Compiling PHP for Development

A quick guide on how to compile PHP (especially new beta versions before packaging) to be able to test software with it

> You have any feedback or ideas to improve this? Contact me on Social Media or per E-Mail. You can find my other projects at lw1.at.
>
>

Preparation

This guide assumes that you will download the source to `~/php/source` and install PHP to `~/php/php8`. It should also work on all Linux/MacOS system assuming you have the right dependencies installed.

If you are using Debian/Ubuntu, you need to have the build-essentials installed.

```
➜  sudo apt install build-essential
```

Fetching source

The names of the directories of course change depending on which version of PHP you are installing

```
→  cd ~/php/source
→  wget https://downloads.php.net/~ramsey/php-8.1.0beta2.tar.gz
→  tar xzf php-8.1.0beta2.tar.gz
→  rm php-8.1.0beta2.tar.gz
```

Configuration and Building

This assumes you are going to run PHP as your user (instead of `www-data` or similar as normaly in production).

Create a new file called `build.sh` inside the source directory with the following content:

```
#!/bin/bash

set -e
set -x

INSTALL_DIR=/home/lukas/php/php8
USER=$(whoami)

mkdir -p $INSTALL_DIR

./configure --prefix=$INSTALL_DIR \
    --enable-bcmath \
    --enable-fpm \
    --with-fpm-user="$USER" \
    --with-fpm-group="$USER" \
    --disable-cgi \
    --enable-mbstring \
    --enable-shmop \
    --enable-sockets \
    --enable-sysvmsg \
    --enable-sysvsem \
    --enable-sysvshm \
    --with-zlib \
    --with-curl \
    --without-pear \
    --with-openssl \
    --enable-pcntl \
    --with-password-argon2 \
    --with-sodium \
    --with-zip \
```

```
    --enable-mysqlnd \
    --with-pdo-mysql \
    --with-pdo-mysql=mysqlnd \
    --enable-gd \
    --with-freetype \
    --enable-opcache
```

This should enable all PHP modules Matomo needs to run. If you need more, adjust the lines above (and don't forget to end them with a `\`)

Now run this script:

```
➜  chmod +x build.sh
➜  ./build.sh
```

Afterwards we can start compiling PHP:

```
➜  make
# or if you want to use 6 processes in parallel
➜  make -j 6
```

You can now optionally run the tests (don't worry if a few of them fail in beta versions)

```
➜  make test
```

And the last step is to install PHP to our `INSTALL_DIR`:

```
➜  make install
```

Run php-cli

You can now run php-cli from `~/php/php8/bin/php` and also start the built-in server:

```
➜  echo "<?php var_dump('Hello World!');" | ~/php/php8/bin/php
string(12) "Hello World!"
➜  ~/php/php8/bin/php -S localhost:1234
```

If this is enough for you, you can stop here. Otherwise continue with setting up php-fpm.

Set up php-fpm

If you want to use the compiled PHP version from your webserver (e.g. Apache or Nginx), you need to use php-fpm.

```
➜  cd ~/php/php8
➜  cp etc/php-fpm.conf.default etc/php-fpm.conf
➜  cp etc/php-fpm.d/www.conf.default etc/php-fpm.d/www.conf
# adapt the two config files if needed
```

If you want to improve performance, you might want to increase `pm.max_children`, `pm.start_servers` and `pm.max_spare_servers`.

You can also create a `lib/php.ini` with custom settings:

```
display_startup_errors = Off
display_errors = Off
log_errors=1
error_log=/tmp/phperror
memory_limit = 512M
```

Now you can start php-fpm using

```
➜  ~/php/php8/sbin/php-fpm --nodaemonize
```

php-fpm will be listening on port 9000 by default.

Now you can update your webserver configuration to point to your php-fpm. How exactly this works, depends on your webserver. But in Nginx something like this will work:

```
location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass 127.0.0.1:9000;
}
```