

## Автоматизация создания и запуска контейнеров с помощью docker-compose

*Вихляев Дмитрий Романович*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### Аннотация

Данная статья содержит описание метода автоматизации процесса создания и запуска группы контейнеров с помощью docker-compose. Для реализации используется сервис Docker и конфигурационные файлы Dockerfile и docker-compose.yml. В результате исследования будут созданы два контейнера Nodejs и PostgreSQL взаимодействующие через имена сервисов.

**Ключевые слова:** Docker, docker-compose, контейнер.

## Automating the creation and launch of containers using docker-compose

*Vikhlyayev Dmitry Romanovich*

*Sholom-Aleichem Priamursky State University*

*Student*

### Abstract

This article describes a method for automating the process of creating and launching a container group using docker-compose. The Docker service and the Dockerfile and docker-compose.yml configuration files are used for implementation. As a result of the study, two containers Node js and PostgreSQL will be created that interact through service names.

**Keywords:** Docker, docker-compose, container.

## 1 Введение

### 1.1 Актуальность

Современные подходы к разработке и развертыванию программного обеспечения требуют высокой скорости и гибкости при управлении сложными приложениями, состоящими из множества взаимосвязанных компонентов. Контейнеризация, ставшая основой для таких процессов, обеспечивает стандартизированную изоляцию приложений и их зависимостей, что облегчает переносимость и масштабируемость решений. В этом контексте docker-compose выступает важным инструментом, способным автоматизировать создание, настройку и запуск много контейнерных приложений, что значительно упрощает работу разработчиков и системных администраторов. Актуальность использования docker-compose обусловлена стремлением к упрощению инфраструктурного управления и повышению эффективности DevOps-процессов. docker-compose позволяет декларировать

конфигурации много контейнерных приложений в виде простого и понятного YAML-файла, что упрощает понимание и поддержку инфраструктуры. Это особенно важно в условиях сложных сред, где необходимо согласовать взаимодействие между множеством контейнеров, включая базы данных, веб-серверы, брокеры сообщений и другие службы.

### **1.2 Обзор исследований**

В.Ф.Захарчёнок, А.Н.Бизюк исследовали контейнеризацию и развертывание приложений с помощью Docker и docker-compose [1]. А.Ю.Ковалёв, А.В.Воруев описали инструмент управления контейнеризированными приложениями docker-compose [2]. Д.Я.Смирнов, А.Д. Ишкова, О.Г. Яскевич провели научную работу по отладке, тестированию и развёртыванию приложений при помощи Docker [3]. М.К.Солонько, К.А.Краснов в своей работе продемонстрировали структурную составляющую Docker-контейнеров [4]. Б.Ю.Егоров, Е.В.Таций, И.Ю.Анохина использовали Docker для создания контейнера python приложения [5].

### **1.3 Цель исследования**

Цель исследования – описать способ создания и запуска нескольких сервисов через один конфигурационный файл.

## **2 Материалы и методы**

Для работы используется программа Docker Desktop и файл с декларативным описанием инструкций формата «yaml».

## **3 Результаты и обсуждения**

Программа Docker предоставляет инструменты для упрощения создания и управления контейнерами. Кроме того, возможно одновременное создание и запуск множества сервисов, каждый из которых будет работать в собственном контейнере. Причём запуск осуществляется всего одной командой, а после создания, каждый контейнер будет иметь свои собственные настройки. Такой метод требует файла с описанием того как нужно всё собрать. Так называемое декларативное описание того как каждый контейнер должен быть настроен. Docker имеет инструмент docker-compose, который читает файл docker-compose.yml и выполняет описанные в нём действия.

В данном примере создаётся контейнер с программой на Nodejs, из базового образа Nodejs, установленного с Docker Hub. Приложение взаимодействует с базой данных Postgresql находящегося в контейнере готового образа. Взаимодействие осуществляется по имени сервиса.

