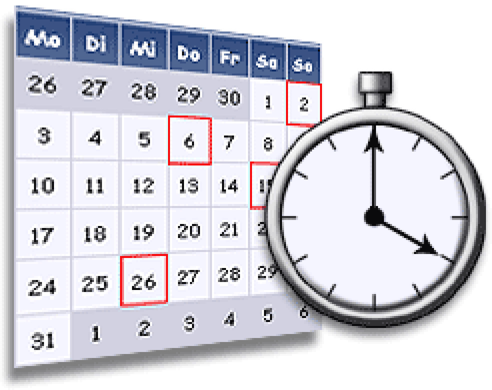
The next thing to do is to follow the instructions on the above link to the Yii site. I won't re-write it all, but I'll give you the high level explanation you need to get the picture so you know what you're looking at as you dive into the deeper details and specifics there. Basically, that tutorial will describe how to connect your new application to your MySQL server and database (which you previously installed). Then it will describe how to run a web interface bundled with Yii that will automatically generate base code based on the tables in your database. Right now would be the time in the development process to build all the tables your database needs. As you'll read in the speccing tutorials, it's very important to prepare as many of your tables as you can ahead of time. If you've specced well, you should be able to know exactly what tables you'll need to accomplish the product goals.

I'd like to point out how configuration works. In Yii, and many software applications, you have a central and single "configuration file" where you can configure the specifics of your application, i.e. database connection details. What this does is make it so you don't have crucial specifics strewn throughout your codebase that are impossible to find at a later point in time. This way at any point in time you know just to go check your configuration file and make adjustments there, rather than spend hours hunting down hidden "switches."

Another thing to note is specifically what the yii code generation tools--called "Gii"--are doing. Specifically they build the base code you need in your Models, Controllers and Views. The most important thing here is that it shows new Yii developers how to build applications. It serves as a great starting point. Eventually you may end up not using the code generation tools, but nevertheless in the beginning, it will point you in the right direction. Make sure to examine all the files generated and what's in them. Check your Models, Controllers and Views folders. They will all be in `/var/www/yoursite.com/protected/` .

5-4 LINUX TOOLS | CRON JOBS

Cron jobs--or more accurately “Crontab”--is a common Linux application you’ll hear about early on in your ventures into web development. Cron jobs let you trigger code in intervals in the background on your server. Many crumby hosting providers provide a way to trigger “cron jobs” in entered intervals.



If you’re running your own server, you’ll type a command such as:

```
# crontab –e
```

and see a simple text file that line by line lists scripts to trigger in given intervals. For example, one might look like this:

```
1 11 * * * commandname >> /var/log/log-name.log
```

That one for example will trigger the command, “*commandname*” every day at 11am and record any output produced by that command into the file log-name.log.

The idea behind cron is simply that you can trigger routine tasks in recurring intervals. Other examples of such intervals are every minute, every 30 minutes, once a week, once a month, etc.

You can even trigger PHP scripts. Yii has tools to create “console applications,” i.e. applications/commands you can trigger from the command line of your server. To learn more about them, check out the tutorial on the Yii site:

http://www.yiiframework.com/doc/guide/1.1/en/topics.console

To learn about using cron in Ubuntu, here’s the official Ubuntu tutorial:

https://help.ubuntu.com/community/CronHowto

Overall, the thing to note is that cron is extremely easy to use. It’s not anything crazy. Write a PHP script using the Yii console application tools, and trigger it with a simple line in the crontab file. That’s it. The next thing to note is that cron isn’t the solution for everything. Often Daemons may be the real answer, and we’ll cover that in the next article. The general idea is that cron scripts can often overlap and get clogged, whereas daemons are constantly running in realtime and you can write logic that solves how to deal with these jams.

I’ll end this tutorial with an example of what cronjobs are good for so you know what challenges to look out for that are nicely served with an entry in your crontab file as a solution. Imagine you want to generate a report each day, maybe with the # of new user signups and other such data. Well, cron is perfect. Trigger your PHP script that will check the database for the # of new users since the same time the day before and email the result to yourself. Or perhaps, you are seeding your site with content from RSS feeds. You can write a PHP script to check these RSS feeds daily (or hourly or whatever) daily and insert the new articles in your own database of articles.

