L.A.M.P. Stack Part I

By George Beatty and Matt Frantz

This lab will cover the basic installation and some configuration of a LAMP stack on a Ubuntu virtual box. Students will download and install the necessary packages and design their own simple webpage, which can then be accessed by their partners or friends.

Introduction

While a LAMP stack is not complicated, it can be an important asset to a network. This lab will give students some light experience with LAMP, and get them comfortable with a service they may encounter in competitions.

What is a LAMP Stack?

According to Wikipedia:

***LAMP** is an archetypal model of web service solution stacks, named as an acronym of the names of its original four open-source components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language.*

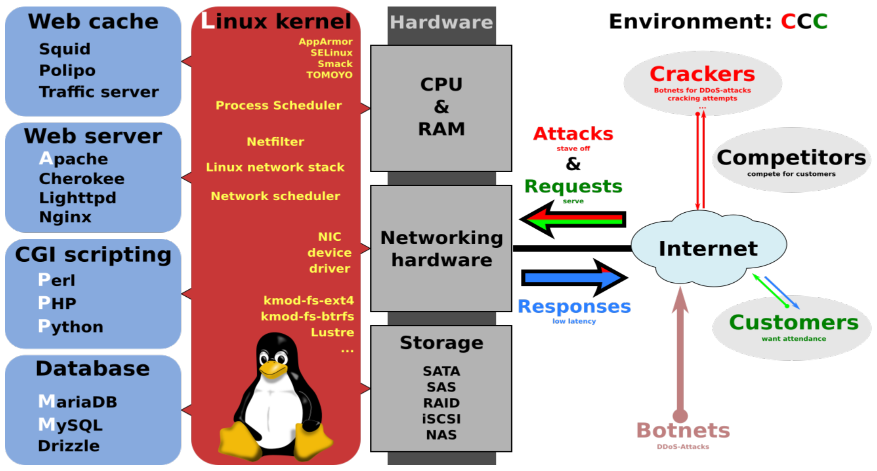


Figure 1: Overview of LAMP building blocks and overall system environment (Wikipedia.org)

If that doesn't make sense at this point, don't worry. The point of this lab is to teach you more about the individual elements and to get you more comfortable with the Linux command-line interface (CLI.) If you already have a lot of experience with the Linux CLI, then this lab will likely be fairly easy for you, and you're encouraged to help those around you.

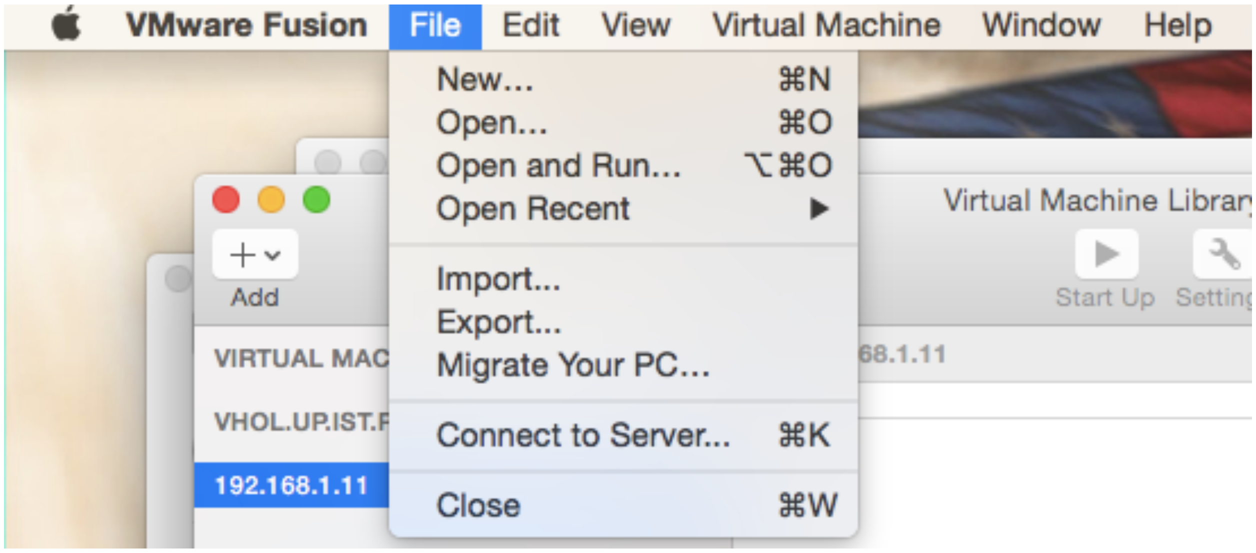
Installing the Software

The following instructions are a step-by-step guide to installing the software and getting it running. If you have questions, work with those around you, but remember the point is learning, so make sure you give an attempt to figure it out on your own first.

