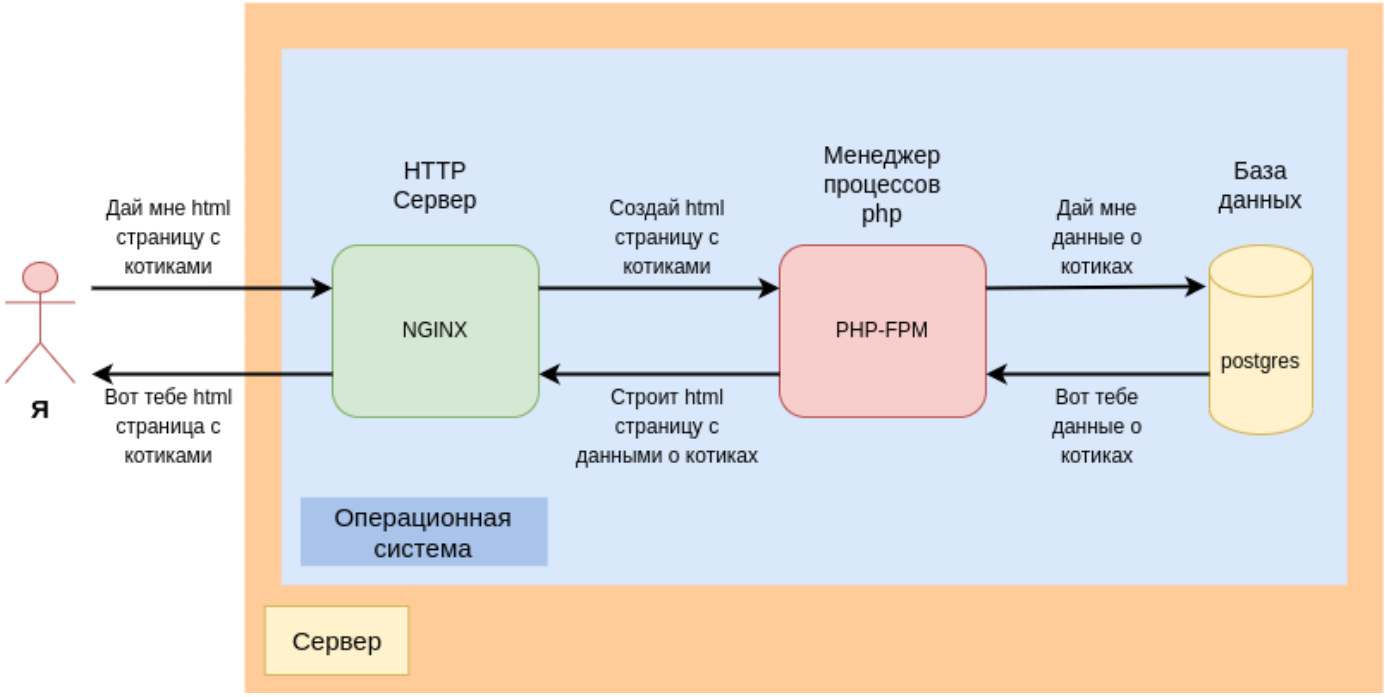
Docker для самых маленьких. php-fpm + nginx + postgres

Представьте что вы приходите в качестве разработчика в новую компанию у которого есть веб-сайт. Для его работы нужны допустим nginx, php-fpm и postgres, вот схематично как работает веб-сайт компании:



1. Вы вводите в браузере адрес веб-сайта, браузер запрашивает html-страницу с котиками по указанному адресу
2. http-сервер **nginx** принимает ваш запрос и делегирует создание страницы **php-fpm**
3. **php-fpm** запрашивает данные о котиках из базы **postgres**, строит html-страницу и отдает обратно его **nginx**-у, а тот в свою очередь клиенту-браузеру и вы видите красивых котиков

Введение в проблему

Перед началом работы над сайтом, вам придется развернуть проект - это значит:

- установить nginx
- установить php-fpm и все нужные расширения
- настроить совместную работу nginx и php-fpm
- установить postgres, создать пользователей и создать нужные базы и схемы

Зачастую это сделать не так просто и не так быстро, также сложности добавляют различия в операционных системах (далее ОС), один коллега предпочитает Mac OS, а другой Ubuntu или Windows.

