



GRAV

TOUR

FEATURES

BLOG

DOWNLOADS



ABOUT

FORUM

LEARN

5,329

9,426

2,226







macOS 10.13 High Sierra Apache Setup: Multiple PHP Versions







Andy Miller posted on **04/07/2018** in [macos + sierra + apache + homebrew + php](#) [14 mins](#)

Warning

[Updated 4/6/2018] Due to [Homebrew/php](#)   tap being [deprecated at the end of March 2018](#), and the moving of all PHP formulas to [Homebrew/core](#)  , we've reworked our Guide to work with this new tap.

Info

If you have followed this guide in the past with the

`Homebrew/php`   tap, and are looking to **upgrade** to the new `Homebrew/core`   approach, then you should first clean-up your current installation by following our new [Upgrading Homebrew](#).

Part 1: macOS 10.13 High Sierra Web Development Environment

Note

This is an updated version of our prior OS X development series. The newly released macOS 10.13 High Sierra and the accompanying updates to Brew require significant changes compared to prior releases, necessitating a thorough revamp in the process. Since macOS 10.12 we now use **Homebrew's Apache**, rather than the built-in version, but this new approach is more flexible and should continue to work on prior OS X versions.

Developing web applications on macOS is a real joy. There are plenty of options for setting up your development environments, including the ever-popular [MAMP Pro](#) that provides a nice UI on top of **Apache**, **PHP** and **MySQL**. However, there are times when MAMP Pro has slow downs, or out of date versions, or is simply behaving badly due to its restrictive system of configuration templates and non-standard builds.

It is times like these that people often look for an alternative approach, and luckily there is one, and it is relatively straight-forward to setup.

In this blog post, we will walk you through setting up and configuring **Apache 2.4** and **multiple PHP versions**. In the

second blog post in this two-post series, we will cover **MySQL**, **Apache virtual hosts**, **APC** caching, and **Xdebug** installation.

! Tip

This guide is intended for **experienced web developers**. If you are a beginner developer, you will be better served using [MAMP](#) or [MAMP Pro](#).





XCode Command Line Tools

If you don't already have XCode installed, it's best to first install the command line tools as these will be used by homebrew:

```
$ xcode-select --install
```



Homebrew Installation

This process relies heavily on the macOS package manager called **Homebrew**. Using the `brew`   command you can easily add powerful functionality to your mac, but first we have to install it. This is a simple process, but you need to launch your **Terminal** (`/Applications/Utilities/Terminal`  ) application and then enter:

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com
```

Just follow the terminal prompts and enter your password where required. This may take a few minutes, but when complete, a quick way to ensure you have installed **brew**

  correctly, simply type:

```
$ brew --version  
Homebrew 1.5.14  
Homebrew/homebrew-core (git revision beff; las
```

You should probably also run the following command to ensure everything is configured correctly:

```
$ brew doctor
```

It will instruct you if you need to correct anything.

Apache Installation

The latest **macOS 10.13 High Sierra** comes with Apache 2.4 pre-installed, however, it is no longer a simple task to use this version with Homebrew because Apple has removed some required scripts in this release. However, the solution is to install Apache 2.4 via Homebrew and then configure it to run on the standard ports (80/443).

If you already have the built-in Apache running, it will need to be shutdown first, and any auto-loading scripts

removed. It really doesn't hurt to just run all these commands in order - even if it's a fresh installation:

```
$ sudo apachectl stop  
$ sudo launchctl unload -w /System/Library/Lau
```

Now we need to install the new version provided by Brew:

