

Создание контейнера PostgreSQL с расширением PostGIS в Docker

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе рассмотрен процесс развёртывания и проверки контейнеризированной пространственной базы данных PostgreSQL с расширением PostGIS с использованием инструментария Docker. Проводится разбор структуры конфигурационного файла `docker-compose.yml`, включающего параметры переменные среды для инициализации базы данных, проброс портов и политику автоматического перезапуска контейнера. Результатом работы станет детальное руководство по автоматизации развёртывания переносимой среды пространственной СУБД, пригодной для выполнения геоинформационного анализа.

Ключевые слова: Docker, docker-compose, PostGIS.

Creating a PostgreSQL container with the PostGIS extension in Docker

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper discusses the process of deploying and verifying a PostgreSQL containerized spatial database with the PostGIS extension using the Docker toolkit. The structure of the docker-compose configuration file is being analyzed. A bml that includes environment variable parameters for database initialization, port forwarding, and an automatic container restart policy. As a result of the research, an autonomous, portable and ready-to-use spatial database environment suitable for performing geoinformation analysis will be obtained.

Keywords: Docker, docker-compose, PostGIS.

1 Введение

1.1 Актуальность

В научных исследованиях и прикладных задачах PostGIS зарекомендовал себя как мощное средство для пространственного анализа, позволяющее выполнять операции над геометрическими и географическими объектами непосредственно в базе данных. Его применение охватывает широкий спектр направлений: от урбанистики и транспортного моделирования до экологического мониторинга и пространственной эпидемиологии. Интеграция с Docker не только ускоряет развёртывание таких

решений, но и способствует лучшему управлению версиями, масштабированию в распределённых средах и автоматизации процессов анализа. Использование контейнеризированного PostGIS является актуальным и обоснованным выбором в современных условиях, объединяя в себе принципы DevOps, reproducible research и геоинформационной аналитики, и способствует эффективному решению задач, связанных с пространственными данными, в исследовательских, образовательных и промышленных контекстах.

1.2 Обзор исследований

В.Ф.Захарчёнок, А.Н.Бизюк исследовали контейнеризацию и развертывание приложений с помощью Docker и docker-compose [1]. Н.М.Лавров, Л.В.Селина, И.А.Платунов описали создание сервиса с применением PostGIS для картографирования корпусов университета [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]. Д.Ю.Овсянникова спроектировала геоинформационную систему и разработала модуль с использованием PostgreSQL, PostGIS и Qgis [5].

1.3 Цель исследования

Цель исследования – демонстрация последовательности шагов по созданию контейнера PostgreSQL и расширения PostGIS с помощью инструментов Docker и Docker Compose.

2 Материалы и методы

Для работы используется программа Docker и встроенный инструмент Docker Compose, официальный образ postgis/postgis:16-3.5-alpine, содержащий PostgreSQL 16 и PostGIS 3.5 на базе Alpine Linux.

3 Результаты и обсуждения

Первым шагом является создание текстового файла docker-compose.yml в корневом каталоге проекта. В данном файле указывается версия схемы Compose, определённая директивой version: '3.8', что указывает на использование совместимого формата конфигурации, поддерживаемого большинством актуальных версий Docker Compose. Далее следует секция services, в которой определяется набор сервисов, каждый из которых представляет собой контейнер или группу контейнеров. В данном случае определяется единственный сервис под названием postgis. В параметре image задаётся Docker-образ, на базе которого будет, развёрнут контейнер. Образ postgis/postgis:16-3.5-alpine представляет собой лёгкую версию PostgreSQL 16 с установленным расширением PostGIS 3.5, основанную на Alpine Linux для минимального размера.