Неплохо было бы автоматизировать эти рутинные действия и иметь одну магическую команду “установи мне всё, и где угодно и настрой как надо”, верно?

Передав такой инструмент коллеге, он сможет развернуть проект за считанные минуты, причём, его приложение будет работать в точности как и ваше. Всему этому есть решение - docker!

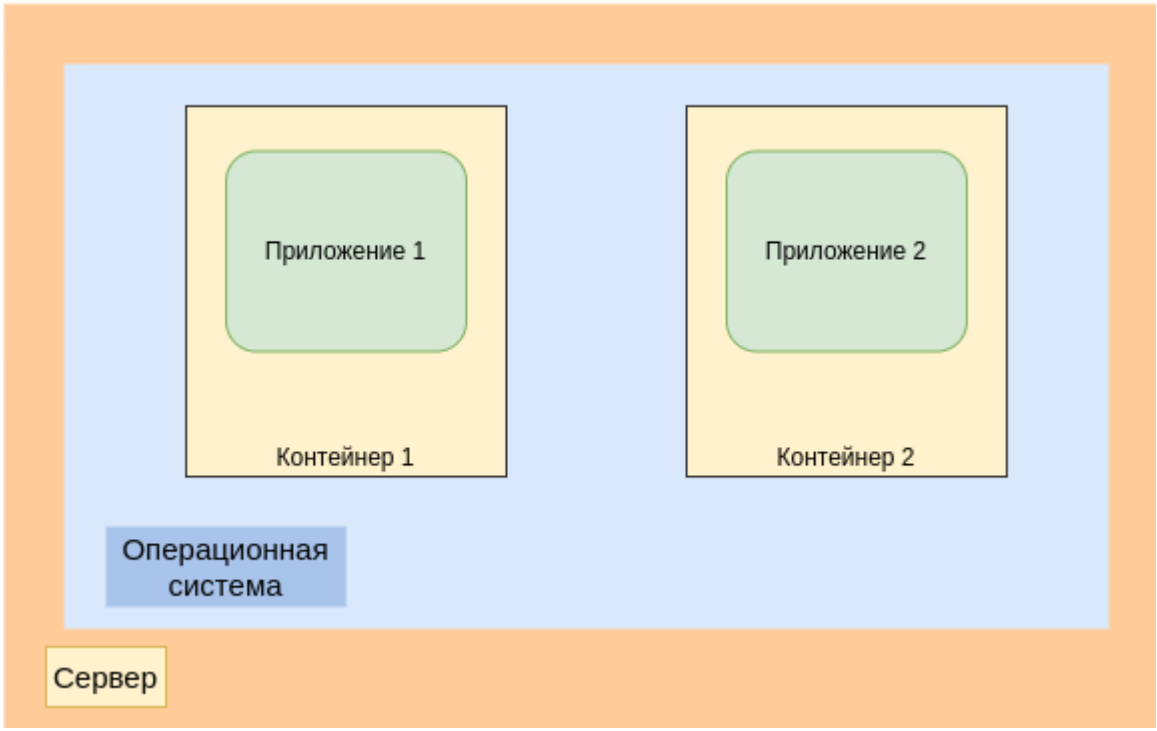
В этой статье мы разберёмся с основами работы docker, docker compose узнаем о контейнеризации и docker-контейнере, научимся скачивать готовые docker-образы и создавать свои, в конце развернём простое веб-приложение с использованием php-fpm + nginx + postgres.

Контейнеризация и docker-контейнер

В основе программы Docker лежит технология контейнеризации.

Контейнеризация (виртуализация на уровне ОС) - это технология, которая помогает запускать приложения изолированно от ОС. Приложение упаковывается в специальную оболочку-**контейнер**, внутри которой - среда, необходимая для работы.

Простыми словами **контейнер** - это некая изолированная песочница для запуска ваших приложений



Приложение 1 и *приложение 2* изолированы друг от друга и от операционной системы.

