**Name of Title:** Learning Nginx

**Video Name:**

**Estimated Length:**

**Author Name:** Michael Jenkins

**Chapter\_Section\_Video:**

**Video Objective:**

At the end of this video the learner will be able to install and configure PHP on a Ubuntu VM.

**Script:**

To get started with our LEMP stack, I’m connected to our development VM with nginx already installed. Now let’s install PHP.

Also known as the Hypertext Preprocessor, PHP is a general-purpose scripting language especially suited to web development. You can learn more about PHP at PHP dot net.

EDITOR: Fly in a link to http://php.net/

We’ll be working as the root user so we should elevate our session with sudo su dash:

CTRL+L  
 sudo su -

And then run apt update to make sure the package repositories are up to date:

CTRL+L

apt update

And now we can install the PHP packages. We’ll do this with

CTRL+L

apt install -y php-fpm php-mysql

This will install the PHP FastCGI Process Manager and any other packages we need for PHP. The FastCGI process manager is a service that will process any PHP files received by the web server.

Once the installation is complete, we can check the version of PHP by running:

php --version

PHP 7.2.5-0ubuntu0.18.04.1 (cli) (built: May 9 2018 17:21:02) ( NTS )

Copyright (c) 1997-2018 The PHP Group

Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies

with Zend OPcache v7.2.5-0ubuntu0.18.04.1, Copyright (c) 1999-2018, by Zend Technologies

In this case, we’ll be using PHP 7.2.5.

Now let’s check the PHP FastCGI Process Manager. This gets installed as a service and we can make sure its up and running with systemctl.

systemctl status php7.2-fpm

\* php7.2-fpm.service - The PHP 7.2 FastCGI Process Manager

...

Great. This output tells us that php-fpm is up and running.

At this point, we’re ready to configure nginx so that it can communicate with the PHP service.

Now let’s edit the conf file for our site:

vim /etc/nginx/conf.d/wisdompetmed.local.conf

In the conf file, we need to add a location with a case sensitive regular expression that matches URIs ending in dot php:

location ~ \.php$ {

}

Next, we’ll use the include directive to bring in some code that’s stored outside of our config file:

location ~ \.php$ {

include snippets/fastcgi-php.conf;

}

The fastcgi-php.conf file contains helpful directives that improve the security and efficiency of site.

Now let’s add the fastcgi\_pass directive:

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass unix:/var/run/php/php7.2-fpm.sock;

}

This directive tells nginx to use a unix socket to communicate with php-fpm. Communicating through a socket keeps the transfers between nginx and PHP on the local system and is much faster than communicating via a TCP port.

We can also add one more directive:

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass unix:/var/run/php/php7.2-fpm.sock;

fastcgi\_intercept\_errors on;

}

This tells nginx that if any errors are returned by the fast CGI server, nginx should try to process the error with an error page directive if possible.

Nginx will consider errors from the the FastCGI process as 500 errors so these would be handled by out custom 50x.html page.

Now let’s save the file and test our configuration:

nginx -t

Everything looks good there so let’s reload the configuration:

systemctl reload nginx

Now the only thing we need is some PHP content. Let’s create a PHP info page in the root directory of the demo site:

vim /var/www/wisdompetmed.local/info.php

In this file we’ll put some PHP code that will return details about our PHP installation.

<?php phpinfo(); phpinfo(INFO\_MODULES); ?>

These commands will provide details on our PHP installation, served as a dynamically generated web page. I should add that making the phpinfo function output available on a web page should only be used for development and never in production. The details on the page might give a hacker or otherwise nosy visitor to our site more information about our site than they need, making it a target for attack.

EDITOR: Fly in a statement that says: PHP INFO is only for development, and should not be used in production sites.

OK. Now we can save this file and open the demo site in a browser to request the info page.

OPEN BROWSER AND GO TO:

<http://www.wisdompetmed.local/info.php>

Nice! Seeing our info page confirms that we have PHP-FPM installed and nginx configured to use it for processing web pages with PHP.

Let’s wrap up this lesson here and move on to installing the database server for our LEMP stack: MariaDB.

**Script and Media:**

Break the script up into parts and align it with any media (slides, web, CLI, etc.)

| **Part** | **Script** | **Media** |
| --- | --- | --- |
|  |  |  |

**Exercise Files:**

**Basement:**

Let’s start our PHP installation by booting up the VM for this lesson. If you’re following along with the exercise files, you can use the Vagrantfile for this chapter. It will boot the VM and install nginx along with our demo site.

If you're not using the exercise files, you can still follow along with a VM running Ubuntu 18.04 LTS where you have root access. You’ll also need to install nginx.

vagrant up