Web Dev Corner
Installing the LAMP Stack - Part 1
by Michael R. Youngblood

The “LAMP Stack” is traditionally Linux, Apache, MySQL, and PHP. This is the most necessary portion of any server to any web developer, whether you’re specializing in PHP or even mobile development. You just simply cannot develop a web site or web app without a web server. Of course, there are other flavors of web servers that use other technology, but this is the standard and will be our starting point.

For the sake of this new column, I have set up a new Virtual Server to go through with you as we go. My server is running Ubuntu Server 64bit 10.10. This month we will get Apache2 installed and configured. I am also assuming you know how to edit files using terminal and vi, this is what we will be using the whole time. Let’s jump right into that.

If you are not root (and you shouldn’t be for security reasons), you need to run apt-get commands using sudo, and all my examples will assume you are logged in as a user. Run the following code to install apache2:

sudo apt-get install apache2

By default, now it works. It is listening to all IP’s available to it, anything coming to that box on port 80 will now go to the default web site. Pretty easy stuff so far. All of your files will be located in the following directory:

/srv/www/

I have a feeling that we will want a few

Webfejlesztői sarok
A LAMP Stack telepítése- 1. rész
Írta Michael R. Youngblood

A “LAMP Stack” jelentése hagyományosan a Linux, Apache, MySQL, és PHP. Ez bármely webfejlesztő bármely szerverének legfontosabb része, akár PHP-ra szakosodsz, akár mobilfejlesztésre. Egyszerűen csak nem tudsz weboldalt vagy webalkalmazást fejleszteni webszerver nélkül. Persze vannak más stílusú webszerverek is, amelyek más technológiát használnak, de ez a szabványos és ez lesz a mi kiindulópontunk.
Az új rovat kedvéért beállítottam egy új virtuális szervert, hogy együtt végighaladjunk rajta. A szerveremen a 64 bites Ubuntu Server 10.10 fut. Ebben a hónapban az Apache2-t fogjuk telepíteni és beállítani. Azt is feltételezem, hogy tudod, hogyan kell fájlokat szerkeszteni a terminál és a vi használatával, ez az, amit egész idő alatt használni fogunk. Ugorjunk egyenesen bele.
Ha nem vagy root (és biztonság kedvéért nem is szabad annak lenned), az apt-get parancsokat sudo használatával kell futtatnod, és az összes példám azt feltételezi, hogy felhasználóként jelentkeztél be. Futtasd a következő kódot az apache2 telepítéséhez:
sudo apt-get install apache2

Ez most alapértelmezettként működk. Figyeli az összes elérhető IP-t, bármi, ami a 80-as porton érkezik, az alaéprtelmezett weboldalra kerül. Eddig egész könnyű a dolog. Az összes fájlod a következő könyvtárban lesz megtalálható:
/srv/www/

Van egy olyan érzésem, hogy akarunk néhány különböző oldalt, hogy játsszunk

different sites to play around with, so I am going to show you how I set things up. Instead of using the default path and apache config setup, we will use virtual hosts. From here on out, I will use example.com. You will want to replace that with your own domain name.

Make a new virtual host config file into /etc/apache2/sites-available/ with the following command:

sudo vi /etc/apache2/sites-available/example.com

Now let's get some config in there. Go ahead and use this simple sample configuration.

```
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /srv/www/example.com/public_html/
    ErrorLog /srv/www/example.com/logs/error.log
    CustomLog /srv/www/example.com/logs/access.log combined
</VirtualHost>
```

Remember to change example.com to your doman name. This stuff is kind of boring so I am just going to run through it really quick. ServerAdmin is for an email address of who or a group that maintains the site. ServerName should be the base name of the site. Please note, if your site is a sub-domain then you will need to put x.example.com in the ServerName. The ServerAlias is the full web address that will be going to your site.

velük, így meg fogom neked mutatni, hogy állítom be a dolgokat. Az alapútvonal és az apache config beállítás használata helyett virtuális hosztokat fogunk használni. Innentől kezdve az example.com-ot használom. Cseréld ki ezt a saját domain neveddel.

