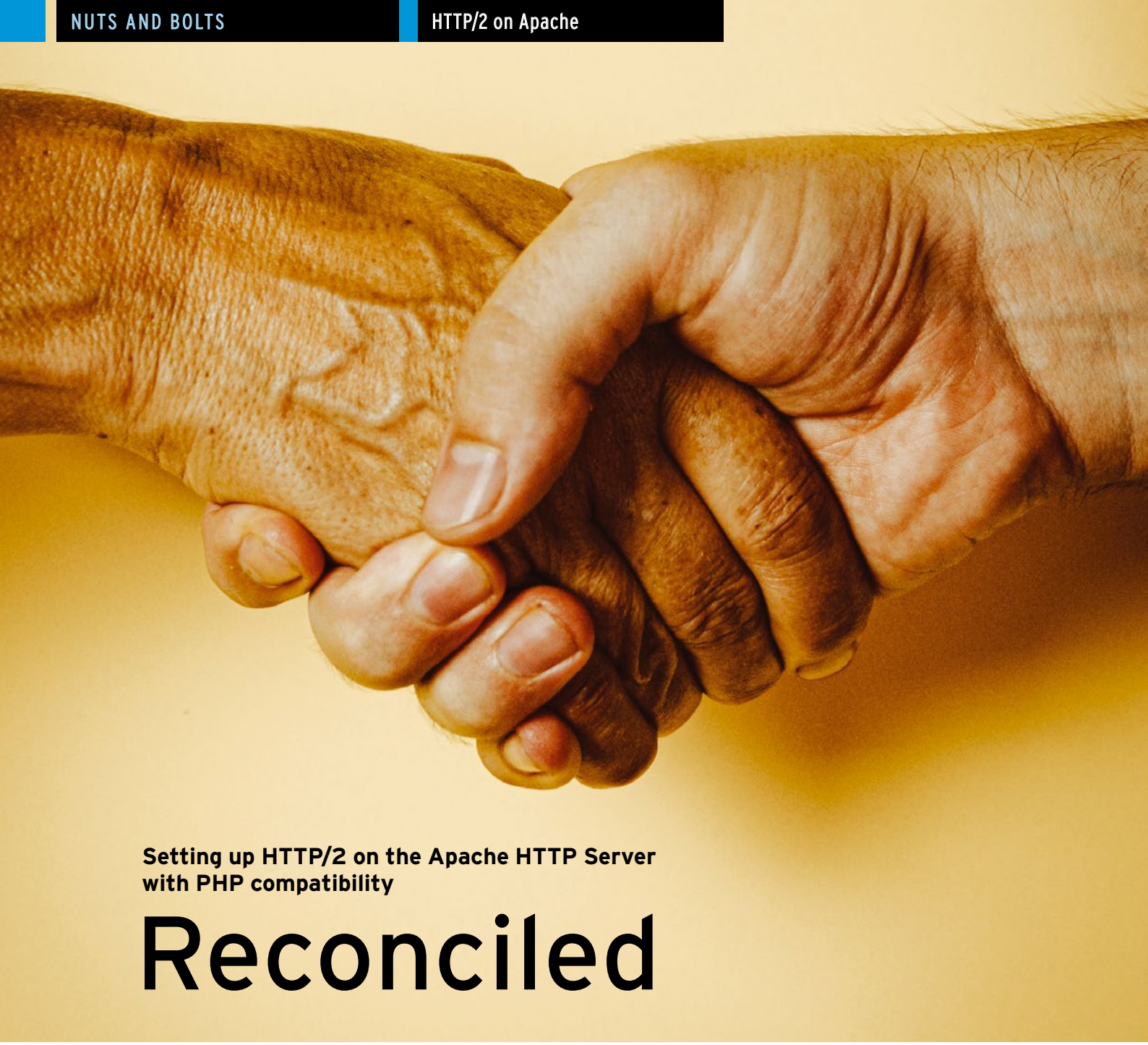

Setting up HTTP/2 on the Apache HTTP Server with PHP compatibility

Reconciled

If you are running PHP applications, setting up HTTP/2 on the Apache HTTP Server can be a bit confusing because of some incompatibilities between the Apache HTTP/2 module and the Prefork multiprocessing module.

