Project 6: Building a LAMP Server with Raspberry Pi

This lesson is adapted from Jonathan Martin's Creating LAMP Infrastructure for DH Projects DHSI 2016 course. Available at https://dhlinux.org/wiki/index.php?title=Main_Page.

Project 6 Objectives:

By the end of this project you will be able to:

- Work at command line to configure the Firewall
- Automate the updates and upgrades on your Pi
- Install and configure Apache, MySQL, and PHP on your webserver
- Explain the function of a web server in delivering content to the WWW

Working at the Command Line

We worked a bit at the command line during the end of the OS installation and Pi configuration. For this next project we will work almost exclusively at the command line. Remember that all of the functions that you see at the GUI (Graphical User Interface) can be performed at the Command Line. While user-friendly and prettier, the GUI has limitations – you can only perform the functions that have been programmed into this interface. The Command Line is the pre-cursor to the windows-style GUI. While it is a bit less dynamic and not immediately intuitive, you have much more power working at the Command Line. As you perform the various tasks, you will see that every aspect of computing can be reduced to a few commands in text form.

To get started, open up a Terminal session on your Pi. What follows is a step by step guide to building a LAMP server. All you have to do is follow the steps, write down what you did in your lab notebook, and record your questions and observations. If you get stuck, turn to your rubber duck, to the instructor, to your classmates, or to the Internet for help.

> ⓘ Server: A server is simply a computer that is designed to provide a service to another computer ("client"). A file server is dedicated to storing and providing access to files, a print server provides printing for many computers, and a mail server handles email within a network. You engage with servers every day when you make a request to view a particular website. The files for that website are hosted on a server. When you type the URL into the search bar or click on a URL from search engine results, your computer (the client) sends a request to the server to see that information.

Task 1: Setting up the Firewall

Before we start, we must make sure that our OS is up-to-date. Security releases and updates are released frequently, so it is good practice to update your OS on a regular basis. You have probably been prompted frequently with these updates on your PCs, Macs, and mobile devices. Our Linux-based OS doesn't come built in with these reminders, so we have to run this set of commands by ourselves. We will automate this is a bit. Run `sudo apt-get update && sudo apt-get upgrade`.

🛠 && is the AND operator for commands. The second command will be executed if and only if the first is executed and exited successfully. Note that you must type sudo for the second command as well or it will not be executed.

ⓘ Firewall: The server's job is to respond to requests from the outside world and provide access to the information on the server. However, we want to make sure that we can control who has access to the server to prevent attacks, viruses, and unauthorized users from gaining access to the server. As you will see, the firewall is just a set of rules that defines how other computers can connect to the server. (see pages 384-385 of this week's reading for more details).

Uncomplicated Firewall (UFW)

1. Run `sudo apt-get install gufw` to install Uncomplicated Firewall (UFW) with the graphical user interface.

🛠 An Introduction to GUFW
http://www.dedoimedo.com/computers/gufw.html

An Introduction to Uncomplicated Firewall
https://www.linux.com/learn/introduction-uncomplicated-firewall-ufw

Uncomplicated Firewall How To Guide
https://www.digitalocean.com/community/tutorials/how-to-setup-a-firewall-with-ufw-on-an-ubuntu-and-debian-cloud-server

UFW Help Guide https://help.ubuntu.com/community/UFW

🛠 To remove installed software run the command `apt-get remove` *`software name`*

2. Open firewall with the command `gufw`
3. This command will open the Graphical User Interface for the Uncomplicated Firewall. You can also configure the firewall at the command line. I'll include both sets of instructions over the next few steps. You can only work at one interface or the other, but not both. Pick either the GUI or CLI to follow over the next few steps.
4. First turn the firewall on:
    - GUI: Click the Unlock button at the bottom right of the open window; enter your password if prompted; Use the slider to change status to ON
    - CLI: `sudo ufw enable`

** If you receive an error about the ipv6 tables, run the command `sudo nano /etc/modules`. When the file opens, add the line `ipv6` to the document. Hit control-X to exit, Y to save. Then reboot your pi with `sudo reboot`.