Docker - это программа, которая является наиболее популярной реализацией технологии контейнеризации, она позволяет запускать docker-контейнеры с приложениями из заранее заготовленных шаблонов - **docker-образов (Docker image)**

Для чего вам может понадобиться docker:

- запуск изолированных приложений, и управление ими
- ускорение и автоматизация развертывания приложений
- доставка этих приложений до серверов
- масштабирование
- запуск на одном компьютере разных версий одной программы

Установка

Для начала нам нужно установить:

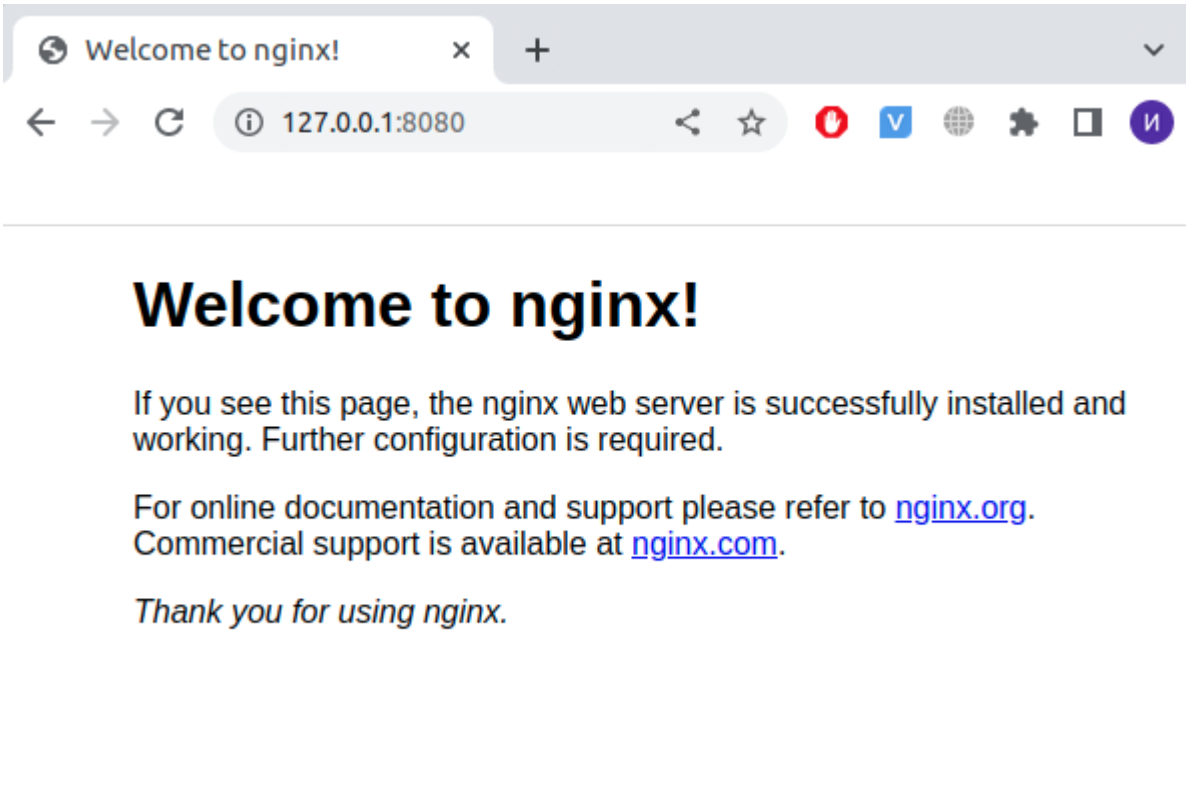
- docker (https://docs.docker.com/engine/install/)
- docker-compose (https://docs.docker.com/compose/install/)

Сразу в бой!

