PHP-FPM

High-performance PHP on apache httpd 2.4.x using mod_proxy_fcgi and php-fpm.

With the release of apache httpd 2.4 upon an unsuspecting populace, we have gained some very neat functionality regarding apache and php: the ability to run PHP as a fastCGI process server, and address that fastCGI server *directly from within apache*, via a dedicated proxy module (mod_proxy_fcgi.)

- starting from release 5.3.3 in early 2010, PHP has merged the php-fpm fastCGI process manager into its codebase, and it is now (as of 5.4.1) quite stable.
- php-fpm was previously found at http://php-fpm.org

This means that we can now run secure, fast, and dependable PHP code *using only the stock apache httpd and php.net releases*; no more messing around with **suphp** or **suexec** - or, indeed, **mod_php**.

php-fpm

Prerequisites:

- installing software packages
- editing configuration files
- controlling service daemons.

From release 5.3.3 onwards, PHP now includes the fastCGI process manager (php-fpm) in the stock source code.

Your distribution or OS will either include it in the stock PHP package, or make it available as an add-on package; you can build it from source by adding `--enable-fpm` to your ./configure options.

This provides us with a new binary, called `php-fpm`, and a default configuration file called `php-fpm.conf` is installed in `/etc`.

The defaults in this file should be okay to get you started, but be aware that your distribution may have altered it, or changed its location.

Inside this configuration file you can create an arbitrary number of fastcgi "pools" which are defined by the IP and port they listen on, just like apache virtualhosts.

The most important setting in each pool is the TCP socket (IP and port) or unix domain socket (UDS) php-fpm will be listening on to receive fastCGI requests; this is configured using the `listen` option.

The default pool, `[www]`, has this configured as `listen 127.0.0.1:9000`: it will only respond to requests on the local loopback network interface (localhost), on TCP port 9000.

Also of interest are the *per-pool* `user` and `group` options, which allow you to run that specific fpm pool under the given uid and gid; goodbye suphp!

Pay special attention to the slowlog settings (request_slowlog_timeout and slowlog directives), that is, through this log and a reasonable amount of timeout, you can easily see which php calls from your application take longer than expected and start debugging them right away.

Let's just use the defaults as shipped and start the php-fpm daemon; if your distro uses the provided init script, run
`/etc/init.d/php-fpm start`
Or if not, start it manually with
`php-fpm -y /path/to/php-fpm.conf -c /path/to/custom/php.ini`
If you don't provide php-fpm with its own `php.ini` file, the *global* php.ini will be used. Remember this when you want to include more or less extensions than the CLI or CGI binaries use, or need to alter some other values there.

You can include per-pool `php.ini` values in the same way you would define these in apache previously for mod_php, using `php_[admin_](flag|value)`.

See the **official PHP documentation for fpm** for all possible configuration options.

I also changed the php-fpm.conf `logging` option so I can easily see what is being logged specifically by php-fpm:
`error_log /var/log/php-fpm.log`
If you don't set a php-fpm logfile, errors will be logged as defined in `php.ini`.

Side note: you can force a running php-fpm to reload its configuration by sending it a SIGUSR2 signal.; SIGUSR1 will cycle the log files (perfect for a logrotate script!). A little experimentation goes a long way 😉

That's php-fpm taken care of; if there were no errors during startup it should be listening and ready for connections.

apache httpd 2.4

prerequisites:

- editing httpd.conf
- understanding the vhost context
- understanding URL-to-filesystem namespace mapping
- controlling the apache httpd daemon

This release of apache httpd has introduced two noteworthy features: a new proxy module specifically for fastCGI (mod_proxy_fcgi), and the move to the **e vent** MPM as the default apache process manager.

As with the worker MPM of the previous version, the threaded model of this MPM causes issues when mod_php is used with *non-thread-safe* third-party PHP extensions.

This has been a bane of mod_php users ever since apache 2.2 was released, practically forcing them to cobble together fastcgi solutions, or use the much slower and memory-hungry prefork MPM.

To work the magic with the PHP fastCGI process manager, we will be using a new module, **mod_proxy_fcgi**, which is intended specifically for communicating with (possibly external) fastCGI servers.

Make sure you include the proxy_fcgi module in your httpd.conf so we can use its features; since this requires the base proxy module, ensure both are loaded (uncommented):

```
LoadModule proxy_module modules/mod_proxy.so

LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
```

Now, there are different ways to actually forward requests for .php files *to* this module, ranging from everything (using [ProxyPass]) to very specific or rewritten files or patterns (using mod_rewrite with the [P] flag).

The method I chose (using ProxyPassMatch) lies somewhere in between these in complexity and flexibility, since it allows you to set one rule for all PHP content of a specific vhost, but will only proxy .php files (or URLs that contain the text `.php` somewhere in the request).

TCP socket (IP and port) approach

Edit the configuration for a vhost of your choice, and add the following line to it:

```
ProxyPassMatch ^/(.*\.php(/.*)?)$ fcgi://127.0.0.1:9000/path/to/your/documentroot/$1

DirectoryIndex /index.php index.php
```

Look confusing ? Let's run through it:

ProxyPassMatch

${renderedContent}
^/(.*\.php(/.*)?)$
${renderedContent}
The ^ (caret) and $ (dollar) signs are used to *anchor* both the absolute start and end of the URL, to make sure no characters from the request escape our pattern match.