5. Deny all incoming connections:
    - GUI: From the Incoming dropdown, select Deny
    - CLI: `sudo ufw default deny incoming`
6. Allow all outgoing connections:
    - GUI: From the Outgoing dropdown, select Allow
    - CLI: `sudo ufw default allow outgoing`

> **Q1:** In your notebook explain this set of commands (steps 4-6) in your own words.

7. While these rules provide protection from incoming attacks, we do need to allow some two-way communication in order to provide some services on our servers. We can do this by setting some basic rules.
8. First allow access on Port 80:
    - GUI: Click on the + in the bottom left corner of the firewall window; in the select the following from the dropdown boxes Allow In Service HTTP
    - CLI: `sudo ufw allow www`

> ⓘ Port: Your computer has physical ports and logical ports. Physical ports are where you physically attach an accessory (keyboard, mouse, monitor) to exchange information with the computer. When we talk about ports on the server, we are talking about logical ports. These are designated connecting "places" that are defined by specific Internet Protocols. See Port 80, below as an example.

> ⓘ Protocol: Very simply, protocols are agreed upon rules governing how computers connect to each other and how information is exchanged. A protocol defines how the information is formatted into packets and exchanged between computers.

> ⓘ Port 80: Is the logical port designated for web-based requests or requests to access the webpages on the server. Port 80 uses the Hypertext Transfer Protocol or HTTP. This should look familiar. This is the protocol that governs the way that information is exchanged between computers on the world wide web. For technical details see http://fab.cba.mit.edu/classes/961.04/people/neil/ip.pdf.

Q2: Where do you commonly see HTTP? Why?

9. If you are working at the command line, check the firewall status with `sudo ufw status` to check to see if your rules have been enabled. If you are working at the GUI, close the firewall window. You may also check the status of the firewall using this command in the Terminal.
10. You have now established a basic firewall. There are many more rules that we could set to control the exchange of information over your server, you can review these rules in the documentation for UFW.

Q3: Now we have a basic firewall established. Can you explain what the firewall is doing in your own words? Would you set any additional rules? Use the TCP/IP Ports and Protocols chart to review common ports and services http://www.pearsonitcertification.com/articles/article.aspx?p=1868080 You may also wish to set additional rules on your firewall using this chart.

Task 2: Automating your Updates and Upgrades

We were sure to update our software before we started this exercise. Ideally, we would want to update the OS on a regular basis. The easiest way to do this is to automate the updates. So, before we install the LAMP software, let's automate this process.

1. First install the unattended-upgrades software with the command `sudo apt-get install unattended-upgrades`
2. Next, we have to write a few commands in the 10periodic file. Open the file in nano with the following command `sudo nano /etc/apt/apt.conf.d/10periodic` and write the following in the blank file:
```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
APT::Periodic::Unattended-Upgrade "1";
```
This set of commands updates the package lists (the new updates available), downloads, and installs the available upgrades every day (designated with a "1"). The AutocleanInterval cleans up your downloads folder once a week (a 7 day interval).
3. <Ctrl> X to exit; Y to save; <enter> to save with the same file name
4. Now we are going to configure which upgrades we would like to install. Open the 50unattended-upgrades file `sudo nano /etc/apt/apt.conf.d/50unattended-upgrades`
5. When you open the file you should see quite a bit of text. Note that here the commented text is preceded by a // instead of a #. We'll see different forms of

commenting throughout the term. Remember that commented text is not read by the computer as a command. This is the explanation of the code that is written there for us.

6. Scroll down to find the uncommented line `Unattended-Upgrade::Origins-Pattern{`, We want to make a few edits here. Uncomment or add the following lines in this section:

```
"o=Raspbian,n=jessie";
"o=Raspbian,n=jessie,1=Raspbian-Security";
```

7. <Ctrl> X to exit; Y to save; <enter> to save with the same file name.
8. Now each day the Pi will automatically check for and install any updates to the OS, including security updates.

Installing the LAMP Stack

What is a LAMP stack? LAMP is very simply a bundle of software that are used together to build dynamic websites and host them on servers. You'll see later this term that many popular platforms like WordPress and Omeka require a LAMP stack. LAMP stands for Linux, Apache, My SQL, and PHP/Python/ Pearl. We already have the L (Linux) with our Raspbian OS (remember this is a flavor of Linux based on the Debian OS). Before we get started, we are going to install a simple web browser that we can use at the command line to check our configuration as we install each piece of software.