Далее, параметр `container_name` задаёт человекочитаемое имя контейнера, что упрощает идентификацию при использовании командной строки Docker. Внутри ключа `environment` перечислены переменные среды, передаваемые в контейнер при его запуске. Эти переменные определяют имя пользователя (`POSTGRES_USER: user`), пароль (`POSTGRES_PASSWORD: 123`) и имя создаваемой по умолчанию базы данных (`POSTGRES_DB: gis_db`). Секция `ports` содержит список сопоставлений портов между хост-машиной и контейнером. Строка - `"5432:5432"` указывает, что порт PostgreSQL по умолчанию (5432) в контейнере будет проброшен на такой же порт на хостовой системе, что позволяет подключаться к базе данных извне, например, через `psql` или `pgAdmin`. Ключ `restart: unless-stopped` задаёт политику автоматического перезапуска контейнера, указывая на необходимость его автоматического восстановления после сбоев или перезагрузки хоста, если только контейнер не был явно остановлен пользователем (рис.1).

```
version: '3.8'

services:
  postgis:
    image: postgis/postgis:16-3.5-alpine
    container_name: postgis
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: 123
      POSTGRES_DB: gis_db
    ports:
      - "5432:5432"
    restart: unless-stopped
```

Рис. 1. Файл инструкций `docker-compose.yml`

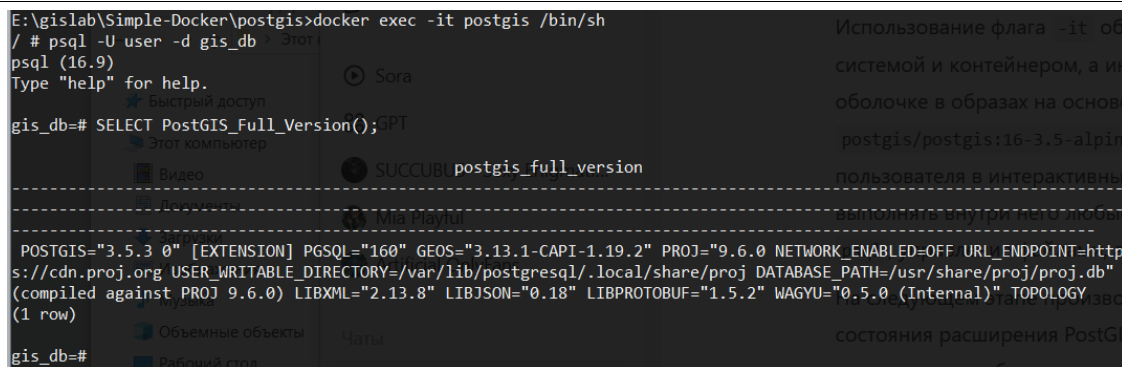
После сохранения данного файла необходимо запустить процесс сборки и запуска контейнера с помощью команды `docker-compose up -d`. Флаг `-d` переводит выполнение в фоновый режим (`detached mode`), что позволяет использовать терминал параллельно с выполнением контейнера. В процессе запуска Docker скачает указанный образ (если он отсутствует локально), создаст тома и сетевые подключения, инициализирует базу данных с заданными параметрами, а также активирует расширение PostGIS, встроенное в образ (рис.2).

```
E:\gislab\Simple-Docker\postgis>docker-compose up -d
time="2025-07-01T16:07:15+10:00" level=warning msg="E:\\gislab\\Simple-Docker\\postgis\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 15/15
  postgis Pulled
    a7b13a32aa7c Download complete 47.7s
    566700a5ef5b Download complete 1.4s
    dc0894634161 Download complete 1.5s
    9808eed8d51a Download complete 1.5s
    c6f99ba99948 Download complete 35.5s
    bef088cae706 Download complete 0.4s
    b2c2e77c01f8 Download complete 1.4s
    0e3293a8957f Download complete 2.5s
    d259b0545f3a Download complete 1.3s
    fe07684b16b8 Download complete 1.5s
    4dff379ecd5a Download complete 2.1s
    2d7ac0aea22 Download complete 1.4s
    6a8b2516865a Download complete 23.8s
    15367ac26b77 Download complete 1.5s
[+] Running 2/2
```