Простое приложение подключается к базе данных и делает три запроса. Сначала создаёт таблицу, затем добавляет запись и в итоге выводит содержимое таблицы в консоль (рис 1,2).

```
const { Pool } = require("pg");
require("dotenv").config();

const pool = new Pool({
  user: process.env.DB_USER,
  host: process.env.DB_HOST,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
  port: process.env.DB_PORT,
});

module.exports = pool;
```

Рис. 1. Подключение к базе данных

```
const pool = require('./db');

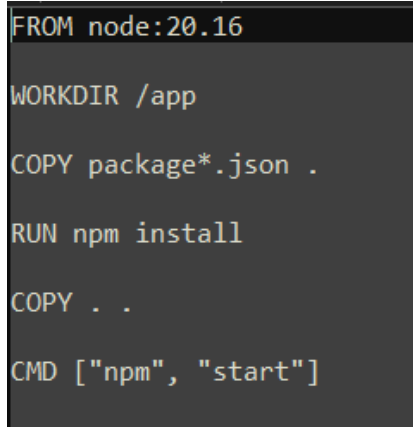
function dbQuery(query)
{
  return pool.query(query)
    .then((res) => Promise.resolve(res.rows))
    .catch((err) => Promise.reject(err));
}

const querySelect = `SELECT * FROM datetable`;
const queryInsert = `INSERT INTO datetable (name, date) VALUES ('first note', NOW())`;
const queryCreate = `
  CREATE TABLE IF NOT EXISTS "datetable" (
    id SERIAL PRIMARY KEY,
    name VARCHAR(15),
    date TIMESTAMP
  );
`;

dbQuery(queryCreate)
  .then(() => dbQuery(queryInsert))
  .then(() => dbQuery(querySelect))
  .then((rows) => console.log(rows))
  .catch((err) => console.error(err))
  .finally(() => pool.end()); // Закрыть соединения
```

Рис. 2. Отправка запросов к базе данных

Чтобы создать контейнер, прежде нужно смонтировать образ. В файле Dockerfile описывается инструкция для создания образа. В качестве базового образа используется «node:20.16». Далее следуют инструкции для создания новых слоёв. Создание рабочей директории установка зависимостей. В текущем примере, модуль «pg» для взаимодействия с PostgreSQL. Копирование оставшихся файлов проекта и команда запуска приложения (рис.3).



```
FROM node:20.16

WORKDIR /app

COPY package*.json .

RUN npm install

COPY . .

CMD ["npm", "start"]
```

Рис. 3. Инструкция создания нового образа

Различие между использованием Docker и docker-compose, в том, что последний позволяет одновременно управлять несколькими контейнерами, объединяя их в группу. Это даёт возможность запускать одновременно несколько контейнеров, используя одну команду. Точно также все запущенные контейнеры можно остановить и удалить одной командой.

Файл docker-compose.yml может содержать списки и словари. У ключа в качестве значения может быть строка, список или словарь. Вначале указывается версия yml файла. Раздел, в котором будут описаны сервисы, начинается с ключом «services». Далее указываются словари с ключом названия сервиса, в примере postgres и node. Для сервиса нужно указать название образа, который либо сначала будет искатся на локальной машине, затем в удалённом хранилище Docker Hub. Если нужно использовать собственный образ из инструкции Dockerfile, вместо ключа «image» указывается «build» со значением путь каталога с файлом. Также можно указать переменные окружения, которые будут автоматически подхвачены приложением. СУБД Postgresql предполагает, что определённые ключи для установки имени пользователя, пароля и базы данных. В пользовательском приложении Nodejs названия ключей определены в описании программы. Удобство в том, что не нужно создавать файл .env все инструкции сервиса указываются в одном файле. В качестве хоста для сервиса node указывается postgres. Это название сервиса базы данных. Контейнеры в Docker могут по имени взаимодействовать между собой без необходимости переброса портов в локальную сеть.

Поскольку контейнеры запускаются практически одновременно, а приложение Nodejs, сразу же обращается за подключением к базе данных, есть большая вероятность, что Postgresql не успеет подготовиться к запросам. Для этого устанавливается проверка «healthcheck» с описанием тестов. К сервису node добавляется зависимость, которая не запускает контейнер, пока тест не будет пройден (рис.4).