5-5 LINUX TOOLS | DAEMONS

A Daemon is a Linux program that is constantly running in the background. In fact, Apache and MySQL are both Daemons. They sit around waiting for connections, i.e. Apache waits for remote web request connections to your sites and MySQL waits for connections from your PHP code trying to select and insert stuff into your database. Daemons are different from cron jobs in that they can run more than every minute, and basically every miniscule moment of the day. In PHP, they're a long-running script that you can program to wake up and go back to sleep while doing tasks whenever it wakes up and sees it has a pending task to do. It does so through loops. One cool thing to note is that most the Daemons on your Ubuntu Linux server store logs of what they're doing in sub-folders and files within /var/log .

Most Daemons are written in C. However there are tools to make it very easy to create powerful Daemons in PHP.

I won't go into the technical details of writing them because the following tutorial is fantastic:

http://kevin.vanzonneveld.net/techblog/article/create_daemons_in_php/

Basically that tutorial describes tools the author made to create daemons, how to install the tool, and how to put it to use. Installing the tool is very easy. Just enter the following command at your command line:

```
# pear install -f System_Daemon
```

That requires the pear application to run, but fortunately you already installed that if you followed our Setup tutorials.

System_Daemon is a “pear package.” Pear packages are bundles of additional code you can attach to your PHP application that gives you additional functions/classes/methods to use. Pear itself is basically a tool designed for nothing bug augmenting the tools available to your PHP installation. You will use it often to install 3rd party PHP tools.

In this case after you’ve installed System_Daemon, you have access to a few a methods of the System_Daemon class that do all the real work of the Daemon, and the result is you can basically think and code in PHP terms you’re used to.

If you read that tutorial, you’ll see that the main function it offers is to loop through a task in an interval you set, i.e. it will attempt to perform the same task over and over again. You can then write code to do nothing if your code determines there is nothing to do. For example, if you want to check every moment for new users in your system and email them every time one is found, you can do so, and decide to do nothing if none are found. You can also safely break the system down if some error occurs.

You can also make it so your daemon script runs automatically every time you boot up your server. That way you don’t have to remember to turn it on and off again.

Another cool thing you can do is you can monitor the log file it’s writing too in realtime via the following command:

```
# tail /var/log/your-log-file-name.log
```

In the php code you write you specify the log file to write to in a simple configuration array and the daemon will write to it as it loops through the task its assigned. It will record each loop it makes, any errors, etc. So therefore you can monitor that file with the “tail” command and see what the daemon is doing in realtime. At FaceySpacey we really love the tail command. It puts what’s going on in your server at your fingertips.

Often many things are going on in your server, and you're not sure if it's working or not, and you want to know immediately, possibly while you're doing something to trigger breaking it.

So for cron-triggered scripts and daemons this is very important since you're not sitting there watching the results output to your screen. It's all happening in the background. So "tail" is your window into what it's doing.

Setting these Daemon tools up with Yii can be a little problematic. I'll cover it in detail outside of these begginer set of tutorials, but the basic idea is that you put all the code you see in the Daemon tutorial (linked to above) into a Yii console application script. You can then trigger your console application script like you would normally from the command line, but then a Daemon is triggered and set into motion in the background. It's really quite a powerful combination since you get access to all the Yii ActiveRecord tools to work with your database and models.

5-6 LINUX TOOLS | DOCUMENTATION (Doxygen)

Documentation is a required component for any long-term software application that has developers coming and going, or generally an expanding team. You won't just have to explain code to new developers, but also to yourself and your current developers when they look back at what they did even just a month ago.

The hardest part is maintaining documentation as your code changes. Often documentation can become outdated, yet it will still be there and you'll be looking at it as if it explains the current code, i.e. the code has changed but the documentation has not kept up.

The way to solve this problem is to automate documentation generation based on code comments (i.e. generate documentation from code comments), and to follow standard documentation practices, i.e. "PHPDoc." PHPDoc is the formal standard for commenting PHP code. It's a simple syntax to comment classes and their methods. So if you keep your comments up to date for just your classes and methods, you'll be good to go because of the following tool I'm about to describe that will automatically generate a mini-site of documentation based on those comments.

Doxygen is the tool we recommend to generate that mini-site. Here's it's site:http://www.stack.nl/~dimitri/doxygen/ .

