МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
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

ОТЧЁТ 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: Никулин В.С.

подпись, дата

Принял: Короленко В.В.

подпись, дата

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.

**Задачи работы:** необходимо выполнить все пункты учебной задачи:

1. Изучить сайт https://www.docker.com, документацию по docker (https://docs.docker.com), зарегистрироваться на Docker Hub. Изучить основные команды docker для консоли.
2. Скачать и установить Docker Desktop. При работе в команде необходимо выполнять работу на различных операционных системах. Приоритетный порядок выбора операционных систем: Astra Linux, Windows 10, CentOS, любая другая система, кроме указанных ранее, включая MacOS. Как минимум в одном варианте необходимо установить Docker и далее работать на виртуальной машине (операционная система – по вашему выбору). Количество вариантов (систем, на которые устанавливается Docker) может не превышать количество участников команды (не запрещается делать больше вариантов, чем участников команды).
3. Скачать и установить средство для работы с СУБД DBeaver. Разрешается использовать другое подобное средство, но необходимо обосновать свой выбор.
4. Скачать docker-образ с СУБД postgres с помощью команды для консоли.
5. Создать Dockerfile со следующим содержанием:

FROM postgres:latest

ENV POSTGRES\_PASSWORD=dbpass

ENV POSTGRES\_USER=dbuser

ENV POSTGRES\_DB=dbname

COPY init\_scripts/init.sql /docker-entrypoint-initdb.d/init.sql

Описать каждую строку данного файла (что означает и какую задачу решает).

Содержание файла init.sql:

CREATE TABLE IF NOT EXISTS public.index\_mass (

user\_id BIGINT,

weight BIGINT,

height BIGINT

);

INSERT INTO public.index\_mass (user\_id, weight, height) VALUES

(1, 75, 175),

(2, 60, 182),

(3, 93, 181);

Описать каждую строку данного файла.

1. Создать образ с помощью команды «docker build -t d\_img:latest .»:
   1. Команду надо запускать из каталога, в котором находится Dockerfile;
   2. В результате запустится Dockerfile и будет создан docker-образ «d\_img»;
2. Чтобы воспользоваться созданной СУБД необходимо запустить контейнер с именем «d\_cont» на основе образа «d\_img:latest» с помощью команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest». Описать данную команду (все составляющие и что происходит при запуске).
3. Проверить через DBeaver развёрнутую СУБД. Подключить базу данных с параметрами, указанными в Dockerfile командами ENV: (POSTGRES\_PASSWORD=dbpass (пароль), POSTGRES\_USER=dbuser (имя пользователя), POSTGRES\_DB=dbname [название базы]), порт – 5432.
4. Написать специальную команду docker, которая позволит подключаться к работающему контейнеру, запускать интерфейс psql и вносить новые данные «на лету»: «docker exec -it d\_cont psql -d -U usr dbn». Теперь есть возможность работать с базой данных из консоли. Необходимо описать данную команду (все составляющие и что происходит при запуске). Вывести в консоль список имеющихся баз данных.
5. Чтобы данные, измененные в процессе работы контейнера, были доступны после удаления контейнера (во вновь созданном контейнере) необходимо создать том (volume).
   1. Для добавления в Dockerfile инструкции для вызова создания контейнера с volume необходимо в Dockerfile добавить строку: «VOLUME /d\_data:/var/lib/postgresql/data».
   2. При этом локально будет создан том (volume) в папке «/d\_data», в котором, в свою очередь, будут сохраняться изменения, аналогичные тем, которые происходят в контейнере в папке «/var/lib/postgresql/data».
6. Удалить созданный контейнер. Создать контейнер с помощью команды: «docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data». Необходимо описать данную команду (все составляющие и что происходит при запуске).
7. Создать текстовый файл cmd.txt, в котором описать все команды, которые необходимо использовать для развертывания базы данных Postgres с помощью Dockerfile (создание образа, запуск контейнера, создание тома). Также в этом файле необходимо указать команду для создания docker-контейнера с томом (volume) на основе docker-образа и команду для запуска контейнера с интерфейсом psql для внесения новых данных в БД.
8. Создать контейнер с использованием файла «docker-compose.yml».  
   Содержание файла docker-compose.yml:

version: "3.9"

services:

 dbpost:

  image: postgres:latest

  environment:

   POSTGRES\_DB: dbcompose

   POSTGRES\_USER: usrcompose

   POSTGRES\_PASSWORD: passcompose

  volumes:

   - ./init\_scripts/init.sql:/docker-entrypoint-initdb.d/init.sql

  ports:

   - 5432:5432

Описать каждую строку файла (что означает, какую задачу решает), процесс работы данного файла и результат.