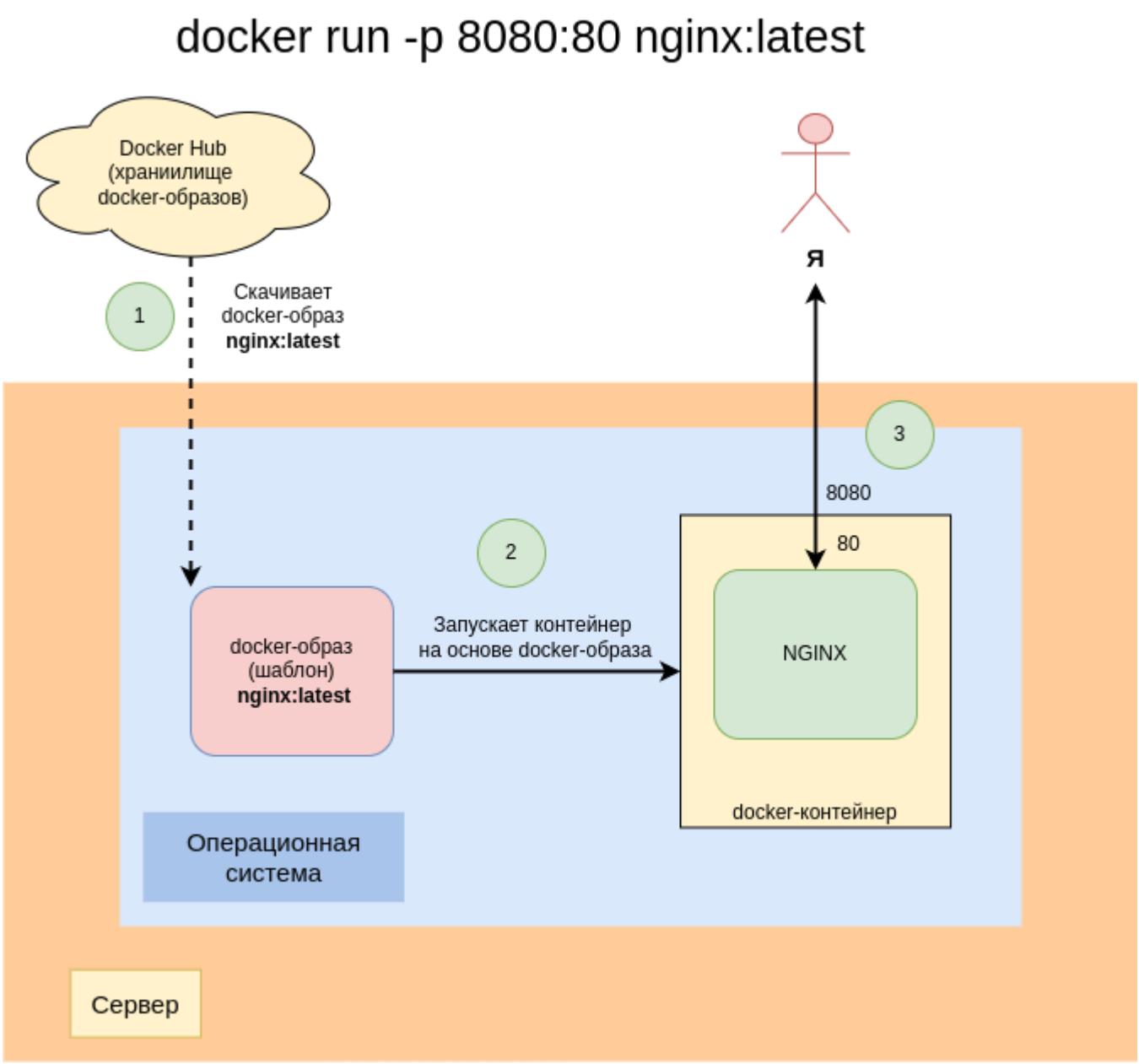
Концепцию docker легче понять на практике. Например, давайте попробуем запустить http-сервер `nginx` на нашем компьютере.

```
docker run -p 8080:80 nginx:latest
```

Идём в браузер, открываем `127.0.0.1:8080`. И уже видим страницу приветствия nginx! И всё!



Теперь разберёмся подробнее, что тут происходит.



Команда `docker run -p 8080:80 nginx:latest` делает следующее:

1. Скачивает docker-образ `nginx:latest` (если ранее не скачивался) из Docker Hub (https://hub.docker.com/)
2. Запускает docker-контейнер используя этот docker-образ
3. Ранее уже упоминалось, что процессы в docker-контейнерах запускаются в изоляции от ОС, то есть все порты между ОС и docker-контейнером закрыты. И для того чтобы мы смогли обратиться к nginx, нужно пробросить порт. Как раз для этого служит опция `-p 8080:80` , **80** - порт nginx-а внутри контейнера, **8080** - порт в локальной сети ОС.

Docker-образ (Docker image) - шаблон для создания docker-контейнеров. Представляет собой исполняемый пакет, содержащий все необходимое для запуска приложения: код, среду выполнения, библиотеки, переменные окружения и файлы конфигурации.