To use it enter the following commands from your Ubuntu command line:

```
# apt-get install doxygen

# apt-get  install graphviz

# apt-get install mscgen

# doxygen –g
```

When you type ***doxygen -g*** it will generate a file in the current directory called "*Doxyfile*." That file is a very large configuration file. You need to edit this file. Here you enter all sorts of options, but most importantly the directory where your code is. You do so by specifying a simple "*input*" directive, as explained here:

http://www.stack.nl/~dimitri/doxygen/config.html#cfg_input

To see all configuration options go here:

http://www.stack.nl/~dimitri/doxygen/config.html

After you've set all your options, simply run the following command (where "Doxyfile" is the name of the config file):

```
# doxygen Doxyfile
```

The doxygen program will then generate a mini site in the location you specify with the "*HTML_OUTPUT*" directive:
http://www.stack.nl/~dimitri/doxygen/config.html#cfg_html_output .

Then obviously you setup Apache to display this site on the internet, and boom you have a mini-site with all your documentation. Just re-run this command any time you want to update it, and it will update it. This site will list all your classes, methods, and interlink pages together nicely, even generate all sorts of UML diagrams. UML diagrams are diagrams explaining code. Learn about it:http://en.wikipedia.org/wiki/Unified_Modeling_Language . Otherwise, all your code comments will be in there as well associated with their corresponding method or class. You of course get several ways to browse your documentation pages: by class name, method name, searching, etc.

That all said, the Yii Framework itself has awesome documentation in its "class reference" on its site. It's obviously automated as well, and they very possibly use Doxygen themselves, plus a few custom enhancements to the generate mini-site. I would love it if the contributors to the framework would share their code documentation tools as a feature of the framework itself so you could easily generate documentation of your Yii apps. I've been meaning to bring it up in their forum and probably will some time soon. If that ever comes out, forget Doxygen. Until then, Doxygen is our pick at FaceySpacey--and yes over PHP Documentor: http://www.phpdoc.org/ . We've used both extensively and Doxygen is a lot better. Doxygen also happens to support many languages besides PHP. You'd expect the one just for PHP to be better, but it isn't.

5-7 LINUX COMMANDS | USING THE LINUX COMMAND LINE

Basic knowledge of the Linux command line is key if you want to have any luck setting up a system to display your PHP code. Though on Windows and Mac (and Linux) you can install a desktop application that will basically do it all for you:

http://www.apachefriends.org/en/xampp.html

We don't recommend it for pro development. XAMPP is cool if you're learning PHP. Fine. But we're assuming the readers of these tutorials are serious about getting their application production ready. So again, if you want to just get practicing PHP, XAMPP may be for you. Go install it. It's really easy.

For everyone else, lets continue delving into the base requirements (i.e. most common commands) you need to know to run a professional Ubuntu setup (note: I actually started with XAMPP myself).

To navigate around directories use "cd" like this:

```
# cd /var/www
```

That will take you to the */var/www* directories.

```
# cd . .
```

will take you one directory lower back to */var*. And when in */var* if you type

```
# cd www
```

that will take you back to */var/www*. I.e. when you're in a directory you can move to its sub-directories by simply typing its name after "*www*" rather than the complete path.

To edit files in a text editor, type:

```
# vim filename
```

To move files type:

```
# mv /var/www/file-name /var/file-name-moved
```

That will move the file name and change its name. To simply change the name of a file of your current working directory type:

```
# mv file-name new-file-name
```

To copy a file in a similar way type:

```
# cp /var/www/file-name /var/file-name-copied
```

To delete a file type:

```
# rm file-name
```

or if not in the current working directory:

```
# rm /var/www/file-name
```

To extract a gzipped archive (i.e. the most popular archive format on linux) type:

```
# tar -xzvf archivename.tgz -C result-folder-name
```

To compress a directory and all its sub-directories and files type:

```
# tar -czvf archivename.tgz input-folder-name
```

The extract and compress commands are so important because so often you'll download compressed libraries of code you want to use, and of course you often want to back stuff up. It's clearly the equivalent of unzipping and zipping up folders/files on your Mac/Windows desktop computer that you've been used to doing.

The next very important skill is being able to change permissions of your files and folders. For example, often you'll want your files accessible by Apache, or more specifically the Apache user group, which is called "www-data". And of course you'll often want to protect files for security reasons. Here's how you change the group that owns a directory and all its sub-directories and sub-files:

```
# chgrp -R www-data yourfolder
```

And to change the user do this:

```
# chown -R root yourfolder
```

The -R specifies to do it recursively, i.e. to get all the sub-folders and the sub-folders' sub-folders and so on. Look up what "recursion" is some time: http://en.wikipedia.org/wiki/Recursion . It's an extremely important programming concept.