1. При работе в команде необходимо выполнять работу на различных операционных системах. Приоритетный порядок выбора операционных систем: Astra Linux, Windows 10, CentOS, любая другая система, кроме указанных ранее.
2. Подготовить отчёт о проделанной работе. Все операции по работе с docker необходимо контролировать через приложение Docker Desktop и делать соответствующие скрины.
3. Отчётные материалы загрузить в репозиторий Git и отправить ссылку на ваш репозиторий на платформе github на почту преподавателю. Репозиторий должен быть публичным.

**Ход работы:**

Лабораторная работа выполнялась на виртуальной машине в операционной  
системе Fedora Linux. Для начала, необходимо было выполнить установку Docker Desktop. Для этого, был выполнен вход на официальный сайт программы, и изучены инструкции по установке (рисунок 1). Далее, была выполнена установка Docker Desktop (рисунок 2, рисунок 3), в соответствии с инструкцией, а также, была выполнена установка средства для работы с СУБД – DBeaver (рисунок 3).

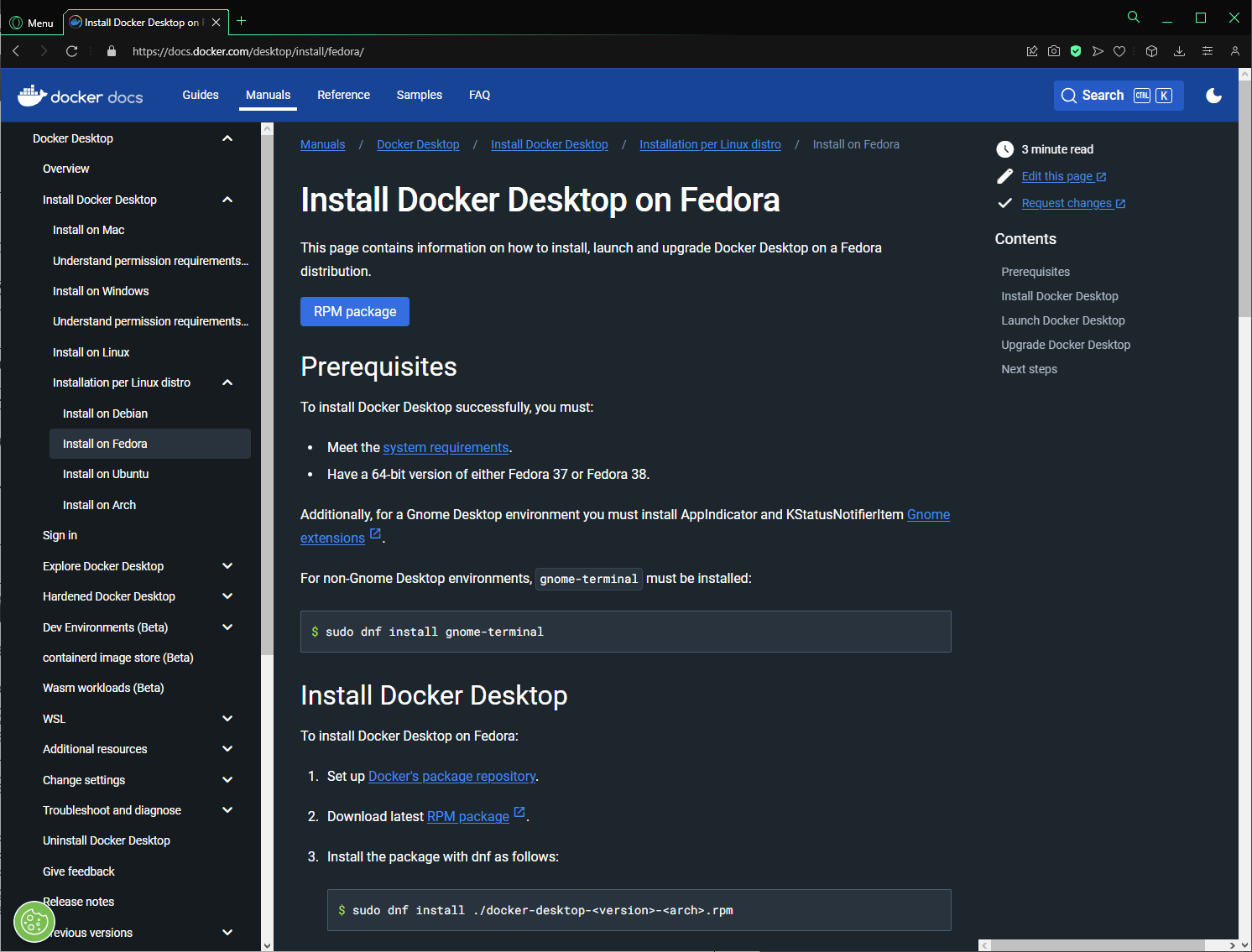


Рисунок - Инструкции по установке Docker Desktop для ОС Fedora Linux

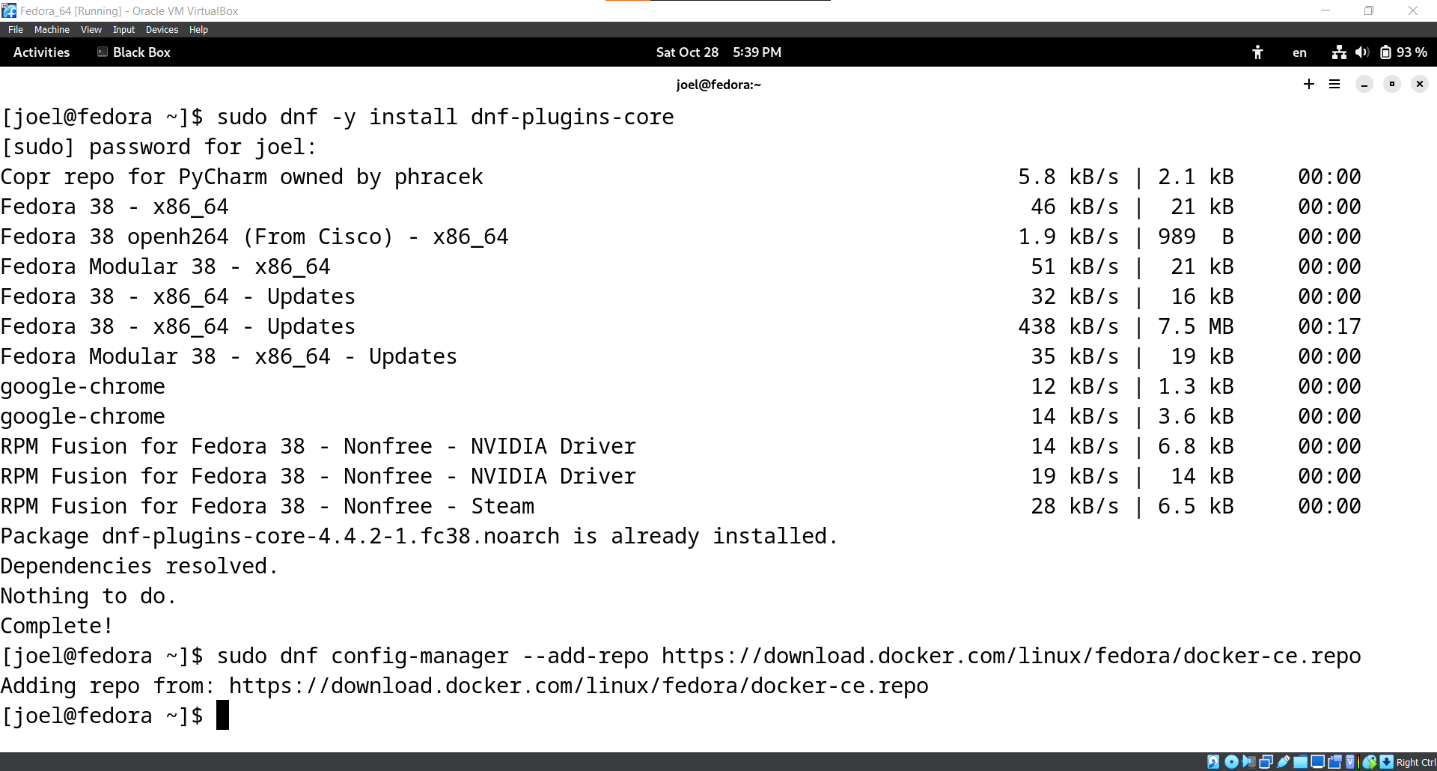


Рисунок - Добавление репозитория программ для Docker

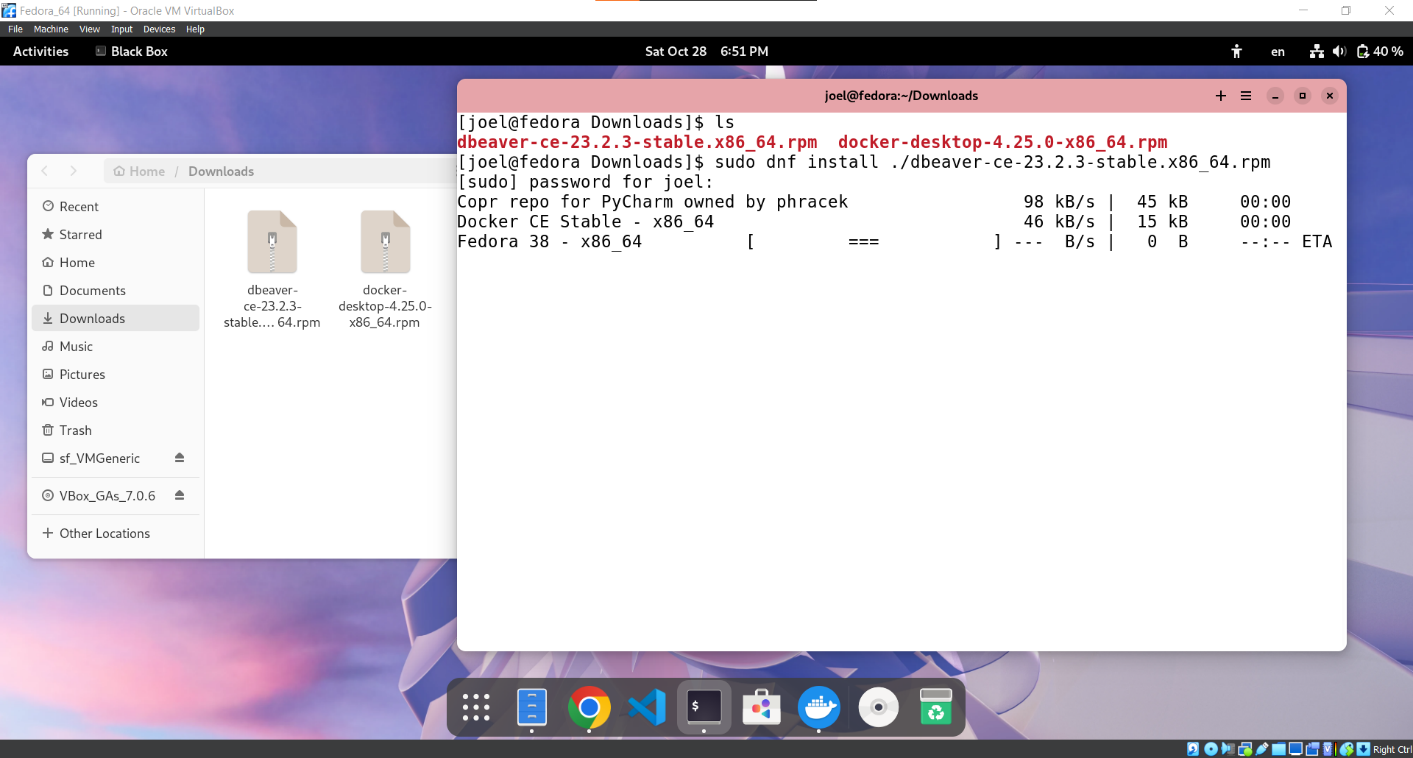


Рисунок - Скачанные RPM-пакеты для Docker Desktop и DBeaver,  
а также их установка через терминал

По завершению установки, программы были запущены, чтобы проверить, что всё в порядке (рисунок 4, рисунок 5).