```
$ brew install httpd
```

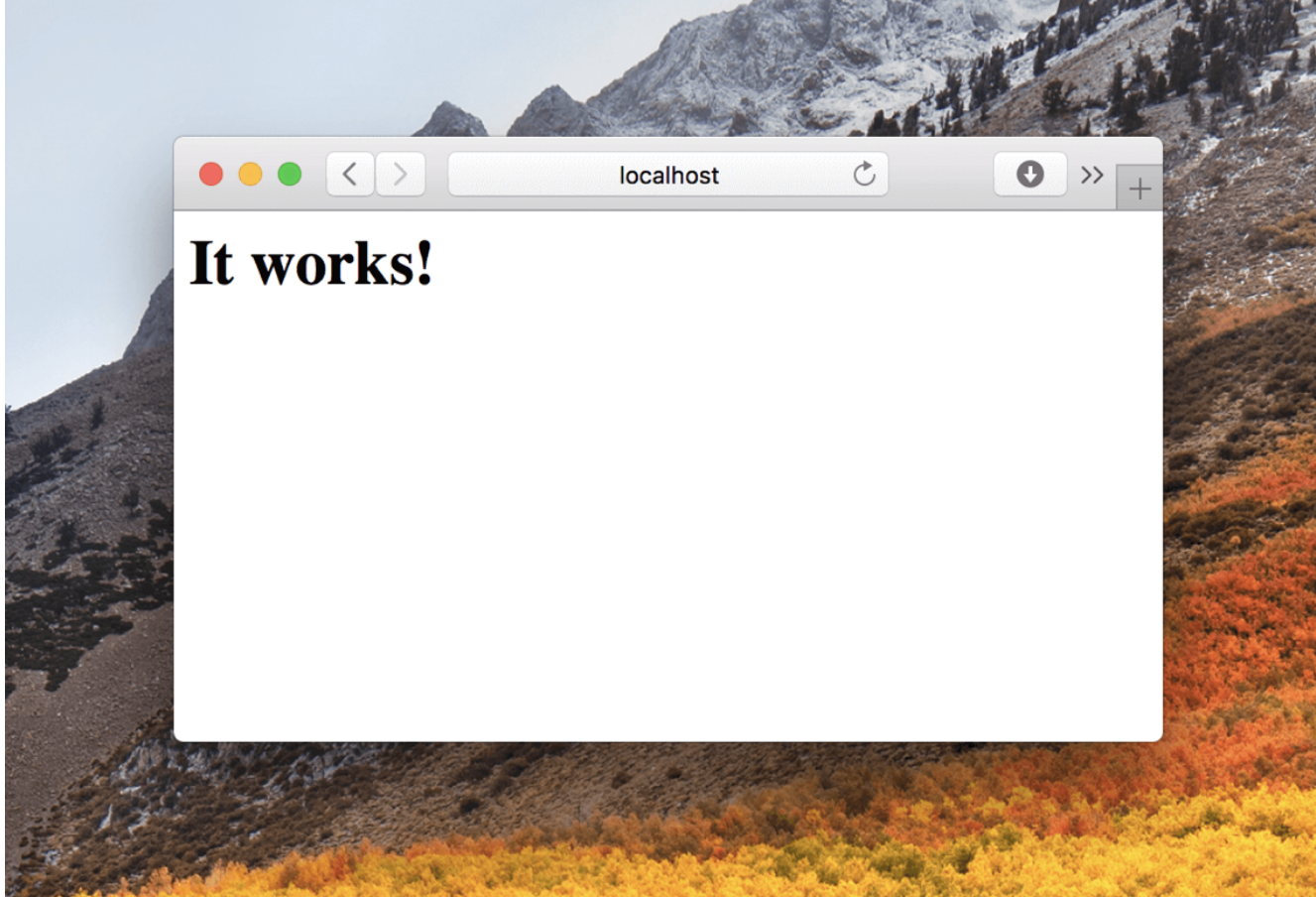
Without options, httpd won't need to be built from source, so it installs pretty quickly. Upon completion you should see a message like:

```
🍺 /usr/local/Cellar/httpd/2.4.33: 1,633 file
```

Now we just need to configure things so that our new Apache server is auto-started

```
$ sudo brew services start httpd
```

You now have installed Homebrew's Apache, and configured it to auto-start with a privileged account. It should already be running, so you can try to reach your server in a browser by pointing it at `http://localhost:8080`, you should see a simple header that says **"It works!"**.



Troubleshooting Tips

If you get a message that the browser can't connect to the server, first check to ensure the server is up.

```
$ ps -aef | grep httpd
```



You should see a few httpd processes if Apache is up and running.

Try to restart Apache with:

```
$ sudo apachectl -k restart
```



You can watch the Apache error log in a new Terminal tab/window during a restart to see if anything is invalid or causing a problem:



```
$ tail -f /usr/local/var/log/httpd/error_log
```

Apache is controlled via the `apachectl` command so some useful commands to use are:

```
$ sudo apachectl start
$ sudo apachectl stop
$ sudo apachectl -k restart
```

! Info

The `-k` will **force a restart immediately** rather than asking politely to restart when apache is good and ready

Apache Configuration

Now that we have a working web server, we will want to do is make some configuration changes so it works better as a local development server.

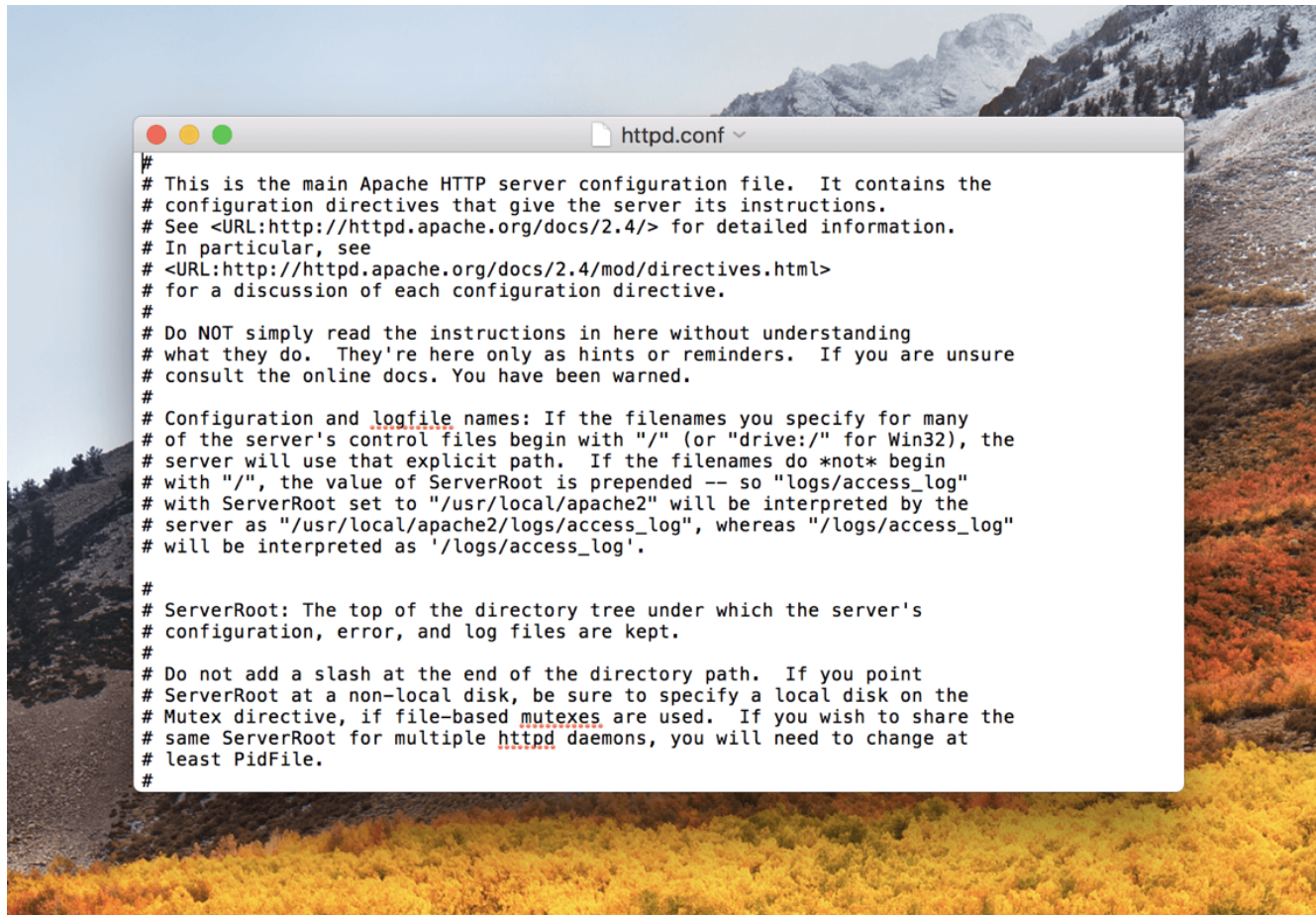
In the latest version of Brew, you have to manually set the listen port from the default of `8080` to `80`, so we will need to edit Apache's configuration file.

```
/usr/local/etc/httpd/httpd.conf
```

For simplicity we'll use the built-in **TextEditor** application to make all our edits. You can launch this from the Terminal by using the `open -e` command followed

by the path to the file:

```
$ open -e /usr/local/etc/httpd/httpd.conf
```



Find the line that says



```
Listen 8080
```

and change it to `80` :

```
Listen 80
```



Next we'll configure it to use the to change the **document root** for Apache. This is the folder where Apache looks to serve file from. By default, the document root is configured as `/usr/local/var/www` . As this is a

development machine, let's assume we want to change the document root to point to a folder in our own home directory.

Search for the term `DocumentRoot`  , and you should see the following line:



```
DocumentRoot "/usr/local/var/www"
```



Change this to point to your user directory where `your_user`   is the name of your user account:

```
DocumentRoot /Users/your_user/Sites
```



You also need to change the `<Directory>`   tag reference right below the DocumentRoot line. This should also be changed to point to your new document root also:

```
<Directory /Users/your_user/Sites>
```



Note

We removed the optional **quotes** around the directory paths as TextEdit will probably try to convert those to smart-quotes and that will result in a Syntax error when you try to restart Apache. Even if you edit around the quotes and leave them where they are, saving the document may result in their conversion and cause an error.

In that same `<Directory>` block you will find an `AllowOverride` setting, this should be changed as follows:

```
# AllowOverride controls what directives may be placed in this configuration
# It can be "All", "None", or any combination of the following:
#   AllowOverride FileInfo AuthConfig Limit
#   AllowOverride None
#   AllowOverride All
```

Also we should now enable **mod_rewrite** which is commented out by default. Search for `mod_rewrite.so` and uncomment the line by removing the leading `#`:

```
LoadModule rewrite_module lib/httpd/modules/mod_rewrite.so
```

User & Group

Now we have the Apache configuration pointing to a `Sites` folder in our home directory. One problem still exists, however. By default, apache runs as the user `daemon` and group `daemon`. This will cause permission problems when trying to access files in our home directory. About a third of the way down the `httpd.conf` file there are two settings to set the `User` and `Group` Apache will run under. Change these to match your user account (replace `your_user` with your real username), with a group of `staff`:

```
User your_user  
Group staff
```

Servername

Apache likes to have a server name in the configuration, but this is disabled by default, so search for:

```
#ServerName www.example.com:8080
```

and replace it with:

```
ServerName localhost
```

Sites Folder

Now, you need to create a **Sites** folder in the root of your home directory. You can do this in your terminal, or in Finder. In this new **Sites** folder create a simple **index.html** and put some dummy content in it like: `<h1>My User Web Root</h1>`.


```
$ mkdir ~/Sites  
$ echo "<h1>My User Web Root</h1>" > ~/Sites/i
```

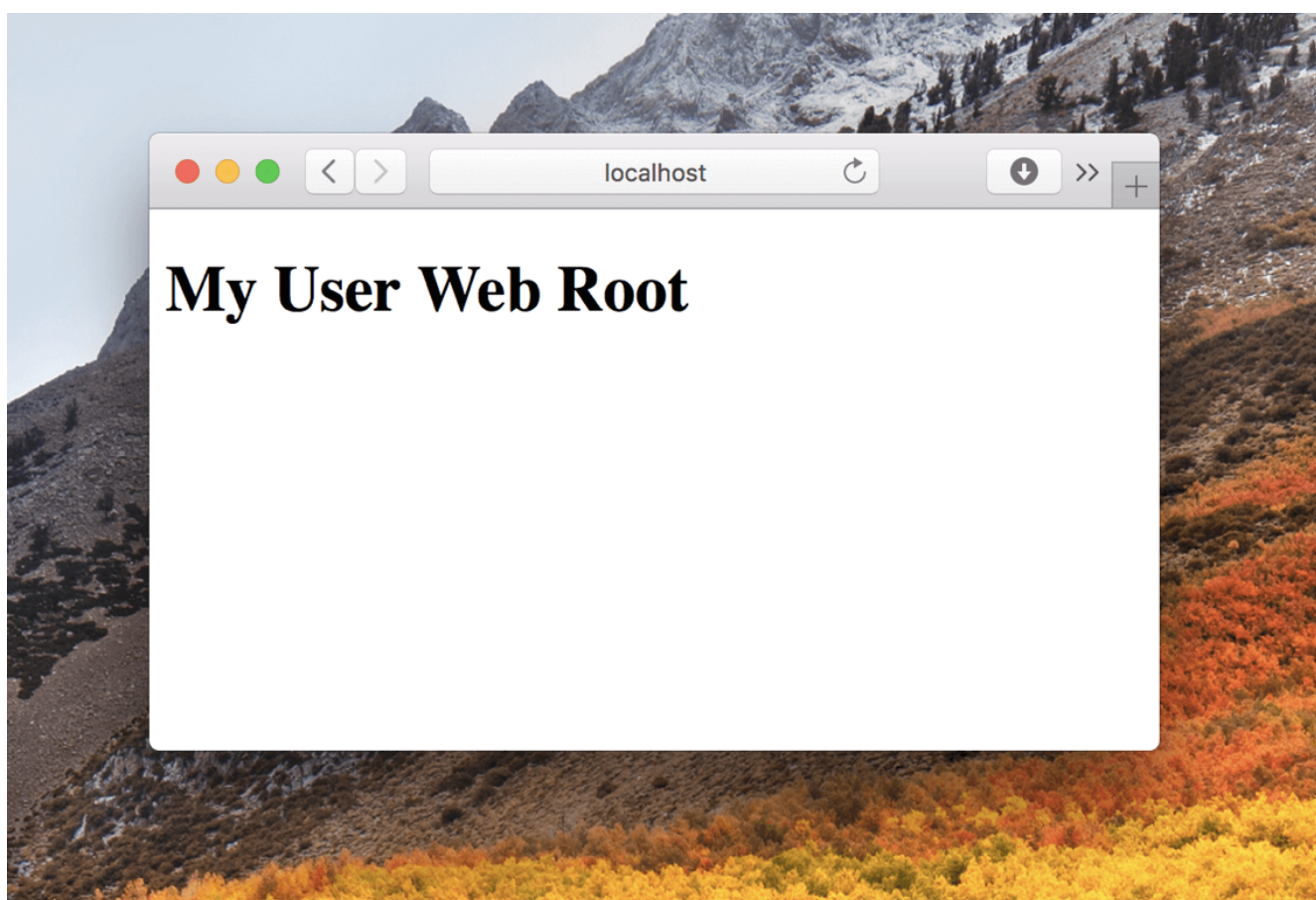
Restart apache to ensure your configuration changes have taken effect:

```
$ sudo apachectl -k restart
```

! Note

If you receive an error upon restarting Apache, try removing the quotes around the DocumentRoot and Directory designations we set up earlier.





Pointing your browser to `http://localhost`   should display your new message. If you have that working, we can move on!




PHP Installation

! Warning

If you have existing PHP installations via Brew, you need to first cleanup your setup with our [Upgrading Homebrew](#) guide before continuing with this section.

We will proceed by installing **PHP 5.6**, **PHP 7.0**, **PHP 7.1** and **PHP 7.2** and using a simple script to switch between them as we need. Up until the end of March 2018, all PHP related brews were handled by **Homebrew/php**   tab, but that has been deprecated, so now we use what's available in the **Homebrew/core**   package. This should be a better maintained, but is a much less complete, set of packages.





```
$ brew install php@5.6
$ brew install php@7.0
$ brew install php@7.1
$ brew install php@7.2
```

The first one will take a little bit of time as it has to install a bunch of brew dependencies. Subsequent PHP versions will install faster.

Note

You no longer have to **unlink**   each version between installing PHP versions as they are not linked by default

Also, you may have the need to tweak configuration settings of PHP to your needs. A common thing to change is the memory setting, or the **date.timezone**   configuration. The **php.ini**   files for each version of PHP are located in the following directories:



```
/usr/local/etc/php/5.6/php.ini
/usr/local/etc/php/7.0/php.ini
```

```
/usr/local/etc/php/7.1/php.ini  
/usr/local/etc/php/7.2/php.ini
```

Let's switch back to the first PHP version now:

```
$ brew unlink php@7.2 && brew link --force --o
```

! Info

At this point, I strongly recommend closing **ALL your terminal tabs and windows**. This will mean opening a new terminal to continue with the next step. This is strongly recommended because some really strange path issues can arise with existing terminals (trust me, I have seen it!).

Quick test that we're in the correct version:


```
php -v
```

```
PHP 5.6.35 (cli) (built: Mar 31 2018 20:21:31)  
Copyright (c) 1997-2016 The PHP Group  
Zend Engine v2.6.0, Copyright (c) 1998-2016 Ze  
with Zend OPcache v7.0.6-dev, Copyright (c
```

Apache PHP Setup - Part 1

You have successfully installed your PHP versions, but we need to tell Apache to use them. You will again need to edit the `/usr/local/etc/httpd/httpd.conf` file



scroll to the bottom of the `LoadModule`   entries.

If you have been following this guide correctly, the last entry should be your `mod_rewrite`   module:

```
LoadModule rewrite_module lib/httpd/modules/mo
```

Below this add the following `libphp`   modules:

```
LoadModule php5_module /usr/local/opt/php@5.6/  
#LoadModule php7_module /usr/local/opt/php@7.0  
#LoadModule php7_module /usr/local/opt/php@7.1  
#LoadModule php7_module /usr/local/opt/php@7.2
```

We can only have one module processing PHP at a time, so for now, so we have left our `php@5.6`   entry uncommented while all the others are commented out. This will tell Apache to use PHP 5.6 to handle PHP requests. **(We will add the ability to switch PHP versions later).**

Also you must set the Directory Indexes for PHP explicitly, so search for this block:

```
<IfModule dir_module>  
    DirectoryIndex index.html  
</IfModule>
```


and replace it with this:



```
<IfModule dir_module>
    DirectoryIndex index.php index.html
</IfModule>

<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
```





Save the file and **stop Apache then start again**, now that we have installed PHP:



```
$ sudo apachectl -k stop
$ sudo apachectl start
```



Validating PHP Installation

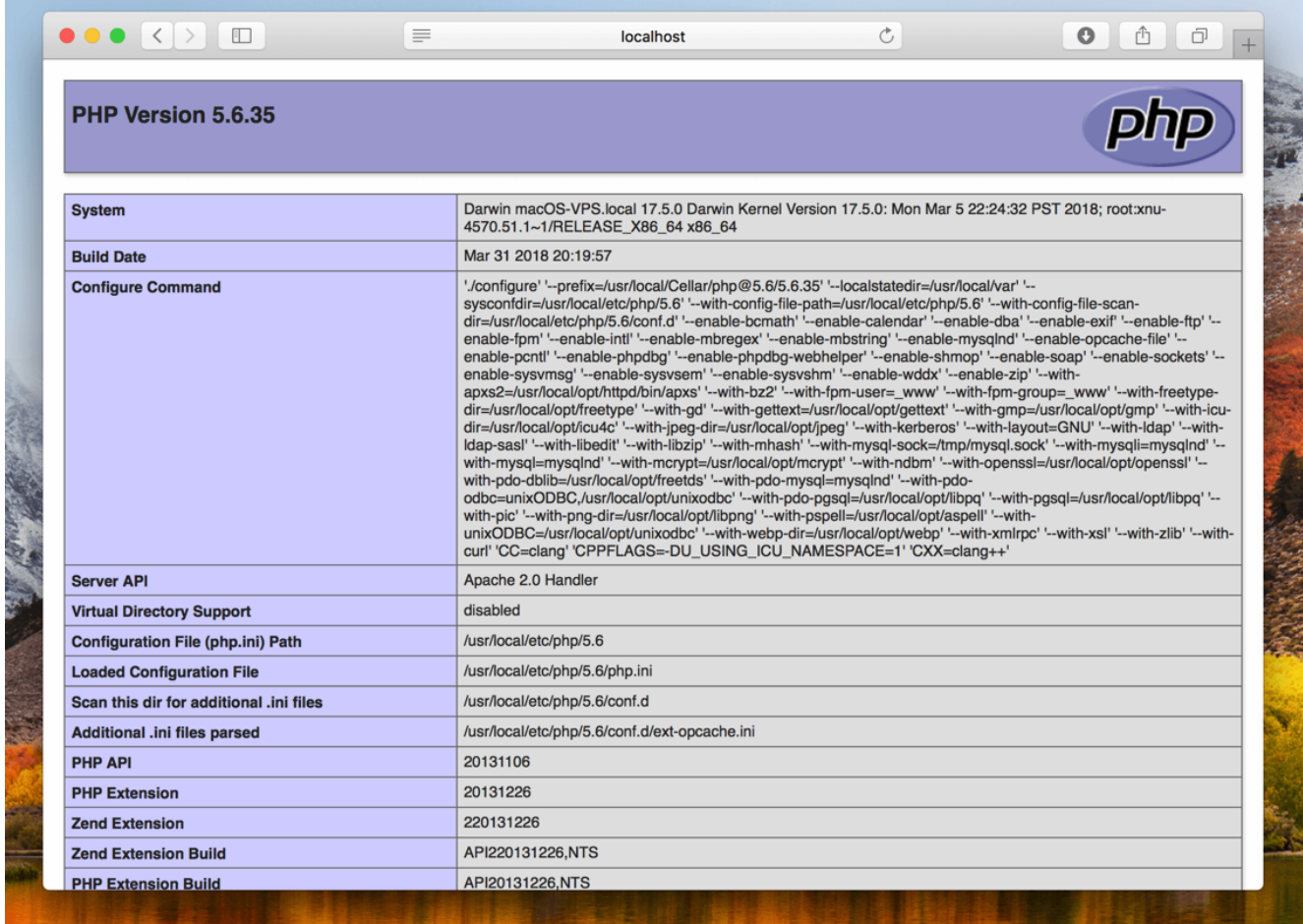
The best way to test if PHP is installed and running as expected is to make use of [phpinfo\(\)](#). This is not something you want to leave on a production machine, but it's invaluable in a development environment.

Simply create a file called `info.php`   in your `Sites/`   folder you created earlier with this one-liner.



```
echo "<?php phpinfo();" > ~/Sites/info.php
```

Point your browser to `http://localhost/info.php`   and you should see a shiny PHP information page:



PHP Version 5.6.35	
System	Darwin macOS-VPS.local 17.5.0 Darwin Kernel Version 17.5.0: Mon Mar 5 22:24:32 PST 2018; root:xnu-4570.51.1~1/RELEASE_ARM_T8020
Build Date	Mar 31 2018 20:19:57
Configure Command	'./configure' '--prefix=/usr/local/Cellar/php@5.6/5.6.35' '--localstatedir=/usr/local/var' '--sysconfdir=/usr/local/etc/php/5.6' '--with-config-file-path=/usr/local/etc/php/5.6' '--with-config-file-scan-dir=/usr/local/etc/php/5.6/conf.d' '--enable-bcmath' '--enable-calendar' '--enable-exif' '--enable-ftp' '--enable-fpm' '--enable-intl' '--enable-mbregex' '--enable-mbstring' '--enable-mysqlnd' '--enable-opcache-file' '--enable-pcntl' '--enable-phdbg' '--enable-phdbg-webhelper' '--enable-shmop' '--enable-soap' '--enable-sockets' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--enable-zip' '--with-apxs2=/usr/local/opt/httpd/bin/apxs' '--with-bz2' '--with-fpm-user=_www' '--with-fpm-group=_www' '--with-freetype-dir=/usr/local/opt/freetype' '--with-gd' '--with-gettext=/usr/local/opt/gettext' '--with-gmp=/usr/local/opt/gmp' '--with-icu-dir=/usr/local/opt/icu4c' '--with-jpeg-dir=/usr/local/opt/jpeg' '--with-kerberos' '--with-layout=GNU' '--with-ldap' '--with-ldap-sasl' '--with-libedit' '--with-libzip' '--with-mhash' '--with-mysql-sock=/tmp/mysql.sock' '--with-mysqli=mysqlnd' '--with-mysql=mysqlnd' '--with-mcrypt=/usr/local/opt/mcrypt' '--with-ndbm' '--with-openssl=/usr/local/opt/openssl' '--with-pdo-dblib=/usr/local/opt/freetds' '--with-pdo-mysql=mysqlnd' '--with-pdo-odbc=unixODBC,/usr/local/opt/unixodbc' '--with-pdo-pgsql=/usr/local/opt/libpq' '--with-pgsql=/usr/local/opt/libpq' '--with-pic' '--with-png-dir=/usr/local/opt/libpng' '--with-pspell=/usr/local/opt/aspell' '--with-unixODBC=/usr/local/opt/unixodbc' '--with-webp-dir=/usr/local/opt/webp' '--with-xmlrpc' '--with-xsl' '--with-zlib' '--with-curl' 'CC=clang' 'CFLAGS=-DU_USING_ICU_NAMESPACE=1' 'CXX=clang++'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php/5.6
Loaded Configuration File	/usr/local/etc/php/5.6/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/5.6/conf.d
Additional .ini files parsed	/usr/local/etc/php/5.6/conf.d/ext-opcache.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API20131226,NTS
PHP Extension Build	API20131226,NTS

If you see a similar **phpinfo** result, congratulations! You now have Apache and PHP running successfully. You can test the other PHP versions by commenting the `LoadModule ... php@5.6 ...` entry and uncommenting one of the other ones. Then simply restart apache and reload the same page.

PHP Switcher Script

We hard-coded Apache to use **PHP 5.6**, but we really want to be able to switch between versions. Luckily, some industrious individuals have already done the hard work for us and written a very handy little [PHP switcher script](#).

We will install the `sphp` script into brew's standard `/usr/local/bin`:

```
$ curl -L https://gist.githubusercontent.com/r
$ chmod +x /usr/local/bin/sphp
```

Check Your Path

Homebrew should have added its preferred

`/usr/local/bin` and `/usr/local/sbin` to your path as part of its installation process. Quickly test this by typing:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

If you don't see this, you might need to add these manually to your path. Depending on your shell you are using, you may need to add this line to `~/.profile`, `~/.bash_profile`, or `~/.zshrc`. We will assume you are using the default bash shell, so add this line to a your `.profile` (create it if it doesn't exist) file at the root of your user directory:

```
export PATH=/usr/local/bin:/usr/local/sbin:$PA
```

Testing the PHP Switching

After you have completed these steps, you should be able to switch your PHP version by using the command `sphp` followed by a two digit value for the PHP version:

```
$ sphp 7.0
```


You will probably have to enter your administrator

password, and it should give you some feedback:

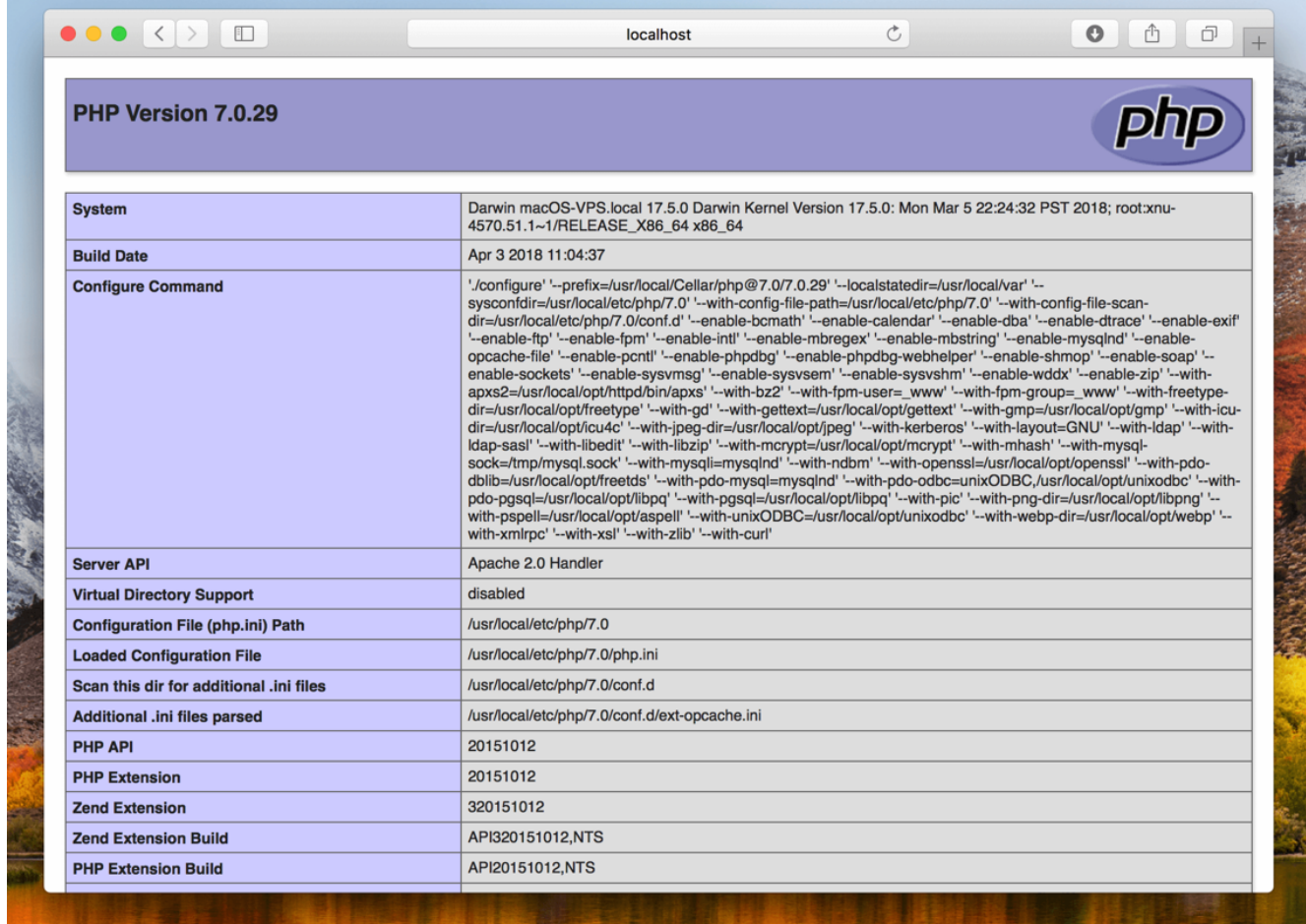


