# Настройка среды разработки Apache, PHP и HTTPS с помощью Docker

9 ноября 2022 г.
*1500 слов, 8-минутное чтение*

---

*В этом руководстве по Docker объясняется, как запустить PHP-приложение с использованием сертификатов Apache и real SSL на любом КОМПЬЮТЕРЕ для разработки Windows, Mac OS или Linux.*

---

PHP, возможно, не самая модная технология, но она используется многими разработчиками и проектами. Согласно W3Techs, PHP используется на 78% всех веб-сайтов. Возможно, это заниженная оценка, поскольку сайты не могут – и в идеале *не должны* – объявлять свой стек. Более достоверная статистика заключается в том, что WordPress поддерживает 43% веб-ресурсов, а CMS использует PHP.

Я редко берусь за новые PHP-проекты, но у меня много устаревших сайтов и приложений с папками, полными `.php` файлов. Установка PHP может занять много времени и привести к ошибкам. Существуют различные версии, и вы столкнетесь с дополнительными сложностями при интеграции PHP с веб-сервером, таким как Apache, для соответствия реальным решениям для хостинга.

Кроме того, пользователям Windows предлагается запутанный набор опций, хотя ситуация скоро станет проще – *Microsoft прекращает поддержку PHP в Windows*:

> *“Мы не собираемся поддерживать PHP для Windows в любом качестве для версии 8.0 и выше”.*

Кто-то, скорее всего, будет компилировать версии Windows, а подсистема Windows для Linux предоставляет другой вариант. Однако суть в том, что обслуживание одной или нескольких сред разработки PHP может быть затруднено…

...*если вы не используете Docker*.

# Зачем использовать Docker?

Docker - это инструмент, который может устанавливать, настраивать программное обеспечение и управлять им. Он создает оболочку вокруг исполняемых файлов, известную как *контейнер*. Контейнеры запускаются из предварительно настроенных *образов*, которые представляют собой снимок исполняемого файла и его библиотек.

*В моей книге и видеокурсе "Docker для веб-разработчиков" кратко объясняется, как использовать Docker для ваших новых и существующих проектов.*

Docker предоставляет готовые образы Apache и PHP, которые можно загрузить и запустить в любой ОС, где установлен Docker (см. Инструкции по установке Docker).

В следующих разделах описывается, как подготовить среду разработки Docker, которая может выполнять PHP-файлы, расположенные на вашем главном компьютере.

# Создайте SSL-сертификаты

Веб-приложения используют HTTPS для обеспечения того, чтобы связь между клиентом и сервером была зашифрована и не могла быть перехвачена. Google также наказывает сайты с контентом, которые остаются на HTTP.

Для локальной разработки разработчики либо:

1. Используйте HTTP
   Это означает, что локальная и производственная версии отличаются. Выявить проблемы, такие как привязка к небезопасным ресурсам, может быть сложнее.

2. Или используйте (поддельный) самозаверяющий сертификат
   Это ближе к рабочей версии, но браузер по-прежнему обрабатывает запросы по-другому. Например, поддельные ресурсы SSL не кэшируются.

Третьим менее известным вариантом является mkcert. Это создает новый локально доверенный центр управления и SSL-сертификаты. Что касается браузера, соединение HTTPS полностью безопасно, несмотря на то, что выполняется в локальном домене.

Настраивать сертификаты нужно только один раз, и создание их на вашем локальном компьютере также будет работать в контейнерах Docker или WSL2. Следуйте инструкциям по установке mkcert, затем установите новый локальный центр сертификации в своих браузерах:

```
mkcert -install
```

Firefox требует некоторой дополнительной настройки:

1. Найдите сгенерированный `rootCA.pem` файл, введя `mkcert -CAROOT` в свой терминал.
2. Откройте меню Firefox и выберите Параметры, затем Конфиденциальность и безопасность. Прокрутите страницу вниз и нажмите Просмотреть сертификаты. Выберите вкладку Authorities, нажмите Import ..., откройте `rootCA.pem` файл и перезапустите браузер.

Теперь создайте локально доверенные сертификаты разработки для вашего домена разработки:

```
mkcert localhost 127.0.0.1 ::1
```

Она проще в использовании `localhost`, но вы можете создать любое доменное имя, если на него есть ссылка в вашем `hosts` файле.

Переименуйте созданные файлы:

- `cert.pem` для SSL-сертификата и
- `cert-key.pem` для файла ключа SSL-сертификата

Создайте каталог где-нибудь в вашей системе, например, `dockerphp`, и скопируйте в него два `.pem` файла.