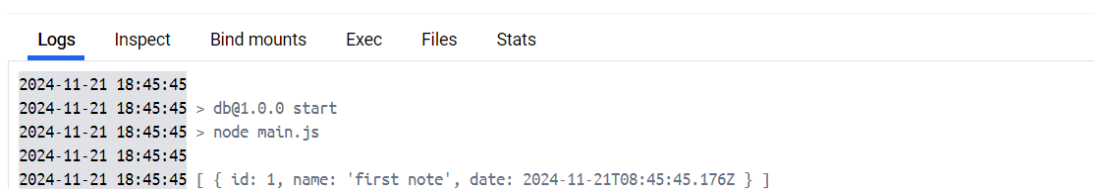
```
version: "3.8"

services:
  postgres:
    image: postgres:13.3
    environment:
      POSTGRES_DB: "meteo"
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "pass"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 10s
      timeout: 5s
      retries: 5

  node:
    build: "./app"
    environment:
      DB_HOST: "postgres" # Хост PostgreSQL – имя сервиса
      DB_PORT: 5432       # Порт PostgreSQL
      DB_NAME: "meteo"
      DB_USER: "postgres"
      DB_PASSWORD: "pass"
    depends_on:
      postgres:
        condition: service_healthy
```

Рис. 4. Содержание файла docker-compose.yml

Чтобы запустить сервисы нужно выполнить команду `docker-compose up` с указанием пути к `yml` файлу. Начнётся загрузка и сборка образов, а затем развёртывание контейнеров. Для закрытия и удаления сервисов нужно выполнить команду `docker-compose down` с указанием пути к `yml` файлу. В начале запуска, контейнер `node` находится в ожидании, и только когда тест, свидетельствующий о готовности базы данных отвечать на запросы, пройдёт сервис начнёт работу. Результат работы сервиса представлен на рисунке 5.



```
Logs    Inspect  Bind mounts  Exec  Files  Stats
2024-11-21 18:45:45
2024-11-21 18:45:45 > db@1.0.0 start
2024-11-21 18:45:45 > node main.js
2024-11-21 18:45:45
2024-11-21 18:45:45 [ { id: 1, name: 'first note', date: 2024-11-21T08:45:45.176Z } ]
```

Рис. 5. Выполнение работы сервиса `node`

В результате данной статьи, был описан способ автоматизации создания и запуска группы контейнеров. Представлены примеры создания сервисов и их взаимодействия в программе на Nodejs и СУБД PostgreSQL. Рассмотрен алгоритм монтирования собственного образа. Продемонстрированы особенности настройки и результаты работы сервисов.

## Библиографический список

1. Захарчёнок В.Ф., Бизюк А.Н. Контейнеризация и развёртывание приложений с помощью Docker и docker-compose // В сборнике: Материалы докладов 56-й международной научно-технической конференции преподавателей и студентов. в двух томах. Витебск, 2023. С. 80-82.

2. Ковалёв А.Ю., Воруев А.В. Инструмент управления контейнеризированными приложениями docker-compose // В сборнике: Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. Материалы XXIV Республиканской научной конференции студентов и аспирантов. Министерство образования Республики Беларусь; Учреждение образования «Гомельский государственный университет имени Франциска Скорины». 2021. С. 228-229.
3. Смирнов Д.Я., Ишкова А.Д., Яскевич О.Г. Отладка, тестирование и развёртывание приложений при помощи Docker // В сборнике: Интеллектуальные информационные системы. Труды Международной научно-практической конференции. Воронеж, 2023. С. 152-154.
4. Солонько М.К., Краснов К.А. Docker-контейнеры // Телекоммуникации и информационные технологии. 2020. Т. 7. № 1. С. 65-69.
5. Егоров Б.Ю., Таций Е.В., Анохина И.Ю. Докеризация python приложения // В сборнике: Современные информационные технологии в образовании и научных исследованиях (СИТОНИ-2023). Сборник материалов VIII Всероссийской научно-технической конференции. Донецк, 2023. С. 98-102.
6. Титанов П.О. Преимущества использования docker контейнеров в разработке ПО // В сборнике: Проблемы управления в социально-экономических и технических системах. Материалы XX Международной научно-практической конференции. Сборник научных статей. Саратов, 2024. С. 453-455.
7. Мареев Н.А. Сравнение контейнерных оркестраторов для реализации микросервисной архитектуры // В сборнике: Технологии информационного общества. Сборник трудов XVII Международной отраслевой научно-технической конференции. 2023. С. 228-230.