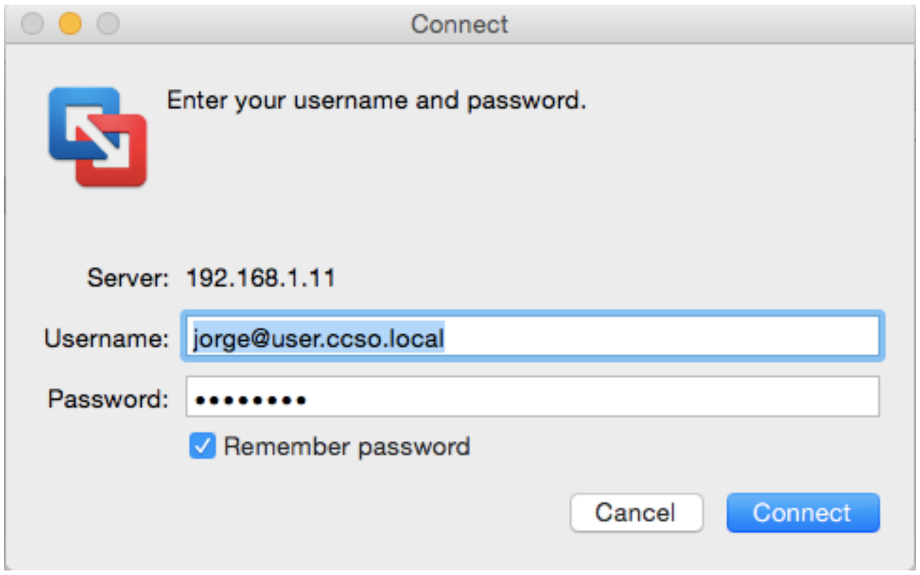
Starting the virtual machine

If you have used a virtual machine before, and are comfortable getting started, feel free to skip down to logging in.

Do: First, you will need to connect to the CCSO network and login using the username you selected and password. After that, open your virtual machine software and connect to the server. Depending on your operating system and software, it will be different, however, it should look something like this:



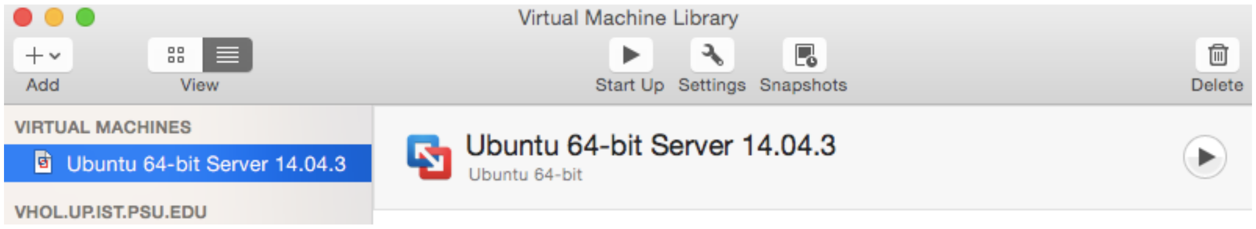
After that, you'll need to enter your username and the server info, which will look something like the screenshot below:



Enter your username and password.

Server: 192.168.1.11
Username: jorge@user.ccso.local
Password: ••••••••
Remember password

Navigate through and select an Ubuntu virtual machine and power it on. It should look something like this:



Ubuntu 64-bit Server 14.04.3
Ubuntu 64-bit

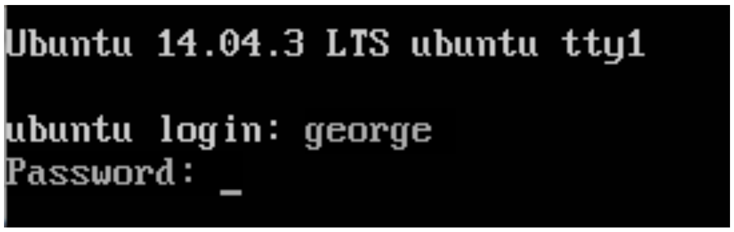
Congratulations. Your virtual machine is likely starting up.