Next, now that you have groups and users that own your files, you can adjust the permissions of these files like this:

```
# chmod -R 777 yourfolder
```

That will give all users, groups and the public read/write access to the file. To give just the user owner read/write access type:

```
# chmod -R 755 yourfolder
```

The first number corresponds to the permission of the owner, the second the group, and the 3rd the plug. 7 is the most loose permission and 5 is tighter. 6 is in between obviously. There is also an alternate sytax for creating such permissions. Google “Chmod tutorial” some time to learn more.

5-8 LINUX COMMANDS | MORE LINUX COMMANDS

Let’s dive a little deeper into some more commands, building on what we just explored.To list all users in the system open up the following file:

```
# vim /etc/passwd
```

If it’s not quite clear, the idea is Linux allows you to have multiple users. Remember in these tutorials we’ve been logging in as the “root” user, but you can also login as other users, which perhaps have less permission to read and edit files. If you have other developers in your team you will want to only give them access to the files they’re working on. Also note to more experienced linux users why i recommend using “root” to start: the reason is because it removes complication and makes it so new Linux users have less to learn. Eventually they’ll learn about using the “sudo” command to preface their commands. For now, guys, google what “sudo” is.

To create a group type:

```
# groupadd groupname
```

To list all user groups open up the /etc/group file:

```
# vim /etc/group
```

To list all a specific user’s groups type:

```
# groups username
```

To add a new user type:

```
# adduser username
```

To add new users to a group type (this will also create the user):

```
# useradd –G groupname,groupname,groupname, username
```

To add an already existing user to groups type:

```
# usermod –a –G groupname,groupname,groupname username
```

To change an existing user’s primary group type:

```
# usermod –g groupname username
```

The benefit of using groups is that you can designate permissions by group, and therefore give access (or remove access) to a folder for a many users all at the same time. That’s the general idea of groups in programming in genera--it allows you to do less work by making the code you write function on behalf of a group. Then you simply need a mechanism to assign users to groups. Above are the standard user grouping mechanisms of Linux.

To update your Ubuntu installation, type:

```
# apt-get update
```

To install an application (as you saw in the server Setup tutorials) type:

```
# apt-get install app-name
```

e.g:

```
# apt-get install lamp-server^
```

To backup and restore MySQL databases use the following commands. First backup:

```
# mysqldump –u user_name –pYourPassword database_name > backup-
file.sql
```

That will log you into your MySQL application (aided by a complementary application called “mysqldump”) and make a backup of the database called “database_name” and save it in a file called “backup-file.sql” in your current working directory.

Now here’s how you restore that backup:

```
# mysql –u user_name –pYourPassword database_name < backup-file.sql
```

That will log you directly into MySQL and immediately restore the contents of “backup-file.sql” to the database named “database_name”.

One key gotcha about these commands which fooled me the first (and second and probably third) times I used them was that the “-p” parameter has the password attached to it with no space in between like the “-u” username parameter.

Another cool feature I’ll just describe now is you can actually log into remote databases on a different server and restore their contents into your own database on your server.

For example:

```
# mysqldump -u remote_user_name -pRemotePassword -h remoteserver.com
databasename | mysql -u local_user_name -pLocalPassword -C
```

```
new_database_name
```

That will make a backup on the remote server and then “pipe” the results of that command via the “|” symbol into a second command, specifically the mysql command and create a new database and fill it with the results taken from the remote server.

Piping one command into another FYI is a key tool to use at the Linux command line. All these commands have output, and if that output is suitable as input to another command, just put a pipe in between. You can continue that process of “piping” one command to another as long as makes sense for what you’re trying to accomplish. You can also use an arrow (“>”) to pass any output to a text file or use the double arrow (“>>”) to append the text output at the end of the file without replacing what’s already there. That’s very useful for tracking the results of your commands.

5-9 LINUX COMMANDS | SHELL SCRIPTING (Bash)

The final lesson I’d like to teach in the Linux Commands section is an explanation of what “shell scripts” are. Shell scripts are simple scripts you can make that have multiple commands in them. Then you can just execute the shell script and all the commands written it will execute without you have to typing them one after another. You do this coding in a simple language called “BASH,” which is the language the Ubuntu shell uses. But for basic scripts you don’t really need to know any bash. You could write something like:

```
#!/bin/sh

cd /var/log
```

```
rm log-file-name1

rm log-file-name2

rm log-file-name3
```

The first line is just required for the script to work. The #! symbols together are called "shebang" and basically are a signifier that the following is a bash a script. Don't worry about it much yet.