The nested parentheses enable us to refer to the entire request-URI (minus the leading slash) as `$1`, while still keeping the trailing pathinfo optional.

fcgi://127.0.0.1:9000

${renderedContent}

This determines which fastcgi *pool* will serve requests proxied by this rule.
/path/to/your/documentroot/
${renderedContent}
php-fpm just interprets the php files passed to it; it is not a web server, nor does it understand your web servers' namespace, virtualhost layout, or aliases.

IMPORTANT! Read the above again
$1

${renderedContent}
DirectoryIndex /index.php index.php
${renderedContent}

unix domain socket (UDS) approach

Edit the configuration for a vhost of your choice, and add the following line to it:

```
ProxyPassMatch ^/(.*\.php(/.*)?)$ unix:/path/to/socket.sock|fcgi://localhost/path/to/your/documentroot/
```

unix:/path/to/socket.sock

${renderedContent}
Note that with this approach, the captured request URI ($1) is not passed after the path

Proxy via handler

With this approach, you can check for the existence of the resource prior to proxying to the php-fpm backend.

```
# Defining a worker will improve performance

# And in this case, re-use the worker (dependent on support from the fcgi application)

# If you have enough idle workers, this would only improve the performance marginally

<Proxy "fcgi://localhost:9000/" enablereuse=on max=10>

</Proxy>

<FilesMatch "\.php$">

{{ <If "-f %{REQUEST_FILENAME}">}}

{{ # Pick one of the following approaches}}

{{ # Use the standard TCP socket}}

{{ #SetHandler "proxy:fcgi://localhost/:9000"}}

{{ # If your version of httpd is 2.4.9 or newer (or has the back-ported feature), you can use the unix domain socket}}

{{ #SetHandler "proxy:unix:/path/to/app.sock|fcgi://localhost/"}}

{{ </If>}}

</FilesMatch>
```

Combining FilesMatch and If can achieved as such:

```
<If "-f %{REQUEST_FILENAME} && %{REQUEST_URI} =~ /.+\.ph(ar|p|tml)$/" >
```

For the impatient

Very simple example

If you're interested into the proof of concept and want to leave the tweaking for later, you can use the following recipe. It'll conjure up the standard php info page listing all compiled-in and loaded extensions, and all runtime configuration options and script info.

First, create a file, /var/www/info.php containing:

```
<?php phpinfo() ?>
```

The assumption is that /var/www is the **DocumentRoot** of an existing vhost.

Inside this vhost, add the following line:

```
ProxyPassMatch ^/info$ fcgi://127.0.0.1:9000/var/www/info.php
```

Reload apache with `apachectl graceful` and you can now call up the phpinfo page using `http://example.com/info`

This is a very simple example, mapping one unique URL to a single PHP file.

A more flexible example

To proxy **all** `.php` files in your vhost to the fcgi server using their real php file locations, you can use a more flexible match:

```
ProxyPassMatch ^/(.*\.php)$ fcgi://127.0.0.1:9000/var/www/$1
```

Again, assuming `/var/www` is the **DocumentRoot** of the vhost in question.

Reload apache with `apachectl graceful` and you can now call up the phpinfo page using `http://example.com/yourscript.php`

Performance and Pitfalls

mod_proxy_fcgi now supports unix domain sockets since 2.4.9 (Unix domain socket support for mod_proxy_fcgi)

It is easy to overwhelm your system's available sockets, pass over ulimits, etc. Some tips to avoid this:

Using too many sockets will cause apache to give a `(99)Cannot assign requested address:` error. This means your operating system is not allowing new sockets to be created.

On linux you can use /proc/sys/net/ipv4/tcp_tw_reuse to not build up as many sockets, but there are warnings associated with using this behind a NAT.

Be sure to modify ulimit and allow for enough open files and processes for both the apache user and the php-fpm user.
`ulimit -n` and `ulimit -u` (nofile and nproc)

If php-fpm does not have a large enough nproc it will exit (`code 255`, no additional information as of php 5.3) in a loop without additional messages.

If php-fpm does not have a large enough nofile you will probably not be able to get logging per child, as shown above. It will give this in the general error log.

If apache and php-fpm run as the same user (not necessary or recommended) and nproc is too small, apache will not startup with the following message (`11)Resource temporarily unavailable: AH02162: setuid: unable to change to uid: 600`

Warning: when you ProxyPass a request to another server (in this case, the php-fpm daemon), authentication restrictions, and other configurations placed in a Directory block or .htaccess file, may be bypassed.

Caveats

One might be tempted to point out that a greedy **ProxyPassMatch** directive might allow some malicious content uploaded by a HTTP client to be served.

This is by no means a comprehensive security document, but instead will point out a possible injection vector that could be generated from the directives in this document.

Take, for example:

```
/uploads/malicious.jpg/lalalaalala.php
```

Would lead php-fpm to process that file (/uploads/malicious.jpg), and without certain sanity check, possibly lead to a compromised server.

This, of course, is not recommended. Content uploaded using php should be saved safely outside the **DocumentRoot**, and the pathinfo should be scrutinized.

Additionally, php-fpm should check if the script being invoked is allowed.

If such restrictions cannot be implemented easily, then checks could be performed prior to proxying with a **RewriteCond** or **FallbackResource** to ensure that the URI is not altered by the HTTP client.

Experiments with timeouts

For long-running scripts, setting a large timeout might help. This approach will set the time out value to 300 seconds, and allow you to be selective about what requests are proxied:

```
<Proxy "unix:/var/run/php-fpm/example.com.sock|fcgi://localhost">
   ProxySet timeout=300
</Proxy>

<FilesMatch \.php$>
   SetHandler "proxy:fcgi://localhost"
</FilesMatch>
```