# Конфигурация Apache

Создайте файл с именем `000-default.conf` в том же каталоге со следующей конфигурацией Apache HTTP и HTTPS. Это установит для Web значение root `/var/www/html` и будет ссылаться на SSL-сертификаты, созданные вами с помощью mkcert:

```
<VirtualHost *:80>

  ServerAdmin admin@localhost
  DocumentRoot /var/www/html
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>

<VirtualHost *:443>

  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/cert.pem
  SSLCertificateKeyFile /etc/apache2/ssl/cert-key.pem

  ServerAdmin admin@localhost
  DocumentRoot /var/www/html
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

# Конфигурация Docker

Создайте файл с именем `Dockerfile` в своем каталоге и добавьте следующее содержимое для создания образа PHP и Apache. Вы можете выбрать

из десятков начальных образов в Docker Hub, но в этом примере используется `php:8-apache` последняя версия PHP 8 для Apache 2.4:

```
FROM php:8-apache

RUN a2enmod ssl && a2enmod rewrite
RUN mkdir -p /etc/apache2/ssl
RUN mv "$PHP_INI_DIR/php.ini-development" "$PHP_INI_DIR/php.ini"

COPY ./ssl/*.pem /etc/apache2/ssl/
COPY ./apache/000-default.conf /etc/apache2/sites-available/000-

EXPOSE 80
EXPOSE 443
```

`Dockerfile`:

1. Включает модули SSL и перезаписи Apache. При необходимости можно включить дополнительные модули.
2. Скопируйте файл конфигурации разработки PHP в `php.ini`, чтобы отображались ошибки и предупреждения.
3. Создает `/etc/apache2/ssl` каталог и копирует файлы `.pem` сертификатов SSL, созданные выше.
4. Скопируйте файл конфигурации Apache.
5. Предоставляет порты 80 и 443 для HTTP и HTTPS соответственно.

При необходимости вы можете определить свой собственный `php.ini` файл и `COPY` поместить его в изображение по адресу `/usr/local/etc/php/php.ini`.

---

*Примечание: отдельные `Dockerfile` `RUN` команды могут быть объединены в одну строку и разделены символом `&&`. Это ускоряет и повышает эффективность процесса сборки Docker, хотя код становится более сложным для чтения.*

---

# Создайте образ PHP Docker

Создайте образ Docker с именем `php8` из вашего `Dockerfile`, перейдя в каталог в терминале и введя:

```
docker image build -t php8 .
```

*(Последний `.` период важен!)*

Предполагая, что у вас нет ошибок, будет создан новый образ Docker. Запустите, `docker image ls` чтобы просмотреть `php8` список образов.

# Запустите PHP-контейнер

Теперь вы можете запустить контейнер Docker из `php8` изображения. Перейдите в любой каталог, содержащий PHP-проект, и выполните следующую `docker` команду:

```
docker run \
  -it --rm \
  -p 8080:80 -p 443:443 \
  --name php8site \
  -v "$PWD":/var/www/html \
  php8
```

---

*Пользователи Windows Powershell должны удалить разрывы строк и `\` обратную косую черту из команды. Кроме того, `$PWD` ссылается на текущий каталог в Linux и macOS. Это невозможно использовать в Windows, поэтому полный путь должен быть указан в нотации Linux, например*

```
-v /c/projects/mysite:/var/www/html
```

---

Контейнер будет продолжать выполняться до тех пор, пока он не будет остановлен с помощью `Ctrl` | `Cmd` + `C`.

В качестве альтернативы вам может оказаться проще запустить контейнер с помощью Docker Compose. Создайте новый `docker-compose.yml` файл в каталоге проекта PHP со следующим содержимым:

```
version: '3'
services:

  php8site:
    image: php8
    container_name: php8site
    volumes:
      - .//var/www/html
    ports:
      - "8080:80"
      - "443:443"
```

Затем контейнер Apache / PHP можно запустить из этого каталога с помощью:

```
docker-compose up
```

и остановился в другом терминале с:

```
docker-compose down
```

# Запустите PHP-код

Каталог узла, в котором запускается контейнер Docker, монтируется с привязкой к контейнеру в корне Apache `/var/www/html`. Стандартный порт `443` доступен для подключений по протоколу HTTPS, а порт `8080` перенаправляет на порт HTTP `80`, чтобы избежать конфликтов с такими приложениями, как Skype.

Вы можете протестировать выполнение PHP с помощью примера `index.php` файла:

```
<?php
  1 |  phpinfo();
```

Запустите ее в своем браузере по адресу `http://localhost:8080/` или `https://localhost/`. Версия HTTPS будет использовать mkcert SSL, но, в отличие от самозаверяющих сертификатов, браузер не будет выдавать предупреждение системы безопасности.

# Динамическая разработка Docker

Небольшое знание Docker - это все, что требуется для создания безопасной среды разработки Apache и PHP. Преимущества:

- вам не нужно было вручную загружать, устанавливать или настраивать дополнительное программное обеспечение
- ваша операционная система не изменилась – контейнер не может конфликтовать с другими версиями Apache или PHP, которые вы установили
- контейнер будет работать идентично на любой другой ОС без изменений.

Наконец…

*Вы не ограничены PHP и Apache! Docker может управлять любым сервером, языковыми средами выполнения, базами данных или другими программными зависимостями, необходимыми вашему проекту.*

Темы: apache php ssl

# Контакты

craig@dockerwebdev.com

комната чата курса

@craigbuckler