Task 3: Install Lynx

📖 What is Lynx? https://en.wikipedia.org/wiki/Lynx_(web_browser)

1. Install Lynx `sudo apt-get install lynx`
2. With Lynx installed, use the command `lynx` followed by a web address to navigate the web. Try navigating to a few of your favorite websites.

Q4: Describe the differences between viewing websites in Lynx and through Firefox/Chrome/Safari/your favorite web browser.

3. When finished type q to quit

Task 4: A is for Apache

What is Apache? Apache is the Web server. This is the software package that allows your computer to offer services to the web. Apache2, and other web servers, use HTTPD or Hypertext Transfer Protocol daemon. This is the program that sits and waits for HTTP requests from other computers and then responds to these requests.

ⓘHTTPD: Remember, HTTP is the (Hypertext Transfer Protocol) or the set of rules that allow for the transfer of files (text, images, sound, etc.) on the World Wide Web. D stands for daemon, defined below.

ⓘdaemon: a program that runs continuously and exists for the purpose of handling service requests that a computer system expects to receive. The daemon program forwards the requests to other programs (or processes) as appropriate.[1] In this case Apache is the HTTP daemon. Other daemons handle e-mail and printing requests, for example. See also http://www.linfo.org/daemon.html.

1. Let's install Apache `sudo apt-get install apache2 -y`

   Q5: What does the –y mean in this command? Use Explain Shell http://explainshell.com to find out.

2. Many applications will use Apache's Mod_Rewrite capabilities. This function allows you to create human-readable URLs (i.e. gizmo.com/latest_and_greatest/specific_gadgets/exactly_what_youre_looking_for VS. http://www.gizmo.com/gp/itemB004RYVI0Q/ref=as_li_ss_tl?) More on this later. Activate this rewrite function with the command `sudo a2enmod rewrite`
3. Restart using the requested command. Remember to use sudo
4. Now we are going to take a look at our hosts file `sudo nano /etc/hosts`
   a. When you open this file you should see something like this:

```
127.0.0.1       localhost
::1             localhost ip6-local host ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

127.0.1.1       raspberrypi
```

   b. The hosts file maps your computer to the network, that is, it defines the host name (the human readable name of the computer) and the IP address.

ⓘIP address: An identifier for a computer or device on a TCP/IP network. Networks using the TCP/IP protocol route messages based on the IP address of the destination. The format of an IP address is a 32-bit numeric address written as four numbers separated by periods. Each number can be zero to 255.[2]

[1] daemon, Seaarch SOA http://searchsoa.techtarget.com/definition/daemon
[2] White, *How Computers Work*, 312.

Q6: The definition above states that the IP address is a 32-bit address, explain what this means in your own words (think back to the binary lesson from a few weeks ago).

ⓘ ip6 or IPv6: The 32-bit IP addresses are based on the IPv4 or version 4 of the IP protocol. On the Internet, each computer is assigned a unique IP address, however, the 32-bit address scheme from IPv4 allows for just over 4 billion addresses. With the growing number of devices connected to the web, we have run out of IP addresses. IPv6 is a 128-bit IP address. In the example above, the ip6 address begins with an ff, this should look familiar as some of the digits are represented in hexadecimal rather than binary. A full IPv6 address could be written like this 3ffe:1900:4545:3:200:f8ff:fe21:67cf.

Q7: In your own words, explain why a 32-bit address scheme provides just over 4 billion addresses. How many unique addresses does a 128-bit scheme provide for?

c. "localhost" always refers to your local computer. This is your Pi. "raspberrypi" also refers to your computer. Remember that at the command line you are pi@raspberrypi, or the user Pi on the computer raspberrypi. The localhost IP address is always 127.0.0.1. The "loopback" that you see referenced in the file refers to the fact that this IP address will always loop back you your computer. We cannot access each other's Pi's using the local host. Instead we would need to define a unique IP address in this file. We won't worry about this at the moment.

