Stacking LAMPs

Tom Ryder
tom@sanctum.geek.nz
https://sanctum.geek.nz/

What is the LAMP stack?

- A **web service** stack:
  - **L**inux
  - **A**pache HTTPD
  - **M**ySQL (or MariaDB)
  - **P**HP
- Very mature
  - In web years, anyway...the late 90s, even
  - Making a web page change dynamically was *a really big deal* back then
- Very popular (especially for **Content Management Systems**)
  - WordPress, Joomla!, Drupal, WooCommerce, SilverStripe, Magento, October...

What are the pieces?

- **Linux**: The operating system and kernel at the bottom of the stack
- **Apache HTTPD**: The venerable Hypertext Text Transport Protocol daemon: read and route requests, send the formed responses
- **MySQL/MariaDB**: The database server: store and retrieve persistent data in tables
- **PHP**: Run programs using data in the requests with reference to data in RAM/on disk, build responses to send to Apache HTTPD
  - Sometimes people will try to tell you that the "P" can stand for Perl.
  - It almost never actually does...not since the mid-90s, anyway.
  - But if you run a **LAMP(erl)** stack, Tom would *love* to know about that.
  - Python is a bit more common an alternative (Django, mostly).

Why LAMP?

- Three really big reasons that feed into each other:
  - It's **easy**, so the amount of code written for it and documentation on it is huge.
  - It's **everywhere**, in operating system repositories and on third-party webhosts, dirt cheap.
  - It's **mature**, as people have people been using it since the late nineties.
- Note that none of those reasons have much to do with *pure technical virtues*.
- If you're just beginning web programming, or if you're just a ruthless pragmatist, even in 2018 LAMP is *still* a reasonable place to start.
- Bonus reason: **WordPress**. Love it or hate it, it's friggin *everywhere*, and it runs on a LAMP stack.

Why not LAMP?

- You're a **programming purist**, and can't abide PHP.
  - And, well, fair enough...
- You have **very specific programming requirements**.
  - Maybe you need to do things with the JVM, or with .NET.
- You **really do need to scale**.
  - LAMP scales only to a point. PHP's runtime is *big*, partly because so much stuff is built into its core.
  - Forking many times a second can be painful.

LAMP Example: Installation

- We'll install a LAMP stack suitable for use on a trusted LAN in a few lines, on an internet-connected Debian GNU/Linux stable server.
  - Virtual machine, headless, connected over SSH.

    ```
    # apt install apache2 libapache2-mod-php php7.0 php7.0-mysql
    mariadb-server
    ```
- This is about as close to "it just works" as web programming gets.
- Versions:
  - Apache HTTPD 2.4
  - PHP 7.0
  - MariaDB 10
- All from packages.
- We'll take a quick look around how it's structured.

LAMP Example: WordPress

- Installing the WordPress content management system by unpacking a tarball.
- There's a package as well, but it makes running multiple sites awkward.

```
MariaDB> create user 'wordpress'@'localhost'
identified by 'WXAK2ZWY26wC2yAN';

MariaDB> create database wordpress;

MariaDB> grant all on wordpress.* to
'wordpress'@'localhost';
```

LAMP Example: phpMyAdmin

- Installing the phpMyAdmin MariaDB management system by installing another package.
- Lock it down to only specified hosts...

Architecture: Apache (MPM)

- **Prefork**: Default. Keep a certain number of processes running. One request per process.
  - Still required for classic mod_php, due to lack of thread safety in some libraries (and maybe even the core, depending on whom you ask)
  - Definitely good enough for your home webserver
- **Worker**: Keep a certain number of processes running. *Multiple* requests per process, using threads.
  - Kinder on system resources and can serve more requests
  - Breaks anything that wasn't designed with threads in mind
- There's **event** too, but that's essentially a more elegant **worker**, and there's not much difference if all your connections in production are HTTPS.
  - ...all your connections in production *are* HTTPS, right?

Architecture: PHP

- **CGI:** Old school, Perlish way. Figure out the type of script file at request time, and `exec()` the whole runtime for it.
  - *Slooooooooow,* but hey, it worked in the 90s
- **mod_php**: What we've done. Each Apache HTTPD server thread loads up the entire PHP runtime. All the server processes have PHP resident, ready and waiting.
  - Simple, quick, good for servers that run tons of PHP
  - Big resident size for each process, though
- **FastCGI**: Best of both worlds. PHP spawns processes in the background to wait for Apache to call on them.
  - Recommended method if the server only runs your sites, and you're keen for something a bit more advanced

Security: Server side

- **Separate users** for separate sites
  - For **mod_php**, use **apache_mpm_itk**
  - For **FastCGI**, it's probably built-in
- **chroot** to defeat elementary automated attacks?
  - For **mod_php**, use **mod_chroot** (or reconsider entirely)
  - For **FastCGI**, it's probably built-in
  - This won't stop a determined attacker
  - Might cause pain for dynamically loading libraries
  - A trade-off ... !

Security: Client side

- A bit out of scope here, *except*:
  - **Deploy HTTPS** for anything that has to go over an untrusted network!
    - Not a panacea, but given that it's free-as-in-beer now, thanks to Let's Encrypt, we're running out of excuses
    - Google cares, especially if your website has forms on it
    - *Especially* if they're username and password forms
    - SNI saved us, you don't need unique IPs anymore

Monitoring

- **~~Nagios~~ Icinga**
- After all, *some* poor jerk needs to get woken up at 3:00am when the whole thing crashes thanks to some kiddie carder in Bulgaria who got a new WordPress spl0it harrassing your customer's beanie baby history website admin login page with malformed Python `urllib` requests
- No, I'm not bitter

Questions?

Email: tom@sanctum.geek.nz

Web: https://sanctum.geek.nz/

Tom had a past life as a web developer, and now suffers for his sins, by running webservers for his former brethren.

Any other sysadmin topics in which people would be interested?