Рис. 2. Сборка контейнера в фоновом режиме

После разворачивания контейнера PostGIS на базе Docker Compose следующим необходимым этапом является проверка его работоспособности. Данная проверка включает в себя как удостоверение факта успешного запуска контейнера, так и функциональную верификацию доступности интерпретатора команд и возможностей расширения PostGIS. Первый шаг заключается в установлении связи с контейнером и доступе к его внутренней среде. Для этого используется команда `docker exec -it postgis /bin/sh`, которая инициирует запуск командного интерпретатора в интерактивном режиме внутри уже запущенного контейнера с именем `postgis`. Использование флага `-it` обеспечивает передачу ввода-вывода между хостовой системой и контейнером, а интерпретатор `/bin/sh` соответствует стандартной оболочке в образах на основе Alpine Linux, к которому относится и образ `postgis/postgis:16-3.5-alpine`. Успешное выполнение данной команды приведёт пользователя в интерактивный командный интерфейс контейнера, что позволяет выполнять внутри него любые системные команды, включая вызов SQL-клиента `psql`, управление файлами, а также диагностику процессов.

На следующем этапе производится проверка доступности PostgreSQL и активного состояния расширения PostGIS. Для этого, находясь внутри контейнера, следует подключиться к базе данных с помощью встроенного клиента PostgreSQL командой `psql -U user -d gis_db`. Здесь `-U` указывает имя пользователя, соответствующее переменной `POSTGRES_USER`, а `-d` задаёт имя базы данных, определённое переменной `POSTGRES_DB` в конфигурации `docker-compose.yml`. При корректном выполнении подключения пользователь попадает в SQL-интерпретатор, в рамках которого возможно выполнение запросов к системе управления базами данных. Ключевым подтверждением наличия расширения PostGIS является выполнение SQL-запроса `SELECT PostGIS_Full_Version();`, возвращающего строку с информацией о версии и конфигурации пространственного расширения. Ответ вида `POSTGIS="3.5.0" PGSQL="16.2" ...` свидетельствует об успешной интеграции PostGIS в экземпляр PostgreSQL (рис.3).



```
E:\gislab\Simple-Docker\postgis>docker exec -it postgis /bin/sh
/# psql -U user -d gis_db
psql (16.9)
Type "help" for help.
gis_db=# SELECT PostGIS_Full_Version();
PostGIS="3.5.3 0" [EXTENSION] PGSQL="160" GEOS="3.13.1-CAPI-1.19.2" PROJ="9.6.0 NETWORK_ENABLED=OFF URL_ENDPOINT=http
s://cdn.proj.org USER_WRITABLE_DIRECTORY=/var/lib/postgresql/.local/share/proj DATABASE_PATH=/usr/share/proj/proj.db"
(compiled against PROJ 9.6.0) LIBXML="2.13.8" LIBJSON="0.18" LIBPROTOBUF="1.5.2" WAGYU="0.5.0 (Internal)" TOPOLOGY
(1 row)
gis_db=#
```

Рис. 3. Проверка установки PostGIS

Таким образом, совокупность вышеуказанных действий позволяет не только подтвердить успешный запуск контейнера, но и удостовериться в корректной работе программного стека, включая интерпретатор оболочки, СУБД PostgreSQL и пространственные функции расширения PostGIS.

4 Вывод

В данной работе был рассмотрен процесс проектирования и развертывания контейнеризированной среды базы данных PostgreSQL с интегрированным расширением PostGIS средствами Docker. Предложенный подход обеспечивает высокую степень переносимости среды, минимизирует временные затраты на настройку и позволяет унифицировать стек разработки и эксплуатации (Dev/Ops). Разработанная конфигурация может служить основой для построения отказоустойчивых геоинформационных систем в облачных архитектурах.

Библиографический список

1. Захарчёнок В.Ф., Бизюк А.Н. Контейнеризация и развертывание 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. Лавров Н.М., Селина Л.В., Платунов И.А. Сервис для картографирования корпусов университета // Интернаука. 2023. № 23-1 (293). С. 38-43.
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. Овсянникова Д.Ю. Проектирование геоинформационной системы и разработка модуля с использованием PostgreSQL, PostGIS и Qgis // Студенческий вестник. 2020. № 5-4 (103). С. 72-75.
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.