5. We will make some modifications to this file later. For now, just <control> X to exit. N to discard any changes if prompted.
6. Let's test the Apache installation using lynx and our local host. At the prompt type, `lynx localhost`. You should see the Apache2 Debian Default Page.

Q8: Open the page in the web browser by using localhost as the URL. Now try the IP address 127.0.1.1 and http://raspberrypi. What happens? Why?

7. Now we are going to configure the server using the .conf file aka the configuration file.
   a. At the command line use sudo nano to open `/etc/apache2/apache2.conf`

b. This file configures the behavior of our server. The commented lines explain the function of the commands that follow. Remember that if we were to set a static IP address (different from 127.0.0.1) and allow web traffic to find our page we would have other computers accessing ours. These settings control how our server reacts when handling requests from multiple machines.
c. We are going to add a few settings that would ensure that our little machine is not overloaded by requests on the web. Scroll down to the bottom of the file and add a line of blank space, then:

!! Be careful with the "white" or blank space here

```
# User Customization
     <IfModule mpm_prefork_module>
     StartServers 2
     MinSpareServers 6
     MaxSpareServers 12
     MaxClients 30
     MaxRequestsPerChild 3000
     </IfModule>
```

d. Save the file
e. Then restart your server `sudo service apache2 restart`

> ⓘ What's going on here? We have a limited amount of memory and processing power on our machines (remember a server is just a computer), so we need to control the traffic on our machine. While the firewall controls access to the machine, this conf file and the piece of code that we added controls the way the machine acts when it receives requests – that is this file defines how your server performs when receiving multiple requests. It's like tuning an engine. For details on what we have just added see https://www.linode.com/docs/websites/apache-tips-and-tricks/tuning-your-apache-server and http://httpd.apache.org/docs/2.2/mod/mpm_common.html#startservers.

Task 5: Setup a Virtual Host

Virtual hosting is a method for hosing multiple domain names on a server. A domain name is the human readable designation for the IP address. For example, a few minutes ago we used "localhost" and "raspberrypi" to resolve to the IP address 127.0.1.1. On the World Wide Web, you must first register a domain name as a place holder because these particular configurations of letters and numbers have been purchased. The conventions for naming domains are dictated by the Domain Name System. See https://www.w3.org/wiki/Getting_your_content_online for more details. A Domain Name Server is like a phonebook, these servers translate the domain name to the unique IP address for your server. So, for example, I purchased the Domain Name

lindsaymattock.net from Google Domains. I pay Google $12 a year to have access to this domain. When you type lindsaymattock.net into your Web Server, a request is sent to Google's Domain Name Servers requesting the IP address that is associated with lindsaymattock.net. The server returns 128.255.54.114 (the IP address for the computer that is running as a server in my office) and connects to my server at that IP address. You can test this by navigating to lindsaymattock.net and then to the IP address 128.255.54.114. If you know the IP address for the machine, then you can type it in a visit it. The Domain Names allow us to use more memorable representations of the domain names so that we don't have to remember that string of numbers.

1. Our server will not be available to the wider WWW, so we don't have to worry about purchasing a domain name or paying for hosting. However, using the localhost IP address we can rename our host. We have already established that "raspberrypi" is the default host. Let's add a host mysite.dev. (You can use any name that you would like here. I suggest using the .dev extension rather than .com, .org, .net, etc so that your web browser does not attempt to connect to already existing sites out there in the world.)
    a. Let's reopen the hosts file `sudo nano /etc/hosts`
    b. We will have to append the file a bit. The entire block of 127.0.0.0 addresses (127.0.0.1 through 127.255.255.255) have been reserved for loopback to local hosts, meaning that we can assign different domain names to different IP localhost IP addresses. So, let's add mysite.dev to our hosts file.
    c. Below the raspberrypi line, type `127.0.1.2 mysite.dev`
    d. exit and save
    e. Now, let's check to make sure that your new domain name works `lynx mysite.dev`. You should see the default Apache page.
    f. The web browser will try to resolve to the www, so make sure to use the full URL http://mysite.dev/ when testing your domain name in the GUI web browser.
2. At the moment, the system is defaulting to Apache's default web page. We will want to add our own content, but we need a place to do this.
    a. At the command prompt, navigate to the www folder `cd /var/www/html`
    b. Now `ls` to see the contents. You should see an index.html file
    c. use `nano` to open this file