By Eugenia Bahit

HTTP/2 is the second version of the hypertext transfer protocol and brings two significant improvements: more efficient network resource usage and reduced latency.

Latency (the elapsed time between the user sending a request to the server and receiving the response) can either increase or decrease according to the way network resources are handled, so higher latency means a longer wait for the user.

To reduce latency, HTTP/1.1 implemented a request pipelining technique, allowing TCP connections to send multiple requests without waiting for a response. However, although request pipelining facilitates concurrency, it does not avoid head-of-line (HoL) blocking, which occurs when a single data packet queue blocks subsequent transmissions while waiting for a response. This phenomenon affects both the TCP and HTTP protocols, so it appears at both the Transport and Application layers of the TCP/IP protocol stack.

HTTP/2 solves the HoL blocking issue for HTTP at the Application layer, improving concurrency and reducing latency. For that reason, setting up HTTP/2 will enhance the performance of the websites running on your server. If your website uses PHP, however, you'll need to go to a little more effort. The HTTP/2 protocol on the Apache HTTP Server requires a conventional multiprocessing module (MPM) such as MPM Event. The problem is that the default PHP binary for Apache installs MPM Prefork as a dependency, which is not compatible with the HTTP/2 protocol.

A straightforward approach to solving the PHP compatibility issue is to

Photo by Ave Calvar on Unsplash

replace the default PHP module with FastCGI and configure Apache to work with MPM Event.
The following guide is tailored to Debian 11 and PHP 7.4, but you can adapt it by applying minor changes, which you can find in the corresponding steps.

Installing MPM Event

As mentioned earlier, the goal is to install MPM Event and then replace the default PHP module with FastCGI before configuring HTTP/2.
When the server is already running MPM Prefork, it is not possible to enable MPM Event simultaneously. You need to disable Prefork. Because Prefork is a PHP package dependency, you must disable PHP before disabling Prefork.
On the other hand, because you need to make several changes to the Apache HTTP Server, stopping Apache to ensure a seamless process is recommended.
Table 1 show the steps to install MPM Event on Apache. The PHP version varies according to the operating system version. Commonly, it is version 7.0 for Debian 9, version 7.2 for Debian 10, and version 7.4 for Debian 11, although it could be any previously installed or manually compiled version. You can find the current version by looking for it in the Apache `mods-enabled` folder:

```
ls /etc/apache2/mods-enabled/ | grep php
```

Note that if you get an empty answer, it could be because the PHP module is already disabled. In that case, verify whether `fcgid` is enabled instead.

Table 1: Install MPM Event

Step	Command
1. Stop Apache	`systemctl stop apache2`
2. Disable PHP	`a2dismod php7.4`
3. Disable MPM Prefork (if enabled)	`a2dismod mpm_prefork`
4. Enable MPM Event	`a2enmod mpm_event`
5. Restart Apache	`systemctl start apache2`

Installing FastCGI and PHP FastCGI Module

FastCGI is a high-performance communication protocol derived from the common gateway interface (CGI). Although the information handled by both interfaces is the same, FastCGI is faster than CGI because of how it processes the data. Although a CGI application starts and finishes with the beginning and end of each HTTP request, a FastCGI application starts only once, handling the subsequent HTTP requests without the need to start again.
Installing FastCGI on Apache HTTP Server to run PHP web applications requires four packages:

- The PHP FastCGI binary (note that it is not an Apache module but a PHP interpreter)
- The FastCGI module for Apache
- The Proxy module (see the "Important Proxy Module Note" box), an Apache requirement when using FastCGI
- The FastCGI Proxy module, another Apache requirement when using FastCGI

Table 2 shows the steps to install FastCGI to run PHP applications. As mentioned earlier, the PHP version varies according to the operating system version. Because FastCGI Proxy enables the Proxy module, you only need to enable the second to enable the first, as well.