The next thing to think is: "Crap, I could trigger this script (or any other script) via a cron job" and have it do routine cleanup for me--in this case remove old log files.

So that's an example of the power of shell scripting. There's obviously a lot more you can do. For example, you can do loops and conditional logic like a PHP script--though you'll have to explore that on your own. For scripts that deal with your application's database, you probably just want to write a PHP script using Yii's "console application" tools, but for scripts that mainly deal in Linux commands, you definitely want to write a bash script.

Conclusion & Further Reading

We at FaceySpacey hope you've enjoyed our FaceySpacey Bible, and are coming away many times more ready to succeed at your next software startup. At the very least, you should have a birds-eye-view of what you need to do to get your startup, and have quelled a lot of insecurities you may have had regarding how you should execute it. That said, I will point to you to what you should do next.

As promised, the following is a list of the precise books I read to master web development using HTML/CSS, Javascript, PHP & MySQL. They are presented in the best order to most efficiently learn the subject at hand. It's similar to the order I read them in, but enhanced based on what I learned and the order I wish I read them in. Good luck:

HTML/CSS:

CSS Mastery: Advanced Web Standards Solutions

http://www.amazon.com/CSS-Mastery-Advanced-Standards-Solutions/dp/1430223979/

Before you start coding PHP, Javascript, etc, understand how HTML works. This is where you start. HTML is easy. Read this book in combination with studying the HTML & CSS tutorials on w3schools.

PHP & MySQL:

PHP & MySQL For Dummies, 4th Edition

http://www.amazon.com/PHP-MySQL-Dummies-Janet-Valade/dp/0470527587

This book--well an older edition--I read a year before I got serious about learning to code. I read it and didn't actually code anything i learned, but what it did was plant seeds in my head with regards to what programming is all about and what databases are all about, and how to connect the two. It assumes very little in what you may already know, and is an excellent start in your journey to becoming a master programmer.

PHP Object-Oriented Solutions

http://www.amazon.com/PHP-Object-Oriented-Solutions-David-Powers/dp/1430210117

This book is where I learned what OOP is. I didn't get the hang of it until reading the following book. Don't worry if you read this and have a hard time with it. This book and the next each have introductory chapters that go over how OOP works. It took me reading basically this book and the next book about the same stuff to get it. This book is a lot less complicated than the following and dives into practical examples & problems, whereas the next is a lot more theoretical.

PHP Objects, Patterns and Practice

http://www.amazon.com/Objects-Patterns-Practice-Experts-Source/dp/143022925X

After reading this book, I basically mastered OOP. It's a very hard book to get through if you're new to OOP, and goes into some very advanced stuff, specifically tons and tons of "design patterns." The design patterns are presented in as basic of a form as possible, but they weren't very practical like examples from the previous book in that you probably will never actually need any of the code used in the book. Either way, this is my favorite Programming of all time because it taught me how to think like a coder and how to solve complex problems with concise refactored solutions. It really showed me what is possible with OOP. You don't know PHP unless you've read this book.

Pro PHP: Patterns, Frameworks, Testing and More

http://www.amazon.com/Pro-PHP-Patterns-Frameworks-Testing/dp/1590598199

I read this book too just to solidify my experience with PHP, and cover all my bases. Check it out. It's optional.

PHP Functions Essential Reference

http://www.amazon.com/Functions-Essential-Reference-Torben-Wilson/dp/073570970X

At some point during my study of PHP I found this book and decided just to learn every PHP function available so that I could better understand the examples in the above books. Start reading this early on, and complete the whole thing. You'll quickly learn patterns in how PHP functions are named, and as a result be able to guess what a function does within the context of the examples in the above books--even if you don't remember precisely what it does.

Agile Web Application Development with Yii 1.1 and PHP5

http://www.amazon.com/Agile-Web-Application-Development-PHP5/dp/1847199585

I read this book in combination with reading the Blog Tutorial and Definitive Guide on YiiFramework.com. When you're done studying all these materials, you'll be amazed with how much power you have. This book isn't hard to read either. You'll love it if you reach this stage!