> **Q9:** Can you describe what you are looking at here? Hint: Take a look at the information between the <title> tags.

    d. What you see here is the html file containing the Apache Default page. This is the file that opens when you request to see mysite.dev. Exit nano.
    e. We are going to create a new folder to hold our web files. First navigate back to the www folder using the command `cd ..`

Q10: What does this command do?

f. You should now be back in the /var/www folder. You can confirm this by looking at the command prompt. You should see "pi@raspberrypi:/var/www$".
g. Now we are going to create a new directory with the name of our site using the command `sudo mkdir mysite.dev`
h. Now change the directory to your new folder `cd mysite.dev`
i. We are going to make another directory that will hold all of the documents that we would like to share with the public on our site. `sudo mkdir public_html`
j. Now change the directory to your new folder `cd public_html`
k. Use the print working directory command to see the current directory by typing `pwd`. The command should return "/var/www/mysite.dev/public_html"
l. The last step we need to take here is to create a simple HTML page so that we have something to test in a few steps: `sudo nano index.html`
m. In this new file, enter this little bit of html (you can type whatever text you would like between the <body> tags):

```
<html>
  <body>Welcome to mysite.dev</body>
</html>
```

n. Save and close the file. We can update this later with your files from the HTML project. Don't worry about it for now.

3. Open mysite.dev in lynx. You'll still see the Apache Default page. We have to tell new site that we would like to make available to the web. We do this by creating a new configuration file for our website. We'll do this in the Apache sites-available file.
   a. Open a new file with the command `sudo nano /etc/apache2/sites-available/mysite.conf`
   b. In the new blank file, add the following:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName mysite.dev
    ServerAlias www.mysite.dev
    DocumentRoot /var/www/mysite.dev/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

   c. Here we are creating the virtual host for mysite.dev. We have the name of the server – mysite.dev instead of raspberrypi (the name of your computer). The alias is the URL www.mysite.dev. This is what the IP address resolves to. The Document Root tells Apache where the files for our site live. In this case it's in the new directory that we just created.

d. Now we need to enable the site using the command `sudo a2ensite mysite.conf`
e. Restart Apache in the manner requested. Remember to use sudo.
f. Open mysite.dev in lynx. You should now see the text from the html file that you created a few steps ago.

Task 6: P is for PHP

P in this case is for PHP. As indicated earlier the P can also stand for Python or Pearl. These are all programming languages. We'll use Python in a few weeks to learn some programming basics. Wordpress, Omeka and other dynamic webpage platforms use PHP, so we'll install it now.

1. Install php using the apt-get command `sudo apt-get install php7.0 libapache2-mod-php7.0 -y`
2. Open the php initialization file `sudo nano /etc/php/7.0/apache2/php.ini`
   a. Check to ensure that the following configuration defaults are correct. You may need to uncomment and modify some lines (you will have to scroll through the file, these lines to no appear together):

```
max_execution_time = 30
memory_limit = 128M
display_errors = Off
log_errors = On
error_log = /var/log/php/error.log
register_globals = Off (May not exist. That's OK)
```

> 🛠 Use <control> W to find text within nano. Repeat the command and press <enter> to find multiple instances of the same string.

> ⓘ The lines that being with a ";" are the commented lines.

   b. Exit and save any changes.
   c. You may have had to enter the error_log location. Check to make sure that it exists using the cd command `cd /var/log/php`
   d. It's ok if you receive an error. Try to resolve this one on your own: navigate to the /var/log/ directory and use `mkdir` to create a php directory and `touch` to create an error.log file in the newly created php directory. For help with the commands search the web, or use explainshell.com, or your classmates for help.
   e. Now restart the server using `service apache2 restart`
3. Now, let's test the PHP install to ensure that it is working properly
   a. Create a new file called index.php in the /var/www/mysite.dev/public_html folder

b. Enter the following php code into the blank file

```
<?php
        echo '<h1>Hello World</h1>';
?>
```

✚ Do you want to try some dynamic PHP? Add the line `<?php echo date('Y-m-d H:i:s'); ?>` to the index.php file. What do you think this command will do?

c. You have just created a basic php file. If you look at the contents of the public_html folder, you should now have an index.html and index.php file in this folder.
d. Now open http://mysite.dev/index.php in a browser. You should see "Hello World" on the screen.

