Setting up a LAMP server
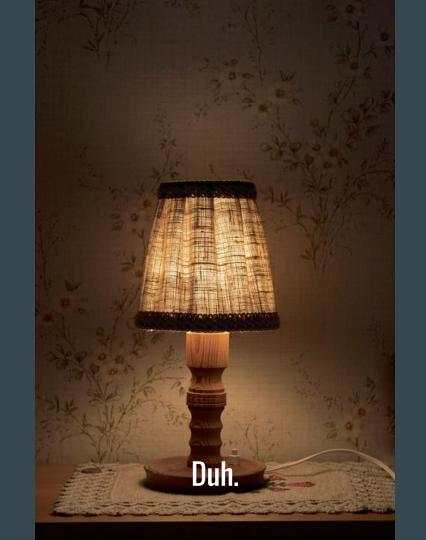
• • •

Created by: Nate Levesque (Feb. 2016)
Updated by: Justin W. Flory (Oct. 2016)

What is LAMP?

Actually, we're interested in...

- *"L*inux, *A*pache, *M*ySQL, and *P*HP"
  - A standard web server setup
  - Not the only technology options!

Linux

- Pick any! Common choices are:
    - Debian
    - Ubuntu Server
    - RHEL / CentOS
    - Related (enterprise) distros

- Normally it's wise to pick a distro that's somewhat stable
    - You can run a web server even on Arch if you're daring

Apache

- Very commonly used web server software

- Available in your package manager!
  - To enable the service if your installation doesn't do it automatically, enable `httpd` or `apache` depending on your distro

MySQL

- Database server

- Available in your package manager
  - Runs as a service, so you may need to enable it if your installation doesn't

- Install PHPMyAdmin to manage your database with a web GUI
  - Requires you to know less SQL

MySQL - PHPMyAdmin

PHP

- Common web application programming language
  - We won't discuss why it is or isn't awful here

- Available in your package manager! (Seems like a trend...aren't package managers great?)

Setting up LAMP

http://www.wikihow.com/Build-a-Lamp

wikiHow to Build a Lamp

Initial Installation

- Install your distro's Apache2, MySQL, and PHP packages
  - You may also need to install an `apache-php` or `php-mysql` package
  - Some distros provide a package called "LAMP" which installs these all for you in one shot. Depending on your distro, there may be good reason not to use it.

- Enable Apache2 and MySQL
  - systemd based systems: `systemctl enable httpd --now && systemctl enable mysql --now`

Tell Apache about PHP

- Apache will not handle PHP scripts by default

- This process may vary slightly depending on your distro and particular versions of Apache and PHP

Tell Apache about PHP

- Edit /etc/httpd/conf/httpd.conf

- Add "LoadModule php7_module modules/libphp7.so" under the line "LoadModule dir_module modules/mod_dir.so"

- Add "Include conf/extra/php7_module.conf" in the "Include" list in the file
  - near the bottom, find the last line that starts with "Include" that's not in a conditional

If you're using PHP 7 like me...

- In that same file:
  - Comment out "`LoadModule mpm_event_module modules/mod_mpm_event.so`"
  - Uncomment "`LoadModule mpm_prefork_module modules/mod_mpm_prefork.so`"

Finally...

1. Restart Apache (or httpd, if that's what your distro calls it)
2. Find the directory Apache will serve files from. Configurable, usually defaults to:
   a. /srv/http/
   b. /var/www/
3. Put a test file there named "test.php"
   a. Put "<?php phpinfo(); ?>" in it
4. Visit http://localhost/test.php
   a. Hopefully, that loads and you get a page with a bunch of PHP information
5. Delete the test file, leaving it up can be a security problem

Alternatives

LNMP

- Linux, **Nginx**, MySQL, and PHP

- Nginx is much lighter-weight, faster, and easier to configure than Apache
  - Frequently used in load balancing and proxy servers due to speed
  - Can do many of the same things, but is less powerful and is a younger project

Install Nginx

- Install nginx, php, and php-fpm
  - Start nginx and php-fpm

- Once started, visit http://localhost and you should see a simple "welcome to nginx" page

Set up Nginx

- Edit /etc/nginx/nginx.conf. In the server{} block, add:

```
location ~ \.php$ {
    root /usr/share/nginx/html # You can change this path!
    fastcgi_pass unix:/run/php-fpm/php-fpm.sock
    fastcgi_index index.php;
    include fastcgi.conf;
}
```

- Restart nginx and php-fpm

Set up Nginx

- Create a file called test.php in your PHP root with the contents:
  - `<?php phpinfo(); ?>`
  - (If you followed the defaults on the previous slide, this will be /usr/share/nginx/html/test.php)

- Visit http://localhost/test.php

LAPP

- Linux, Apache, **PostgreSQL**, PHP

- PostgreSQL is an alternative database, regarded in some ways to be better
    - But, it doesn't have shiny web UIs for management so you need to know some SQL

Or mix and match

- There are other options, depending on what you're building

- You can develop web applications in any language and make Apache, Nginx, or other web server software serve them
  - PHP may not be a good language to start with because it's extremely easy to write awful, insecure code and difficult to master
  - You can even use multiple languages for the same web application

- If you use a technology such as Rails (Ruby web framework), they may provide you with a server

Other tips and tricks

Running on a large scale

- Usually, the web server and the database server are not on the same system

- Multiple web servers that serve requests, with a load balancer in front
  - Does what it implies; it decides which server to send traffic to so things don't get overloaded
  - Often powered by Nginx!

- Frequently run in containers or virtual machines
  - Some companies even consider servers disposable and automatically wipe and rebuild them regularly

Be conscious of security

- Set up your database correctly
  - A disturbing number of sites have their database(s) exposed and open
  - Use a good password and change the defaults
  - Don't open your database to the Internet; only your webserver needs to talk to it
  - HASH (and salt) passwords, don't keep them in plaintext or reversible encryption

- Don't trust any data
  - HTML escape everything you send to the user with an existing library (don't roll your own!)
  - Use prepared statements when talking to your database so users can't run arbitrary SQL

- Don't roll your own encryption

- Learn how to do things properly (this is not an exhaustive list of tips!)

Use HTTPS for Everything!

- Back in the old days of one year ago, verified SSL certs cost money and took a bit of work to get

- [LetsEncrypt / Certbot](#) has changed all of this! Free SSL certs for everyone!

- There is no reason not to use HTTPS for all traffic these days.
  - Protect yourself and your users!
  - What a time to be alive!

Questions?
Comments?
Concerns?