

## Требования

Установить docker на компьютер. Создать локальную среду для разработки. Докеризировать PHP-приложение с помощью Nginx и PHP-FPM

## Настройка Nginx

Мы начнем с веб-сервера, и, исходя из наших требований, это будет контейнер с официальным образом Nginx. Поскольку мы будем использовать Docker Compose, мы создадим следующий файл `docker-compose.yml`, который будет запускать последний образ Nginx и будет предоставлять его порт 80 для порта 8080

**web:**

**image:** nginx:latest

**ports:**

- "8080:80"

Запуск консольной команды:

```
docker-compose build && docker-compose up -d
```

При переходе в браузере по `localhost:8080` вы должны увидеть следующую страницу

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

Теперь, когда у нас есть сервер, давайте добавим немного кода. Сначала нам нужно обновить файл `docker-compose.yml`, чтобы смонтировать локальный каталог. Используем папку с именем `code`, которая находится в том же каталоге, что и файл `docker-compose.yml`, и она будет смонтирована как код корневой папки в контейнере.

```
web:
  image: nginx:latest
  ports:
    - "8080:80"
  volumes:
    - ./code:/code
```

Следующий шаг — сообщить Nginx, что эта папка существует.  
Давайте создадим следующий `site.conf` на том же уровне, что и файл `docker-compose.yml`:

```
server {
  index index.html;
  server_name php-docker.local;
  error_log /var/log/nginx/error.log;
  access_log /var/log/nginx/access.log;
  root /code;
}
```

Вот что мы определяем здесь  
`index.html` будет нашим индексом по умолчанию,  
имя сервера — `php-docker.local`, и он должен указывать (обновить  
файл `hosts`) на вашу среду Docker (`localhost`, если вы работаете в  
Linux, или машина `docker`, если вы работаете на Mac или Windows),  
мы указываем журналы ошибок как те, которые выставлены контейнером  
по умолчанию, чтобы мы видели ошибки в нашем журнале компоновки  
`docker`. , и, наконец, мы указываем корневую папку как ту, которую  
мы смонтировали в контейнере.  
Необходимо также добавить строку в файл `hosts`  
`127.0.0.1 php-docker.local`

Чтобы активировать эту настройку, нам нужно применить еще одну  
модификацию к нашему файлу `docker-compose.yml`:

```
web:
  image: nginx:latest
  ports:
    - "8080:80"
  volumes:
    - ./code:/code
```

Это добавит `site.conf` в каталог, где Nginx ищет файлы конфигурации  
сайтов. Теперь вы можете поместить файл `index.html` в папку кода с  
содержимым, которое вам по душе.

Запуск консольной команды:

```
docker-compose build && docker-compose up -d
```

файл index.html должен быть доступен на php-docker.local:8080

## Добавление PHP-FPM

Теперь, когда у нас запущен Nginx, давайте добавим PHP в игру. Первое, что мы сделаем, — это вытащим официальный репозиторий PHP7-FPM и свяжем его с нашим контейнером Nginx. Наш docker-compose.yml теперь будет выглядеть так:

```
web:
  image: nginx:latest
  ports:
    - "8080:80"
  volumes:
    - ./code:/code
    - ./site.conf:/etc/nginx/conf.d/site.conf
  links:
    - php
php:
  image: php:7-fpm
```

Следующее, что нужно сделать, это настроить Nginx для использования контейнера PHP-FPM для интерпретации файлов PHP. Ваш обновленный site.conf должен выглядеть так:

```
server {
    index index.php index.html;
    server_name php-docker.local;
    error_log /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root /code;

    location ~ /\.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

Чтобы проверить, необходимо переименовать файл index.html в index.php и заменить его содержимое стандартным:

```
<?php
echo phpinfo();
```



Поскольку PHP работает в своей собственной среде (контейнере), у него нет доступа к коду. Чтобы исправить это, нам также нужно смонтировать папку с кодом в контейнере PHP. Таким образом, Nginx сможет обслуживать любые статические файлы, а PHP сможет найти файлы, которые он должен интерпретировать. Последнее изменение в файле `docker-compose.yml`:

```
web:
  image: nginx:latest
  ports:
    - "8080:80"
  volumes:
    - ./code:/code
    - ./site.conf:/etc/nginx/conf.d/site.conf
  links:
    - php
php:
  image: php:7-fpm
  volumes:
    - ./code:/code
```

Запуск консольной команды:

```
docker-compose build && docker-compose up -d
```

При переходе по `php-docker.local:8080` мы увидим информацию об установленном php. Установка завершена.

<div> <div>PHP Version 7.4.27</div> <div>  </div> </div>	
System	Linux 712d2118f5e2 4.15.0-132-generic #136-Ubuntu SMP Tue Jan 12 14:58:42 UTC 2021 x86_64
Build Date	Dec 21 2021 21:35:39
Configure Command	'/configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-openssl' '--with-readline' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.NTS
PHP Extension Build	API20190902.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk
<div> <div>           This program makes use of the Zend Scripting Language Engine:            Zend Engine v3.4.0, Copyright (c) Zend Technologies         </div> <div>  </div> </div>	

## Configuration