Logging in

If your virtual machine has a GUI, it'll look a little different.

Username = vmuser

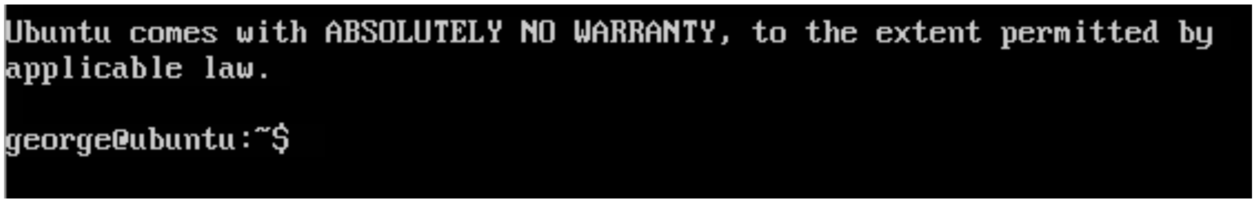
Password = vmuser



```
Ubuntu 14.04.3 LTS ubuntu tty1

ubuntu login: george
Password: _
```

Note: the password field may not populate as you type, however it is still reading your input

You successfully logged in you'll be looking at a screen that looks something like this:



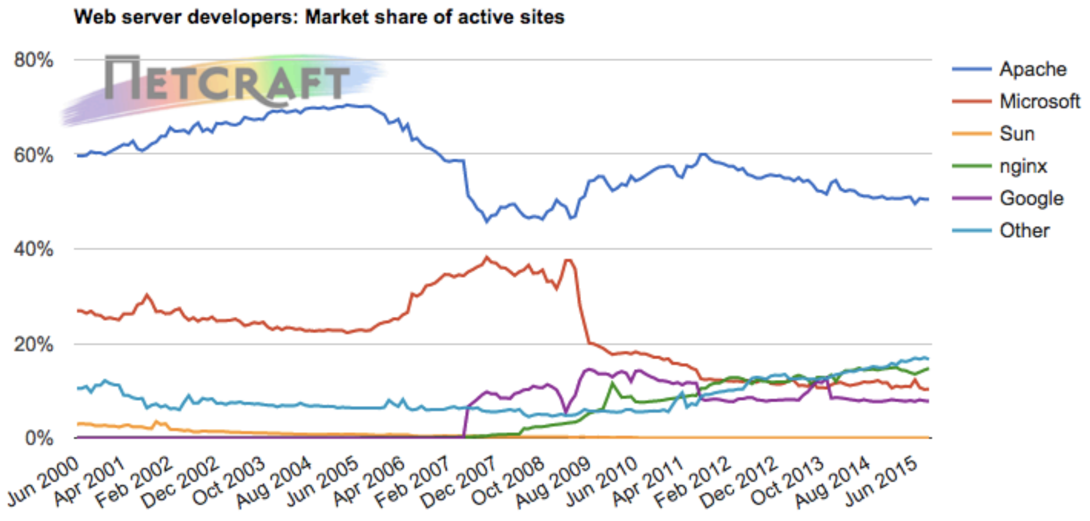
```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

george@ubuntu:~$
```

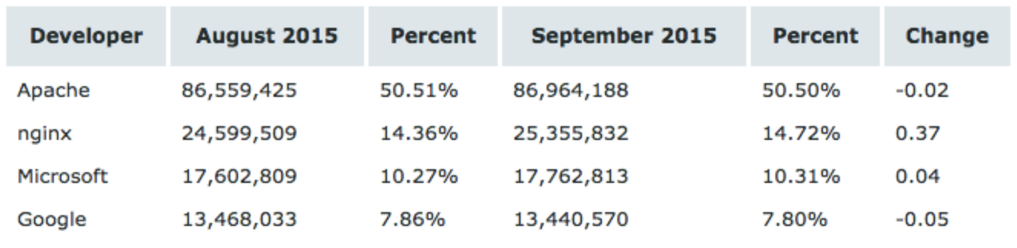
You're now logged in. Manipulating Linux is an important part of any LAMP stack; it is what the 'L' stands for, after all.

What is Apache?

You can skip to "Installing Apache" or read the next two sections for some background.