Docker Hub - это публичный репозиторий docker-образов, куда может любой желающий загрузить его или скачать. На странице nginx в Docker Hub (https://hub.docker.com/_/nginx?tab=tags) как раз можно найти тот самый наш docker-образ nginx:latest

(https://hub.docker.com/layers/nginx/library/nginx/latest/images/sha256-b6a3554b020680898ad2d36f2211e03154766cb9841bf46f64d6259b12c3af5c?context=explore), latest - это tag который ссылается на самый свежий docker-образ

Создание собственных docker-образов

А давайте попробуем создать свой docker-образ, взяв за основу `nginx:latest` ?

Docker умеет создавать docker-образ читая текстовые команды записанный в файл, этот файл называется **Dockerfile**

Пример простейшего Dockerfile:

```
FROM nginx:latest

RUN echo 'Hi, we are building custom docker image from nginx:latest!'

COPY nginx-custom-welcome-page.html /usr/share/nginx/html/index.html
```

- `FROM` - задаёт базовый (родительский) docker-образ, должен идти первой командой
- `COPY` - копирует в docker-контейнер файлы

С помощью команды `COPY` мы заменяем стандартную welcome-страницу nginx-а на:

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome to custom nginx page!</h1>
</body>
</html>
```

Подробнее об этих и других командах тут (https://docs.docker.com/engine/reference/builder/)

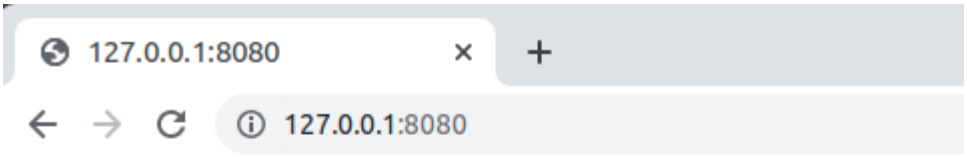
И давайте создадим наш docker-образ из Dockerfile:

```
$ docker build -t nginx_custom:latest -f /opt/src/docker-for-kids/dockerFiles/nginx-custom/Dockerfile /opt/src/docker-for-kids
Sending build context to Docker daemon  139.3kB
Step 1/3 : FROM nginx:latest
latest: Pulling from library/nginx
31b3f1ad4ce1: Pull complete
fd42b079d0f8: Pull complete
30585fbbebc6: Pull complete
18f4ffdd25f4: Pull complete
9dc932c8fba2: Pull complete
600c24b8ba39: Pull complete
Digest: sha256:0b970013351304af46f322da1263516b188318682b2ab1091862497591189ff1
Status: Downloaded newer image for nginx:latest
 ---> 2d389e545974
Step 2/3 : RUN echo 'Hi, we are building custom docker image from nginx:latest!'
 ---> Running in 05ffd060061f
Hi, we are building custom docker image from nginx:latest!
Removing intermediate container 05ffd060061f
 ---> 9ac62be4252a
Step 3/3 : COPY nginx-custom-welcome-page.html /usr/share/nginx/html/index.html
 ---> 704121601a45
Successfully built 704121601a45
Successfully tagged nginx_custom:latest
```

- `-t nginx_custom:latest` - это имя будущего docker-образа, latest - это tag
- `-f /opt/src/docker-for-kids/dockerFiles/nginx-custom/Dockerfile` - путь до Dockerfile
- `/opt/src/docker-for-kids` - директория в контексте которого будет создан docker-образ, процесс создания docker-образа может ссылаться на любой из файлов в контексте. Например, команда COPY

И запустим:

```
$ docker run -p 8080:80 nginx_custom:latest
```



Welcome to custom nginx page!

Замечательно, у нас удалось создать свой docker-образ и запустить его!

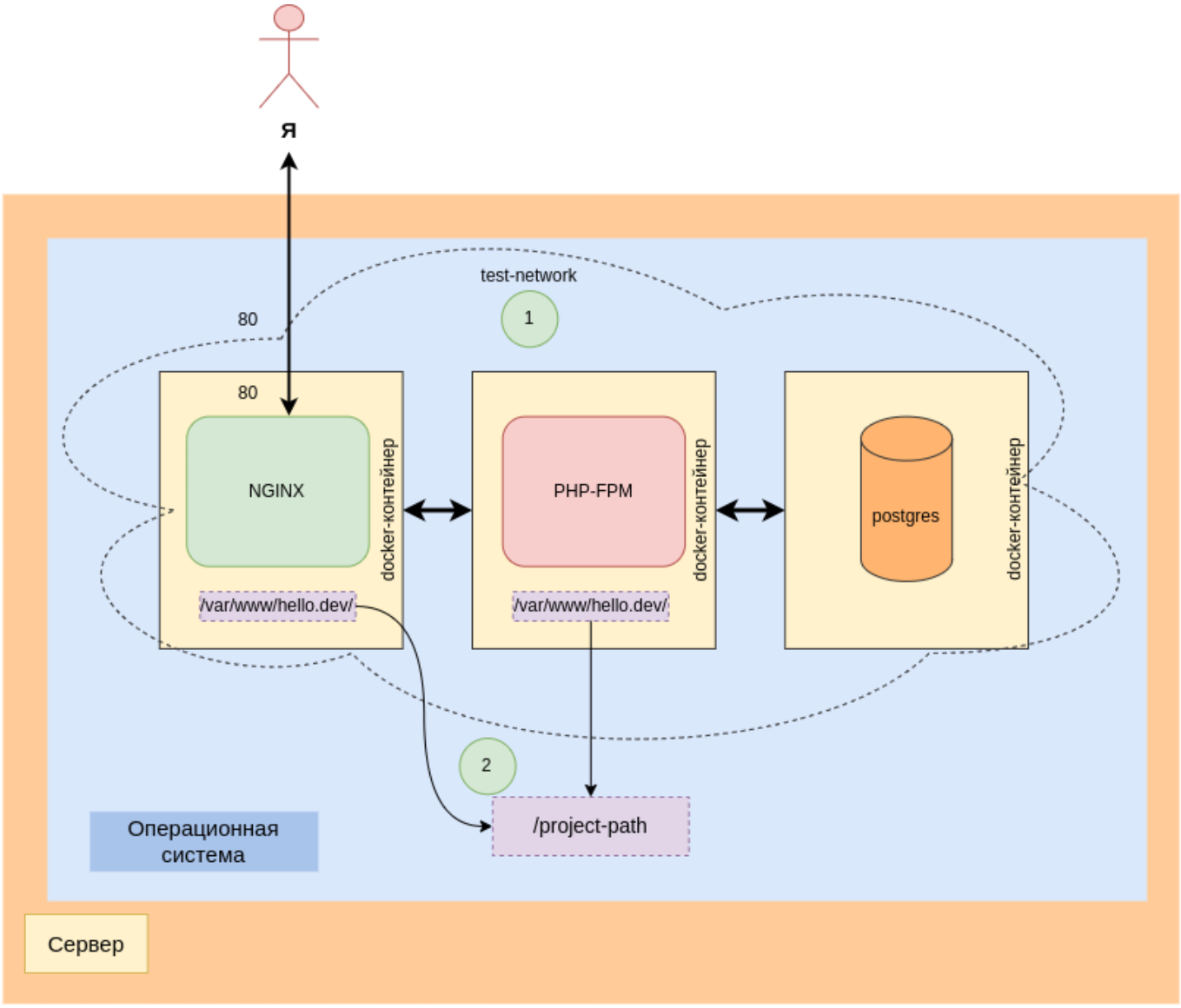
Docker compose

С ростом количества docker-контейнеров поддержка их становится затруднительным. И справится с этим нам поможет Docker compose.

Docker compose - это инструмент для описания и запуска многоконтейнерных приложений. Для описания используется YAML файл.

```
version: '3'

