МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ   
ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

КАФЕДРА КОМПЬЮТЕРНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ  
ПРОЕКТИРОВАНИЯ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6  
  
По дисциплине Системы хранения и обработки данных

Тема: Развёртывание СУБД Postgres с использованием средств автоматизации развёртывания и управления приложениями

Выполнил работу студент группы мИИВТ-231 Ахлестин А.И. (подпись) Фамилия, инициалы

Принял \_\_\_\_\_\_\_\_\_\_\_\_Короленко В.В.  
 (подпись) Фамилия, инициалы

Защищена\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Воронеж 2023

Цель работы: изучить основы работы программного средства для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker на примере решения задачи развёртывания СУБД Postgres.

Задание на лабораторную работу:

1. Установка приложения для работы с Docker-контейнерами;
2. Установка программного средства для работы с СУБД DBeaver;
3. Скачивание docker-образа с СУБД postgres;
4. Создание Dockerfile;
5. Создание образа на основе Dockerfile;
6. Запуск контейнера для развёртывания СУБД Postgres;
7. Подключение к работающему контейнеру и запуск интерфейса psql;
8. Создание контейнера с томом (volume);
9. Создание контейнера с использованием файла docker-compose.yml.

Ход выполнения задания:

# 1 Установка приложения для работы с Docker-контейнерами

# Для начала перейдем на официальный сайт Docker и создадим аккаунт. Официальный сайт отображен на рисунке 1.

# 

# Рисунок 1 – Официальный сайт Docker.

# Далее нам необходимо зарегистрироваться на портале, как показано на рисунке 2.

# 

# Рисунок 2 – Регистрация.

# Следующим шагом станет скачивание Docker Desktop, выбираем версию для windows, как показано на рисунке 3.

# 

# Рисунок 3 – Скачивание программы.

# Далее установим программу. Для этого запустим инсталлятор (рисунок 4), который мы скачали на сайте, и проследуем инструкциям.

# 

# Рисунок 4 – Установка программы.

# 2 Установка программного средства для работы с СУБД DBeaver

# Для начала скачаем установщик для программы, как показано на рисунке 5.

# 

# Рисунок 5 – Выбор версии программы.

# Теперь нам доступен инсталлер (рисунок 6), запустим его и установим программу следуя инструкциям.

# 

# Рисунок 6 – Установщик программы.

# 3 Скачивание docker-образа с СУБД postgres

# Для скачивания docker-образа с СУБД postgres необходимо открыть командную строку и использовать команду, которая показана на рисунке 7.

# 

# Рисунок 7 - Команда для скачивания Docker-образа.

# 4 Создание Dockerfile

# Для следующего этапа создадим Dockerfile в папке репозитория из 4 лабораторной работы. Содержимое файла представлено на рисунке 8.

# 

# Рисунок 8 – Содержимое файла.

# Каждая строка означает следующее:

# FROM postgres:latest: Использует официальный образ postgres в качестве основы.

# ENV POSTGRES\_PASSWORD=dbpass: Устанавливает пароль для пользователя PostgreSQL.

# ENV POSTGRES\_USER=dbuser: Устанавливает имя пользователя PostgreSQL.

# ENV POSTGRES\_DB=dbname: Устанавливает имя базы данных PostgreSQL.

# COPY init\_scripts/init.sql /docker-entrypoint-initdb.d/init.sql: Копирует файл инициализации SQL в нужное место.

# Помимо Dockerfile, у нас есть init.sql, содержание которого представлено на рисунке 9.

# 

# Рисунок 9 – Содержание файла init.sql

# Разберем каждую строку файла init.sql:

# CREATE TABLE IF NOT EXISTS public.index\_mass: запрос создает таблицу с именем index\_mass в схеме public (если она еще не существует).

# Ключевое слово IF NOT EXISTS гарантирует, что таблица будет создана только в том случае, если она еще не существует.

# Структура таблицы определена тремя столбцами: user\_id типа BIGINT, weight типа BIGINT и height типа BIGINT.

# INSERT INTO public.index\_mass (user\_id, weight, height) VALUES: запрос вставляет данные в таблицу index\_mass.

# По одной строке на каждый запрос VALUES, вставляются значения для каждого столбца (user\_id, weight и height) для каждой записи.