READ: If you didn't figure it out yet, the 'A' in LAMP stands for Apache. Apache is an open source HTTP server, the most widely used in the world. Chances are better than not if you deal with an HTTP server in your internship or in your career, it will be an Apache server. While its popularity has decline some over the past 10 years, it still maintains an estimated 50% share of the market of all active websites:





Developer	August 2015	Percent	September 2015	Percent	Change
Apache	86,559,425	50.51%	86,964,188	50.50%	-0.02
nginx	24,599,509	14.36%	25,355,832	14.72%	0.37
Microsoft	17,602,809	10.27%	17,762,813	10.31%	0.04
Google	13,468,033	7.86%	13,440,570	7.80%	-0.05

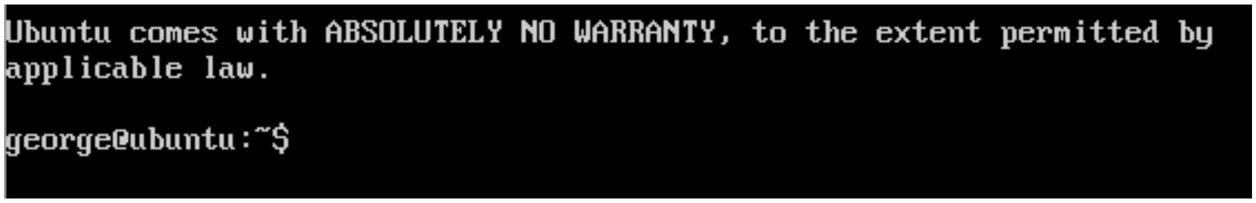
Linux and the Advanced Packaging Tool

READ: We're going to be installing Apache using the Advanced Packaging Tool (APT) that comes built in on many Linux systems, specifically most of the ones derived from Debian, which includes Ubuntu. For reference, it is estimated by www.datamation.com that almost 70% of Linux distributions are derived from Debian directly, or indirectly through Ubuntu.

The most popular alternative is RPM Package Manager, (originally Redhat Package Manager, now a recursive acronym.) RPM is used by Redhat, Fedora and CentOS distributions and is often manipulated by the wrapper Yellowdog Updater, Modified (YUM). These powerful tools allow for easy install of software, while also allowing a user to only keep what they need on a system.

Installing Apache

READ: If your system has GUI, you'll need to go to applications and open up Terminal. If it doesn't you'll already be there. Either way you should be looking at a screen like this:

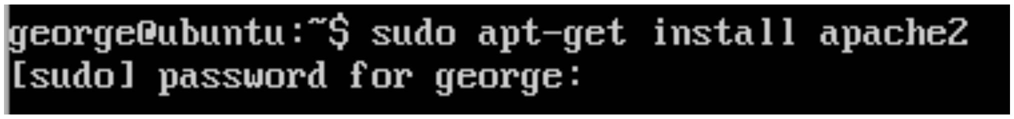


```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

george@ubuntu:~$
```

DO: To install Apache we want to type out the command below and then hit return:

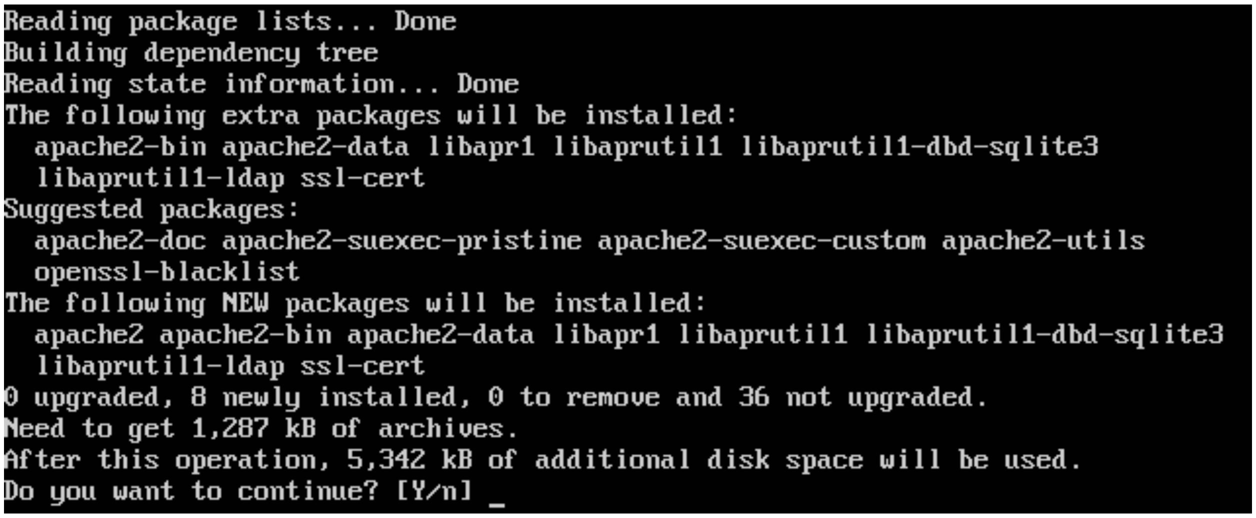
sudo apt-get install apache2

READ: "Sudo" is a "Super user do" command (pronounced sue-dough) which allows us to use commands not available to the average user. Only root users or users in the /etc/sudoers file can use this command, and you should be prompted for a password. Using sudo as opposed to logging in as root is considered much safer, and runs less risk of inadvertently screwing something up. You'll be looking at a screen like this:



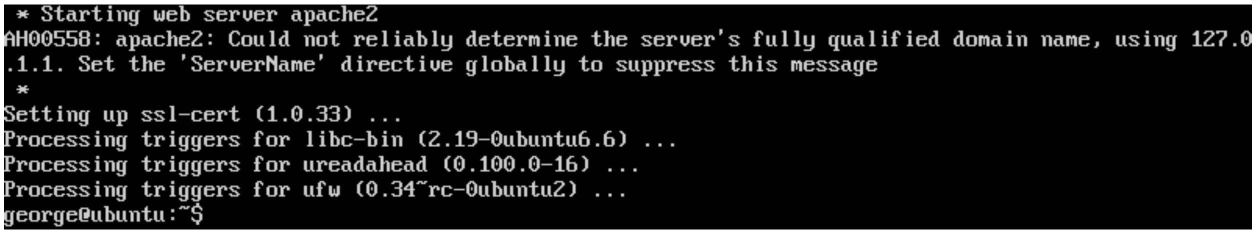
```
george@ubuntu:~$ sudo apt-get install apache2
[sudo] password for george:
```

DO: After that you'll see the system process some stuff and prompt you with how much space the install is going to use. You're going to hit 'y' for yes:



```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom apache2-utils
  openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap ssl-cert
0 upgraded, 8 newly installed, 0 to remove and 36 not upgraded.
Need to get 1,287 kB of archives.
After this operation, 5,342 kB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

After that you'll see Apache downloading and installing and then a message that Apache has been successfully installed.



```
 * Starting web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0
.1.1. Set the 'ServerName' directive globally to suppress this message
 *
Setting up ssl-cert (1.0.33) ...
Processing triggers for libc-bin (2.19-0ubuntu6.6) ...
Processing triggers for ureadahead (0.100.0-16) ...
Processing triggers for ufw (0.34~rc-0ubuntu2) ...
george@ubuntu:~$
```

Congratulations. Apache is now installed. Yep. That's it.

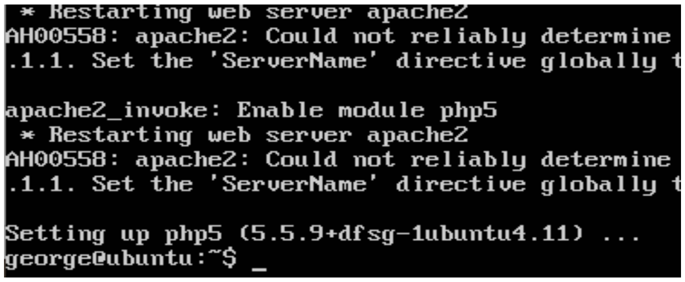
Installing MySQL & PHP

MySQL and PHP will be installed in a very similar way to Apache, so you should be able to figure it out. For MySQL its:

sudo apt-get install mysql-server

You may be prompted to set a password for MySQL, make it the same as the password you used to login. And for PHP it's:

sudo apt-get install php5 libapache2-mod-php5



```
 * Restarting web server apache2
AH00558: apache2: Could not reliably determine
.1.1. Set the 'ServerName' directive globally t

apache2_invoke: Enable module php5
 * Restarting web server apache2
AH00558: apache2: Could not reliably determine
.1.1. Set the 'ServerName' directive globally t