Q11: Here we have called mysite.dev/index.php in the browser. Describe in your own words what we are doing with this command.

e. If you just navigate to mysite.dev you will see the html file that we created earlier. This is because Apache is defaulting to html. We have to tell the server to default to php by changing the settings. We don't need to worry about this now.

Task 7: M is for MySQL

So we've installed the L, A, and P, MySQL (pronounced *My Sequel*) is the final step in creating our LAMP stack. MySQL is an open source database management system. Many blogs and other content management systems used relational databases or organize data such as blog posts. MySQL is used for this purpose.

📖 What is MySQL? What is a Database? What is SQL?
http://www.thesitewizard.com/faqs/what-is-mysql-database.shtml

1. Start with `sudo apt-get install mysql-server php7.0-mysql -y`
2. When the install is complete restart apache `sudo service apache2 restart`

3. We are going to take one more step to secure our MySQL databases. You will be storing quite a bit of sensitive information here for your webpage and we want to ensure that you don't get hacked!
    a. Run the command `sudo mysql_secure_installation`
    b. Hit enter when prompted for a password. We haven't completed this step yet.
    c. When prompted to change the root password type Y. Enter a password for the root user. Be sure to write this down so that you don't forget it. NOTE: YOUR CURSOR WILL NOT MOVE AS YOU TYPE.

> ✳ Did you forget your password? Use these instructions: https://support.rackspace.com/how-to/mysql-resetting-a-lost-mysql-root-password/

> ✳ NOTE: the default answer is in caps, in this case Y. To choose the default press <Enter>.

    d. Remove anonymous users? Y
    e. Disallow root login remotely? Y
    f. Remove test database and access to it? Y
    g. Reload privilege tables now? Y
    h. Your MySQL installation is now secure

4. Next, we are going to create a database and grant other users permission to use the databases.
    a. Log in using the command `mysql -u root -p;` With this command you are requesting to login as the root user with a password
    b. When prompted enter your mysql password
    c. You will now see the mysql prompt `mysql>` . This means that we are now working within MySQL and not the Bash Shell.

    d. Now we are going to create a new database and a new user that can have access to that database.

> ✳ NOTE: the semicolons (;) at the end of each of the following lines are crucial for ending the commands. If you press <enter> before typing the ; the command will not execute. Simply type the ; on the next line to execute the command.

    e. First create a new database called cf2017: `create database cf2017;`

f. Now you'll create a new user of the cf2017 called cf_user that is identified by a new password. In this case substitute a password for the word password: `grant all on cf2017.* to 'cf_user' identified by 'password';`
g. You have now configured MySQL and can pass on these database credentials to other users of your database. To exit use the command `quit`.

5. Our last step is to install phpMyAdmin. This is a GUI interface that can be used for MySQL administration.
   a. Install using the command `sudo apt-get install phpmyadmin -y`
   b. If prompted for the type of web server to reconfigure select "apache2" with the spacebar and press <ENTER>
   c. If prompted to configure the database with dbconfig-common select <No> with your arrow keys and press <ENTER>
   d. Now we need to set up the .htaccess file for phpMyAdmin. .htaccess files are another type of configuration file used on Apache web servers. We first have to make a change to the apache configuration file.
      i. Open the file in nano `sudo nano /etc/phpmyadmin/apache.conf`
      ii. Under the directory section, add the line "AllowOverride All" under "Directory Index"

```
<Directory /usr/share/phpmyadmin>
        Options FollowSymLinks
        DirectoryIndex index.php
        AllowOverride All
```

