**Требования**

Установить docker на компьютер. Создать локальную среду для разработки. Докеризировать PHP-приложение с помощью Nginx и PHP8-FPM. Работа включает в себя общее руководство и задание по вариантам.

**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



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

  - ./site.conf:/etc/nginx/conf.d/site.conf

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:8-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:8-fpm

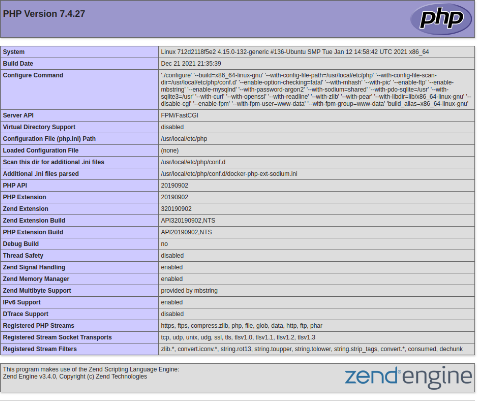
**volumes:**

        - ./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.



**Для каждого варианта необходимо:**

1. Создать docker-compose.yml
2. Настроить конфигурацию Nginx
3. Смонтировать необходимые тома
4. Создать требуемые HTML файлы и директории
5. Протестировать работу конфигурации

1. Докеризировать статическое HTML приложение с помощью Nginx согласно базовому примеру.

**2.** Настроить Nginx для работы с двумя разными доменами (multi-site): php-docker.local и php-docker1.local на порту 8080.

3. Настроить Nginx для работы с индексными файлами в порядке: index.htm, index.html, default.html.