# В данном случае вставляются три записи с данными о пользователях.

# 5 Создание образа на основе Dockerfile

# Теперь мы можем создать образ на основе Dockerfile. Для этого перейдем в папку с файлом, откроем там терминал и запустим команду создания образа, как показано на рисунке 10. Это создаст Docker-образ с именем d\_img и тегом latest.

# 

# Рисунок 10 – Создание образа.

# 6 Запуск контейнера для развёртывания СУБД Postgres

# Запустим контейнер с помощью команды, отображенной на рисунке 11.

# Эта команда запускает контейнер в фоновом режиме, пробрасывает порт 5432, и называет контейнер d\_cont.

# 

# Рисунок 11 – Запуск команды.

# 7 Подключение к работающему контейнеру и запуск интерфейса psql

# Откроем DBeaver и создадим новое подключение к PostgreSQL используя параметры из Dockerfile, как показано на рисунках 12 и 13.

# 

# Рисунок 12 – Выбор базы данных.

# 

# Рисунок 13 – Заполнение настроек и тест подключения к бд.

# Теперь запустим в терминале команду, которая показана на рисунке 14. Она позволяет нам работать с базой данных из консоли. Введем SQL-запрос в интерфейсе psql, например, \dt для просмотра списка таблиц.

# 

# Рисунок 14 – Работа с бд через интерфейс psql.

# docker exec: Эта часть команды указывает Docker'у на выполнение команды внутри контейнера.

# -it: Эти флаги предоставляют интерактивный терминал (или tty) внутри контейнера. Флаг -i означает интерактивный режим, а -t указывает использовать псевдо-терминал.

# d\_cont: Это имя контейнера, в котором вы хотите выполнить команду.

# psql: Это клиент командной строки PostgreSQL, который вы будете использовать для взаимодействия с базой данных.

# -d dbname: Этот флаг указывает имя базы данных, с которой вы хотите взаимодействовать.

# -U dbuser: Этот флаг указывает имя пользователя PostgreSQL, под которым вы хотите войти.

# 8 Создание контейнера с томом (volume)

# Добавим в Dockerfile новую строку, как показано на рисунке 15. Теперь, при создании контейнера, Docker будет использовать том /d\_data, где будут храниться данные.

# 

# Рисунок 15 – Добавление строки в Dockerfile.

# Теперь создадим контейнер с использованием тома /d\_data для хранения данных, но перед этим удалим старый, как показано на рисунке 16.

# 

# Рисунок 16 – Создание нового контейнера.

# Разберем каждую часть команды docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data:

# 1. docker run: Эта команда используется для создания и запуска нового контейнера из образа.

# 2. -d: Этот флаг указывает Docker'у запустить контейнер в фоновом режиме (detached mode).

# 3. -p 5432:5432: Этот флаг пробрасывает порты между хостовой машиной и контейнером. В данном случае, порт 5432 хоста ассоциируется с портом 5432 контейнера, что позволяет подключаться к PostgreSQL извне через порт 5432 хоста.

# 4. --name d\_cont: Этот флаг устанавливает имя для контейнера. В данном случае, контейнер будет назван d\_cont.

# 5. d\_img:latest: Это имя образа и его тег. В данном случае, d\_img - имя образа, и latest - тег, указывающий на последнюю версию образа.

# 6. -v /d\_data:/var/lib/postgresql/data: Этот флаг используется для создания и монтирования тома (volume). Он связывает локальную директорию /d\_data с директорией /var/lib/postgresql/data внутри контейнера. Это позволяет сохранять данные PostgreSQL вне контейнера, что сохраняет состояние базы данных между перезапусками контейнера.

# Таким образом, при выполнении этой команды создается и запускается контейнер PostgreSQL. Контейнер будет работать в фоновом режиме, пробрасывать порт 5432, иметь имя d\_cont, использовать образ d\_img:latest и сохранять данные в директории /d\_data на хосте.