Setting up php5 (5.5.9+dfsg-1ubuntu4.11) ...
george@ubuntu:~$ _
```

Now for some, Apache may have just restarted, but lets go ahead and manually restart it one more time:

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

Checking Apache

Do: Next, we want to see if our webserver is running. If you have a GUI, open a up a web browser and go to http://127.0.0.1:80 or http://localhost. Either should bring you to a page confirming that the server is running and Apache is functioning.

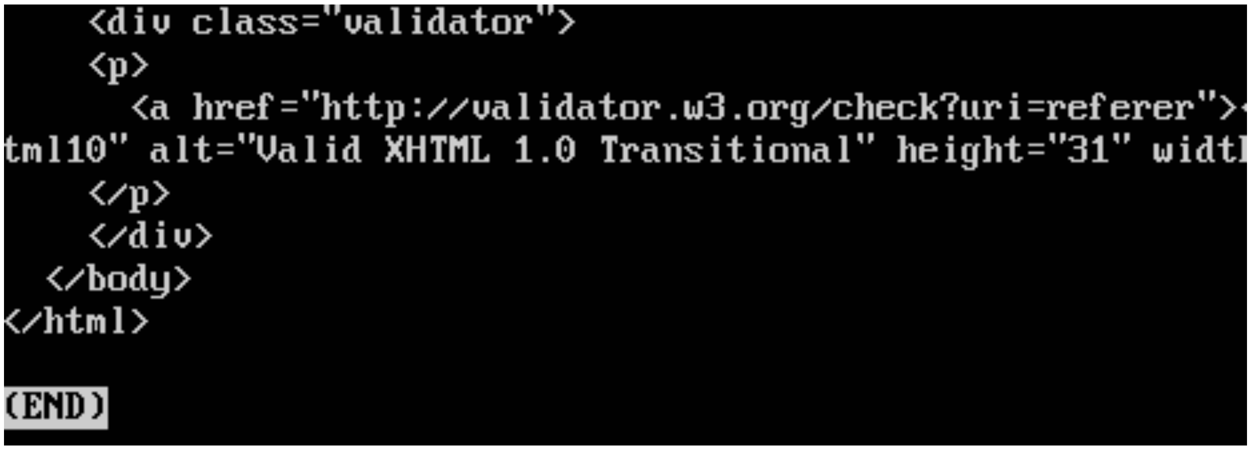
If you don't have a GUI you can download index.html using "wget":

```
wget http://localhost:80
```

This will download index.html to your current directory, which can be opened using the "less":

```
less index.html
```

This will print the HTML file in plaintext and allow you to page through it. After you reach the end hit 'q' to quit.



```
    <div class="validator">
    <p>
      <a href="http://validator.w3.org/check?uri=referer">
tml10" alt="Valid XHTML 1.0 Transitional" height="31" widt
    </p>
    </div>
  </body>
</html>

(END)
```

Using Lynx to Check Apache

If you're interested in learning to use Lynx to view webpages in the CLI, and to check Apache is working correctly, do this step. If not skip to the next.

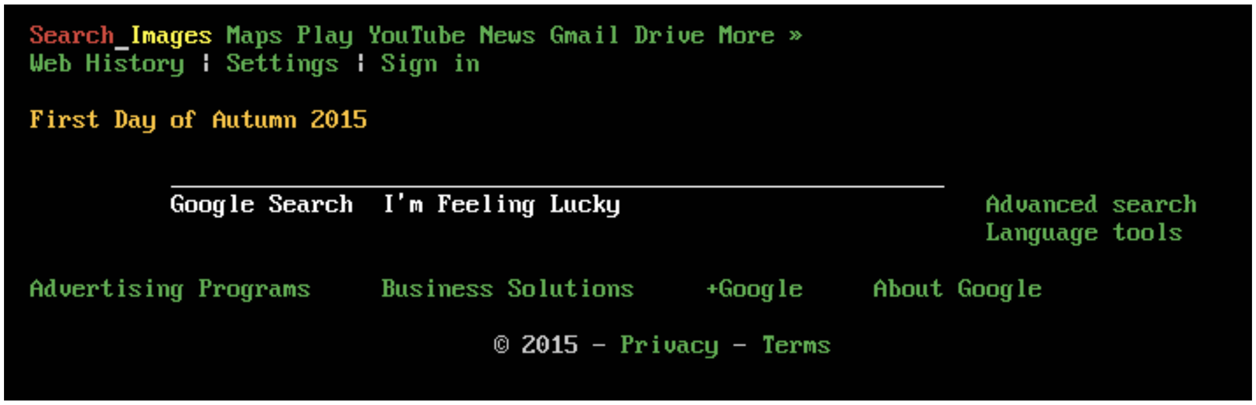
Do: The other option, which can be done in terminal in either case, is to use Lynx. Lynx allows us to view websites in either terminal or the CLI. Enter the command:

```
sudo apt-get install lynx
```

And install it like the other packages we've used. After that you should be able to browse a website using lynx.