```
$ sphp 70
Switching to php@7.0
Switching your shell
Unlinking /usr/local/Cellar/php@5.6/5.6.35...
Unlinking /usr/local/Cellar/php@7.0/7.0.29...
Unlinking /usr/local/Cellar/php@7.1/7.1.16...
Unlinking /usr/local/Cellar/php/7.2.4... 0 sym
Linking /usr/local/Cellar/php@7.0/7.0.29... 47
```

```
If you need to have this software first in you
echo 'export PATH="/usr/local/opt/php@7.0/bin:'
echo 'export PATH="/usr/local/opt/php@7.0/sbin:'
You will need sudo power from now on
Switching your apache conf
Password:
Restarting apache
All done!
```

Test to see if your Apache is now running PHP 7.0 by again pointing your browser to `http://localhost/info.php` 

 . With a little luck, you should see something like this:



Updating PHP and other Brew Packages

Brew makes it super easy to update PHP and the other packages you install. The first step is to **update** Brew so that it gets a list of available updates:

```
$ brew update
```



This will spit out a list of available updates, and any deleted formulas. To upgrade the packages simply type:

```
$ brew upgrade
```



! Info

You will need to switch to each of your installed PHP versions and run **update** again to get updates for each

PHP version and ensure you are running the version of PHP you intend.

Activating Specific/Latest PHP Versions

Due to the way our PHP linking is set up, only one version of PHP is **linked** at a time, only the current **active** version of PHP will be updated to the latest version. You can see the current active version by typing:

```
$ php -v
```



And you can see the specific versions of PHP available by typing:

```
$ brew info php@7.0
php@7.0: stable 7.0.29 (bottled) [keg-only]
General-purpose scripting language
...
```



OK, that wraps up Part 1 of this 3 part series You now have a fully functional Apache 2.4 installation with a quick-and-easy way to toggle between PHP 5.6, 7.0, 7.1 and 7.2. [Check out Part 2](#) to find out how to setup your environment with **MySQL**, **Virtual Hosts**, **APC** caching, **YAML**, and **Xdebug**. Also [take a gander at Part 3](#) to find out how to setup **SSL** for your Apache Virtual Hosts.



RELATED POSTS

14
NOV

JOURNAL

Picking a Development


 ANDY MILLER

🕒 6 MINS

15
DEC

TUTORIAL

Fast Free Grav Development

 ANDY MILLER

🕒 9 MINS

17
NOV

TUTORIAL

Grav Development

 ANDY MILLER

🕒 10 MINS



DOWNLOAD GRAV

NEVER MISS A THING, SIGN UP TO THE GRAV MAILING LIST

email address

Grav was `</>` with  by [RocketTheme](#)

CDN provided by  [MaxCDN](#)

Designed by [Eduardo Santos](#)

- [Contact the Grav Team](#)
- [About Grav](#)
- [Grav Media](#) Information
- [Grav News Feed](#)

Crazy Fast VPS Hosting [Sponsored by Linode](#) - managed by [ServerPilot](#) - monitoring by [Pingometer](#)

Copyright @2018 - Grav CMS - All rights reserved - Grav is released under the [MIT license](#)