   e. We have now allowed configuration with the .htaccess file. Now we can set up a user whose login would be required to access the phpmyadmin login page, adding another layer of security.
      i. Open the .htaccess file with nano `sudo nano /usr/share/phpmyadmin/.htaccess`
      ii. Set up the user authorization within the blank .htaccess file with the following:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /etc/.htpasswd
Require valid-user
```

ⓘ **AuthType:** Refers to the type of authentication that will be used to check the passwords. The passwords are checked via HTTP
AuthName: This is the text that will be displayed at the password prompt. You can type whatever you'd like between the "".
AuthUserFile: This designates the path to the password file (we'll create this next).
Require valid-user: Tells the .htaccess file that only users defined in the .htpasswd file can access the phpMyAdmin login screen

f. Now we need to create the .htpasswd file. We will use the htpasswd command to place the .htpasswd file in the /etc directory. Remember we set this in the last step, so make sure that your .htaccess file and the path to the file are the same. Be sure to put this somewhere where it is not accessible from the browser (i.e. do not use /var/www/mysite.dev/public_html).
   i. Create a new user with the following command. Replace username with the user name that you would like to use: `sudo htpasswd -c /etc/.htpasswd username`
   ii. You will then be prompted to provide and confirm a password. Remember to create a strong password.
   iii. Restart apache `sudo service apache2 restart`
   iv. You have just created a separate user name and password to access the phpMyAdmin login page.

Q12: Use the command `cat /etc/.htpasswd` to return the contents of the .htpasswd file that you just created. What do you see? This file contains an encrypted version of the user name and password that you just created. Why is this important?

ⓘ .htpasswd: encrypts passwords using an version of the MD5 algorithm. To read more on .htpasswd see https://httpd.apache.org/docs/current/programs/htpasswd.html.

MD5: is an algorithm that creates a 128-bit hash value. This means that the original data (in this case your password) is converted into a 128 bit string that must be decoded by the computer.

g. Open the web browser and navigate to http://mysite.dev/phpmyadmin

i. If your setup was successful you should be prompted for a user name and password in a pop-up before you are granted access to the phpMyAdmin site. This is the .htpasswd user and password that we just created.
ii. Now you are at the phpMyAdmin login. Here you need to provide the username and password for the cf2017 database (see step 3 above).
iii. If successful, you should now have access to phpMyAdmin. This tool allows you to administer your databases and users from a GUI interface rather than from the command line. We'll revisit this later this term.
iv. Logout by clicking on the door icon next to the home icon.

Q13: Why do you think that this two-step verification is important?

Congrats! You have now successfully installed a LAMP stack on your Pi. You can host webpages by placing .html or .php files into the public_html directory for the site. There are of course, other server options available (see Nginx [pronounced engine x] as an example https://www.nginx.com/resources/wiki/). In our next project, we will install WordPress and explore how the L-A-M-P work together.

Notebook Questions:

All of the required notebook questions are listed here for you to reference. Be sure to answer each question completely in your notebook, including an explanation of how you arrived at your answer.

Question	Page number
Q1: In your notebook explain this set of commands (steps 4-6) in your own words.	3
Q2: Where do you commonly see HTTP? Why?	4
Q3: Now we have a basic firewall established. Can you explain what the firewall is doing in your own words? Would you set any additional rules? Use the TCP/IP Ports and Protocols chart to review common ports and services http://www.pearsonitcertification.com/articles/article.aspx?p=1868080 You may also wish to set additional rules on your firewall using this chart.	4
Q4: Describe the differences between viewing websites in Lynx and through Firefox/Chrome/Safari/your favorite web browser.	5
Q5: What does the –y mean in this command? Use Explain Shell http://explainshell.com to find out.	6
Q6: The definition above states that the IP address is a 32-bit address, explain what this means in your own words (think back to the binary lesson from a few weeks ago).	7
Q7: In your own words, explain why a 32-bit address scheme provides just over 4 billion addresses. How many unique addresses does a 128-bit scheme provide for?	7
Q8: Open the page in the web browser by using localhost as the URL. Now try the IP address 127.0.1.1 and http://raspberrypi. What happens? Why?	7
Q9: Can you describe what you are looking at here? Hint: Take a look at the information between the <title> tags.	9
Q10: What does this command do?	10
Q11: Here we have called mysite.dev/index.php in the browser. Describe in your own words what we are doing with this command.	12
Q12: Use the command `cat /etc/.htpasswd` to return the contents of the .htpasswd file that you just created. What do you see? This file contains an encrypted version of the user name and password that you just created. Why is this important?	15
Q13: Why do you think that this two-step verification is important?	16