```
lynx {insert URL without brackets}
```

Try visiting our Apache server at http://localhost and then try to visit http://www.google.com and search for something.



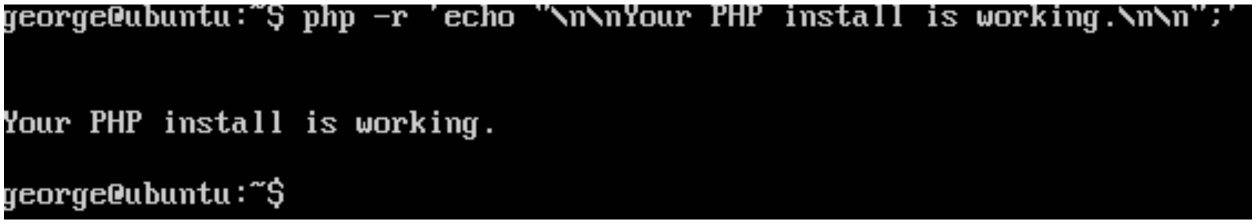
```
Search_Images Maps Play YouTube News Gmail Drive More »
Web History | Settings | Sign in

First Day of Autumn 2015

        ________________________________________
        Google Search  I'm Feeling Lucky                    Advanced search
                                                            Language tools

Advertising Programs     Business Solutions     +Google     About Google

                         © 2015 - Privacy - Terms
```

Checking PHP

Do: To check PHP, run a simple script in the CLI or run a PHP file from within /var/www/. If you don't know PHP or how to run a script in Linux you can run this script, or use it as a model.

```
php -r 'echo "\n\nYour PHP installation is working
fine.\n\n\n";'
```

Which should print out:



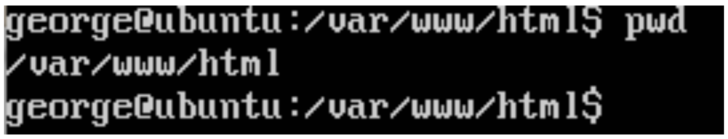
```
george@ubuntu:~$ php -r 'echo "\n\nYour PHP install is working.\n\n";'


Your PHP install is working.

george@ubuntu:~$
```

So now we've installed all the elements, and ensured they are working.

Navigating directories and deleting a file

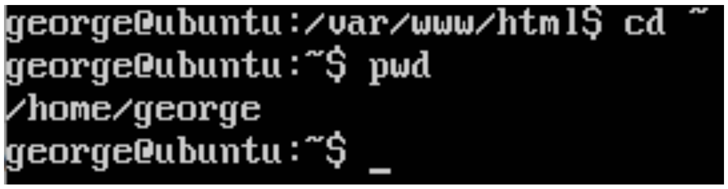
If you already know basic commands to navigate skip to the next step.

Do: Creating a file in Linux is very easy. We're going to navigate to our home directory if we aren't already there. To check our current directory, enter the "pwd" command:



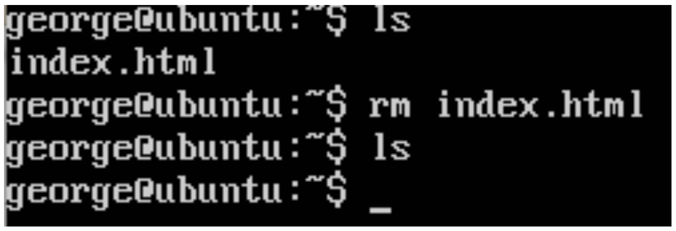
```
george@ubuntu:/var/www/html$ pwd
/var/www/html
george@ubuntu:/var/www/html$
```

DO: If you see /home/username you're likely in your home directory. However we don't want to be in the directory above so use the "cd ~" command to change to the home directory:



```
george@ubuntu:/var/www/html$ cd ~
george@ubuntu:~$ pwd
/home/george
george@ubuntu:~$ _
```

Next use "ls" (that's a lowercase 'L') to list the files in your home directory:



```
george@ubuntu:~$ ls
index.html
george@ubuntu:~$ rm index.html
george@ubuntu:~$ ls
george@ubuntu:~$ _
```

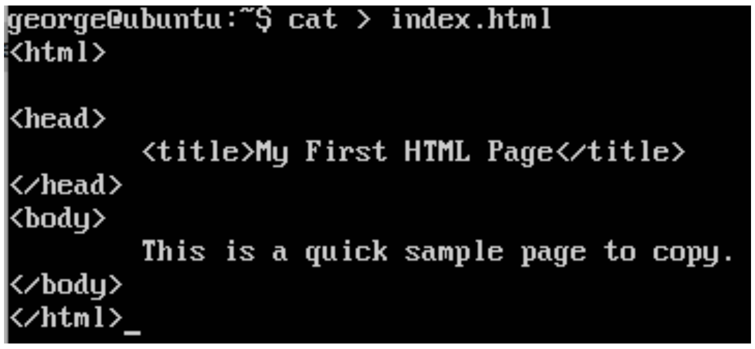
If you see index.html in your home directory because of using wget earlier, use "rm" to remove it as you see above. After, use ls again to check that it was deleted.

Creating a new index.html

DO: To create a new index.html file, we need to use "cat" like this:

```
cat > index.html
```

READ: Now we can type what we want to be in our index.html file similar to how we could in Notepad or TextEdit. Return will only move us to the next line; we need to use ctrl+c to exit the file. If you are comfortable with HTML feel free to put whatever you like in the document, if not plaintext can work. However, it may be best if you work with the person next to you or copy the screenshot.

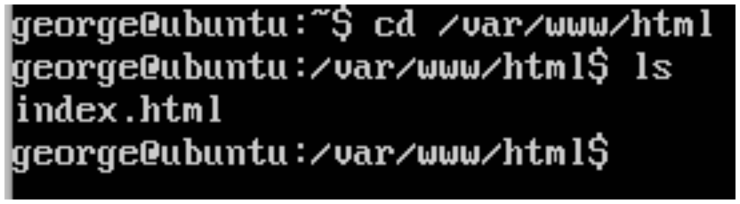

```
george@ubuntu:~$ cat > index.html
<html>

<head>
        <title>My First HTML Page</title>
</head>
<body>
        This is a quick sample page to copy.
</body>
</html>_
```

Replacing the default index.html

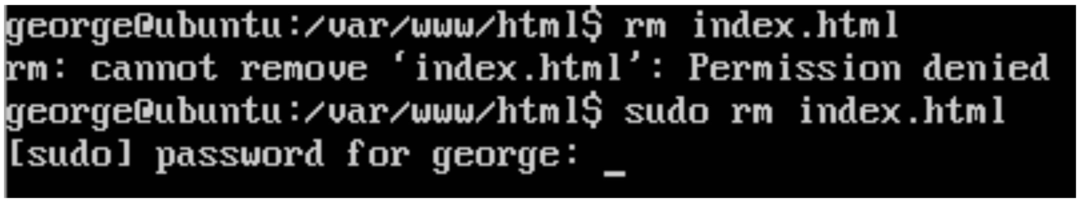
Do: Apache knows what website to show based on what directory its pointed to in its settings. By default that's /var/www/html/index.html. We want to replace that html file with ours so we need to navigate to that directory:

cd /var/www/html

Then list the files; to make sure index.html is there:


```
george@ubuntu:~$ cd /var/www/html
george@ubuntu:/var/www/html$ ls
index.html
george@ubuntu:/var/www/html$
```

Do: We can use "rm" and the filename to delete the file, but if we do that we'll get an error:


```
george@ubuntu:/var/www/html$ rm index.html
rm: cannot remove 'index.html': Permission denied
george@ubuntu:/var/www/html$ sudo rm index.html
[sudo] password for george: _
```

This is because our account doesn't normally have the permission to do that; we need to use sudo to show we are allowed to do that action, like we did earlier.

```
sudo rm index.html
```

Then enter your password. The file should be deleted so quickly use ls to check. Next, we need to use "mv" to move our index.html to the directory Apache is pointed towards. Since the directory is protected, we'll also need to use sudo for this.

```
sudo mv ~/index.html /var/www/html
```

That should do it. You can use Lynx to view your page. Also, if you share you IP address with the people around you they should be able to view it too.