Javascript & jQuery:

Learning jQuery, Third Edition

http://www.amazon.com/Learning-jQuery-Third-Jonathan-Chaffer/dp/1849516545

jQuery is a framework built on top of the native browser language of Javascript. Usually one would recommend you learn the base language--Javascript--before learning an abstracted framework on top of it--jQuery. However because of the nature of jQuery and how comprehensive it is and because of how quirky Javascript is coming from PHP, I found it best to jump to jQuery and immediately start accomplishing the DOM manipulation tasks I needed. And ultimately because of syntax similarities between PHP and Javascript I was able to get productive in Javascript without studying a single book just on Javascript. One thing I did different when studying this book from the PHP books is I did every single tutorial as I read it. The reason is because when I learned PHP, I was learning my first real programming language--so it took me a lot of time to just digest things before I could code a single line, which is why I just read PHP book after PHP book before I got started until it all made sense. However, by the time I got to Javascript & jQuery, I understood how programming in general works and found it helpful for memorization purposes to immediately start doing the tutorials.

jQuery 1.3 with PHP

http://www.amazon.com/jQuery-1-3-PHP-Kae-Verens/dp/1847196985

With this book I didn't do all the tutorials like I did with *Learning jQuery*, but what reading this book did for me is taught me precisely how Ajax works and what it's all about. After reading it, coding features that required Ajax using Yii and PHP was obvious and a no-brainer.

JavaScript: The Good Parts

http://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742

This book gave me a deep understanding of the Javascript language and what it's truly all about. After reading it, many hours of debugging and head-scratching when coding Javascript & jQuery were removed from my schedule--because I finally learned the quirks of the Javascript language I needed to know.

Pro JavaScript Design Patterns

http://www.amazon.com/JavaScript-Design-Patterns-Recipes-Problem-Solution/dp/159059908X

Now this book took my Javascript game to the next level, gave me an idea of how jQuery was built, taught me how to do things similar to how you would in a "classical" OOP language like PHP, and completely ended any remaining head-scratching I was having with Javascript, particularly with how "scope" works in Javascript.

Linux:

The Official Ubuntu Server Book, 2nd Edition

http://www.amazon.com/Official-Ubuntu-Server-Book-2nd/dp/0137081332

Note: by the time I read this book I had already learned Linux through blogs on the internet. The best thing I can recommend you do is install Linux on your computer from the Ubuntu website, and start navigating around the command line, practicing Linux commands you learn off the web. Just google "linux tutorials" and you'll be off to a running start. That said, by the time I got proficient in Linux and after I read this book, I felt confident that I really knew what I was doing and had practical solutions for the most common problems you'll face at the command line.

Apache Cookbook: Solutions and Examples for Apache Administrators

http://www.amazon.com/Apache-Cookbook-Solutions-Examples-Administrators/dp/0596529945

This book I treat like a pocket reference and still refer to it often since it's impossible to remember all the different Apache configurations, given how comprehensive this web server application is. I did read it through when I first got it. I kinda skimmed it though--just to get an idea of what is possible. Getting an idea of what is possible without mastering a subject matter is so important in programming because you'll know where to look when you face a challenge that the subject matter can solve.

Pro Bash Programming

http://www.amazon.com/Bash-Programming-Experts-Voice-Linux/dp/1430219971

This is a little advanced for readers of the *FaceySpacey Bible*, but I'm putting it here because it really took my Linux skills to the next level.

NON-TECHNICAL BOOKS:

Smart and Gets Things Done: Joel Spolsky's Concise Guide to Finding the Best Technical Talent

http://www.amazon.com/Smart-Gets-Things-Done-Technical/dp/1590598385

If you plan to grow a small application into a large company, this book is a must. It's short. Read it.

SEO Book.com

http://www.seobook.com

This obviously isn't a book, but I read their entire site like a book, and its creator, Aaron Wall, expects you to read it like a book. When I was done reading it, I felt I was completely up to speed regarding what SEO is, how search engines work, and practical techniques to get better rankings in search engines.

For daily Startup Wisdom, checkout FaceySpacey.com/blog daily. And don't forget to download the entire No Bullshit FaceySpacey Bible or more individual chapters here: http://www.faceyspacey.com/resources?section=book .

*Thanks again from FaceySpacey and be sure to check out **FaceySpacey.com** often for further knowledge we kick to take your Startup to the Stars!*