services:
  nginx:
    container_name: nginx-test # имя docker-контейнера
    build: # создать docker-образ из dockerFile
      context: . # путь в контексте которого будет создан docker-образ
      dockerfile: ./dockerFiles/nginx/Dockerfile # путь до dockerFile из которого будет собран
docker-образ
    ports: # проброс портов
      - "80:80"
    networks: # имя сети к котором будет подключен docker-контейнер
      - test-network
    depends_on: # данный сервис будет запущен только после запуска сервиса под именем php-fpm
      - php-fpm
    volumes: #  монтирование директорий, директория-на-хост-машине:директория-в-докере
      - ./:/var/www/hello.dev/
  php-fpm:
    container_name: php-fpm-test
    build:
      context: .
      dockerfile: ./dockerFiles/php-fpm/Dockerfile
    networks:
      - test-network
    volumes:
      - ./:/var/www/hello.dev/
  postgres:
    container_name: postgres-test
    image: postgres:14.1-alpine # тэг docker-образа из https://hub.docker.com/
    environment:
      POSTGRES_PASSWORD: mysecretpass # переменные окружения которые использует docker-контейн
ep
    networks:
      - test-network
networks: # явно объявленные сети
  test-network:
    driver: bridge
```

Давайте разбираться что тут происходит и посмотрим на рисунок.



Каждый сервис у нас находится внутри docker-контейнера. Точкой входа в наше приложение как и в случае с веб-сайтом компании, является NGINX. Пользователи веб-сайта делают запросы к NGINX у которого проброшен порт 80.

1. `test-network` . Мы помним что каждое приложение в docker-контейнере находится в изоляции. `test-network` объединяет все docker-контейнеры в одну сеть, и это позволяет обращаться к нужному контейнеру по имени сервиса.
2. **Volumes** - это механизм для хранения данных вне docker-контейнера, т.е в файловой системе нашей ОС. И решает проблему совместного использования файлов.

Все примеры, а так же исходники dockerFile-ов можно взять из репозитория на https://github.com/ildarsaitkulov/docker-for-kids (https://github.com/ildarsaitkulov/docker-for-kids)

Простое веб-приложение

Создадим самое простое веб-приложение, которое показывает нам сообщение об успешном подключении к базе данных. Вместо адреса базы данных мы используем `host= postgres` , такое же имя нашего сервиса как и в YAML-файле, помним что эта возможность появилась благодаря общей сети `test-network`

`index.php`

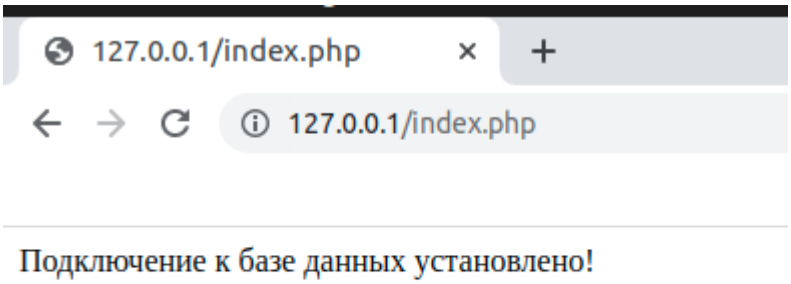
```
<?php

try {
    $pdo = new \PDO("pgsql:host=postgres;dbname=postgres", 'postgres', 'mysecretpass');
    echo "Подключение к базе данных установлено! <br>";

    return;
} catch (PDOException $exception) {
    echo "Ошибка при подключении к базе данных<br><b>{$exception->getMessage()}</b><br>";
}
```

PDO - это интерфейс для доступа к базам данных в PHP, подробнее (https://www.php.net/manual/en/intro.pdo.php).

Теперь, для того чтобы создать все docker-образы и запустить docker-контейнеры нужно выполнить:

```
docker-compose up --build
```

Выполним наш index.php



Увидим успешное соединение с базой данных!

Заключение

Как и обещал одной лишь командой мы развернули все сервисы, и это можно сделать где угодно, нужен только docker, и везде у вас будет единое окружение!

Придя на новую работу, вместо долгой и сложной подготовки проекта к работе, вам стоит лишь выполнить одну команду.

В этой статье мы разобрались с основами работы docker, docker compose узнали о контейнеризации и docker-контейнере, научились скачивать готовые docker-образы и создавать свои, развернули простое веб-приложение с использованием php-fpm + nginx + postgres.

Docker отличный инструмент для быстрого развертывания приложений, доставки до серверов, тестирования. Подробнее можно почитать на официальном сайте (https://www.docker.com/).

Веб-приложение для самостоятельного запуска можно найти по ссылке https://github.com/ildarsaitkulov/docker-for-kids (https://github.com/ildarsaitkulov/docker-for-kids)

saitkulovim@gmail.com (mailto:saitkulovim@gmail.com)
Github (https://github.com/ildarsaitkulov)
@saitkulovim (https://t.me/saitkulovim)