Készíts egy új virtuális hoszt config fájlt az /etc/apache2/sites-available/ helyre a következő paranccsal:

sudo vi /etc/apache2/sites-available/example.com

Most végezzünk el benne néhány beállítást. Menj előre és használd ezt az egyszerű mintabeállítást.

```
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /srv/www/example.com/public_html/
    ErrorLog /srv/www/example.com/logs/error.log
    CustomLog /srv/www/example.com/logs/access.log combined
</VirtualHost>
```

Ne felejtsd el az example.com-ot a saját domain nevedre módosítani. Ez a dolog elég unalmas, így csak gyorsan átfutok rajta. A ServerAdmin egy olyan felhasználó vagy csoport email címe, akik karbantartják az oldalt. A ServerName az oldal alapneve. Kérlek, jegyezd meg, ha az oldalad aldomain, akkor az x.example.com-ot kell a ServerName-be tenni. A ServerAlias a teljes webcím, amely az oldaladra vezet.
A DocumentRoot az a hely, ahol az összes

DocumentRoot is where all of your public files will be held. I took the liberty of giving you error log reporting to make finding and fixing problems easier in the future. Before any of that will work, we need to create those directories for real. That, of course, is as easy as making directories:

mkdir -p /srv/www/example.com/public_html

mkdir /srv/www/example.com/logs

Sweet, now we got some stuff going on. Now let's activate that bad boy:

sudo a2ensite example.com

sudo /etc/init.d/apache2 reload

The a2ensite is actually a really cool command. It says apache2, enable site x. There is also a2dissite for disabling. This will use the site config files we made in the sites-available directory and copy them into the sites-enabled directory. Although we could do it ourselves, it is just good practice to let apache handle its own files when it is able. The other statement there is telling apache to reload its configuration files.

Well, that is it for this month. Next time we will be installing PHP and MySQL to complete the LAMP stack.

BIO: **Michael Youngblood** has been in the industry of web design and development for 13 years. He has been working for a world wide wireless tech corp for six years and is working on his bachelor's of science in mobile development.

nyilvános fájlodat tartani fogod. Bátorkodom neked hibanapló jelentést adni, hogy a jövőben egyszerűbbé tegyem a problémák megtalálását és kijavítását. Mielőtt bármi működne, valóban létre kell hozni azokat a könyvtárakat. Ez persze ugyanolyan könnyű, mint a könyvtárak létrehozása:
mkdir -p /srv/www/example.com/public_html

mkdir /srv/www/example.com/logs

Király, most van néhány működő dolgunk. Most aktiváljuk ezt a rossz fiút:

sudo a2ensite example.com

sudo /etc/init.d/apache2 reload

Az a2ensite valóban tök jó parancs. Ez azt mondja az apache2-nek, engedélyezze az x oldalt. Van egy a2dissite is letiltáshoz. Ez az oldalkonfigurációs fájlokat használja, amiket a sites-available könyvtárban készítettünk és másoljuk át őket a sites-enabled könyvtárba. Bár ezt magunktól is megtehetnénk, jó gyakorlat, ha hagyjuk az apache-t a saját fájljait kezelni, amikor képes rá. A másik utasítás megmondja az apache-nak, hogy töltse be újra a konfigurációs fájljait.
Nos, ennyit erre a hónapra. Következő alkalommal telepíteni fogjuk a PHP-t és a MySQL-t a LAMP stack teljessé tételéhez.

Önéletrajz: **Michael Youngblood** 13 éve dolgozik a webtervezés és fejlesztés iparában. Hat évig egy világszerte működő vezetéknélküli technikai vállalatnak dolgozott, most pedig a bachelor fokozatán dolgozik a mobilfejlesztés területén.