# 9 Создание контейнера с использованием файла docker-compose.yml

# Для начала создадим файл docker-compose.yml. Содержание файла представлено на рисунке 17.

# 

# Рисунок 17 – Содержание файла.

# Разберем содержание файла docker-compose.yml.

# version: "3.9": Это версия синтаксиса Docker Compose. В данном случае, используется версия 3.9.

# services: Этот раздел определяет сервисы (контейнеры) и их конфигурацию.

# dbpost: Название сервиса (контейнера), в данном случае, это PostgreSQL.

# image: postgres:latest: Указывает Docker использовать официальный образ PostgreSQL с тегом latest.

# environment: Задает переменные окружения для контейнера, такие как имя базы данных (POSTGRES\_DB), имя пользователя (POSTGRES\_USER), и пароль (POSTGRES\_PASSWORD).

# volumes: Монтирует локальный файл ./init\_scripts/init.sql внутрь контейнера по пути /docker-entrypoint-initdb.d/init.sql. Это позволяет выполнить SQL-скрипт при инициализации базы данных.

# ports: Пробрасывает порт 5432, что позволяет подключаться к PostgreSQL извне через порт 5432 хоста.

# Процесс работы:

# При запуске команды docker-compose up, Docker Compose создает и запускает контейнер на основе указанных параметров в файле docker-compose.yml.

# Результат:

# Создается контейнер PostgreSQL с именем dbpost.

# Устанавливаются переменные окружения для базы данных (dbcompose), пользователя (usrcompose), и пароля (passcompose).

# Монтируется SQL-скрипт (init.sql), который будет выполнен при инициализации базы данных.

# Пробрасывается порт 5432.

# Для запуска контейнера с использованием Docker Compose выполним команду docker-compose up в каталоге, содержащем файл docker-compose.yml. Результат выполнения команды представлен на рисунке 18.

# 

# Рисунок 18 – Выполнение команды.

Вывод: в процессе выполнения работы были освоены основы работы с Docker, а также инструментами для взаимодействия с контейнерами.

Контрольные вопросы:

1. Что такое Docker?

Docker - это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры обеспечивают легкость в развертывании и масштабировании приложений, а Docker предоставляет инструменты для упаковки и запуска этих контейнеров.

1. Зачем нужен Docker?

Docker облегчает процессы разработки, развертывания и масштабирования приложений. Он позволяет упаковывать приложение и его зависимости в контейнер, который может быть запущен на любой совместимой с Docker системе, обеспечивая консистентность окружения.

1. Что такое docker-образ?

Docker-образ - это портативный и легковесный исполняемый пакет, который включает в себя приложение и все его зависимости, включая файловую систему, настройки и код. Образ используется для создания контейнера.

1. Что такое docker-контейнер?

Docker-контейнер - это экземпляр Docker-образа, который запущен и выполняется в изолированном окружении. Контейнер обеспечивает консистентность и переносимость приложения между различными средами.

1. Что такое volume и зачем он нужен?

Volume в Docker представляет собой механизм для сохранения и обмена данными между контейнером и хост-машиной, а также между разными контейнерами. Volume обеспечивает постоянное хранилище данных и позволяет сохранять данные даже после остановки и удаления контейнера.

1. Что такое docker-compose?

Docker Compose - это инструмент для определения и запуска многоконтейнерных приложений с использованием файла конфигурации в формате YAML. Он упрощает управление сложными приложениями, включающими множество сервисов и контейнеров.

1. В чем разница между dockerfile и docker-compose?

Dockerfile - это текстовый файл, который содержит инструкции для построения Docker-образа. Docker-compose - это инструмент для определения и запуска многоконтейнерных приложений с использованием файла конфигурации.

1. Какая команда позволяет отправлять различные задания в запущенный докер-контейнер?

Команда docker exec позволяет отправлять различные задания в запущенный докер-контейнер.

1. С помощью какого инструмента можно сохранить важные данные после аварийного отключения контейнера?

Для сохранения важных данных после аварийного отключения контейнера используется механизм volume, который монтирует определенные директории из контейнера на хост-машину, обеспечивая сохранность данных.