Table 2: Install FastCGI

Step	Command
1. Install PHP FastCGI	`apt install php-fpm`
2. Install the Apache FastCGI module	`apt install libapache2-mod-fcgid`
3. Load the PHP FastCGI configuration	`a2enconf php7.4-fpm`
4. Enable Proxy and FastCGI Proxy modules	`a2enmod proxy_fcgi`
5. Restart Apache (or keep working until the end of this example)	`systemctl restart apache2`

Setting Up HTTP/2

Now, it's time to finish the entire process by setting up HTTP/2 on Apache in three steps (Table 3). It is important to note that you should not execute the steps in Table 3 if you have not previously installed MPM Event and FastCGI, as shown previously.
The HTTP/2 module shown in step 1 is already available in the Apache `/mods-available/` folder, but it is not enabled. No additional installation is needed. Just run the `a2enmod` Debian command.
The Apache `Protocols` directive sets one or more allowed HTTP protocols, as shown in Table 3, step 2. You will prefer both if you serve web applications under HTTPS and HTTP (recommended option), or choose only one in other cases.

Table 3: Set Up HTTP/2 on Apache

Step	Command
1. Enable the HTTP/2 module	`a2enmod http2`
2. Add (or modify) the Apache `Protocols` directive	
Default	`http/1.1`
HTTP/2 over TCP	`h2c`
HTTP/2 over TLS	`h2`
3. Restart Apache	`systemctl restart apache2`

Important Proxy Module Note

Ensure that the `ProxyRequests` directive remains `Off`. If it is not, Apache could be used as a forward proxy server, which would be unsafe for both your server and network if you do not control who can access it by using the `<Proxy>` control block [1].

The order of priority protocols is defined by the order in which you write the protocol values in the `Protocols` directive, prioritizing the server configuration over that of the client. If you do not want to do that, you can explicitly set the `ProtocolsHonorOrder` directive to `Off` so that the client preference will be prioritized over that of the server.
Both directives can be set at the server configuration or virtual host level, allowing you to serve some websites while enforcing a specific protocol order. For the purpose of this example, I set only the `Protocols` directive at the configuration file level. Open the Apache configuration file (commonly located in the path `/etc/apache2/apache2.conf`) and add the following instruction at the beginning:

```
Protocols h2 h2c http/1.1
```

This line indicates that HTTP/2 over TLS has precedence over HTTP/2 over TCP, which has precedence over HTTP/1.1.

Table 4: Reverting

Step	Action
1. Stop Apache to ensure a seamless reverting process	`systemctl stop apache2`
2. Delete the `Protocols` directive	Open the Apache config file (`/etc/apache2/apache2.conf`) and remove the line `Protocols h2 h2c http/1.1`
3. Disable the HTTP/2 module	`a2dismod http2`
4. Disable the Proxy module	`a2dismod proxy`
5. Disable FastCGI Proxy module	`a2dismod proxy_fcgi`
6. Remove the PHP FastCGI configuration file	`a2disconf php7.4-fpm` (remember to change the PHP version to your current number)
7. Disable MPM Event	`a2dismod mpm_event`
8. Enable MPM Prefork	`a2enmod mpm_prefork`
9. Enable the original PHP module	`a2enmod php7.4` (remember to change the PHP version to your current number)
10. Start Apache again	`systemctl start apache2`

After restarting Apache, you can test your websites with `curl` by forcing it to use the HTTP/2 protocol,

```
curl -I --http2 <URL>
```

where `<URL>` is the URL you want to test – or *http://localhost* if you have not yet configured one.

Troubleshooting and Reverting

If for any reason you need to revert the entire previous process, follow the step-by-step guide in Table 4. ■

Info

[1] Controlling access to your proxy: [https://httpd.apache.org/docs/2.4/mod/mod_proxy.html#access]

The Author

Eugenia Bahit (she/her) is a theoretical computer scientist and bilingual science writer specializing in Linux programming, software engineering, and high-performance computing.

■