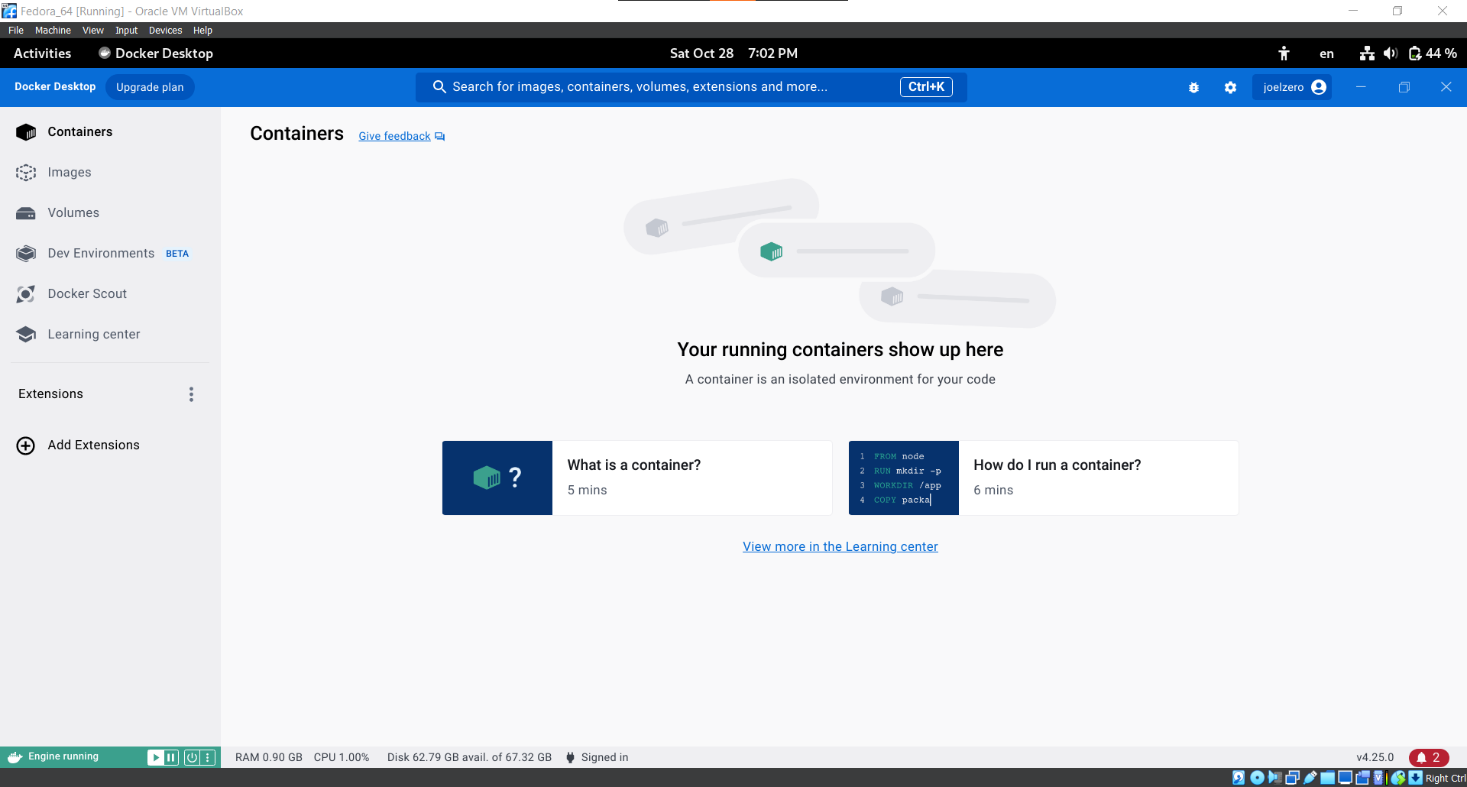


Рисунок - Docker Desktop успешно установлен

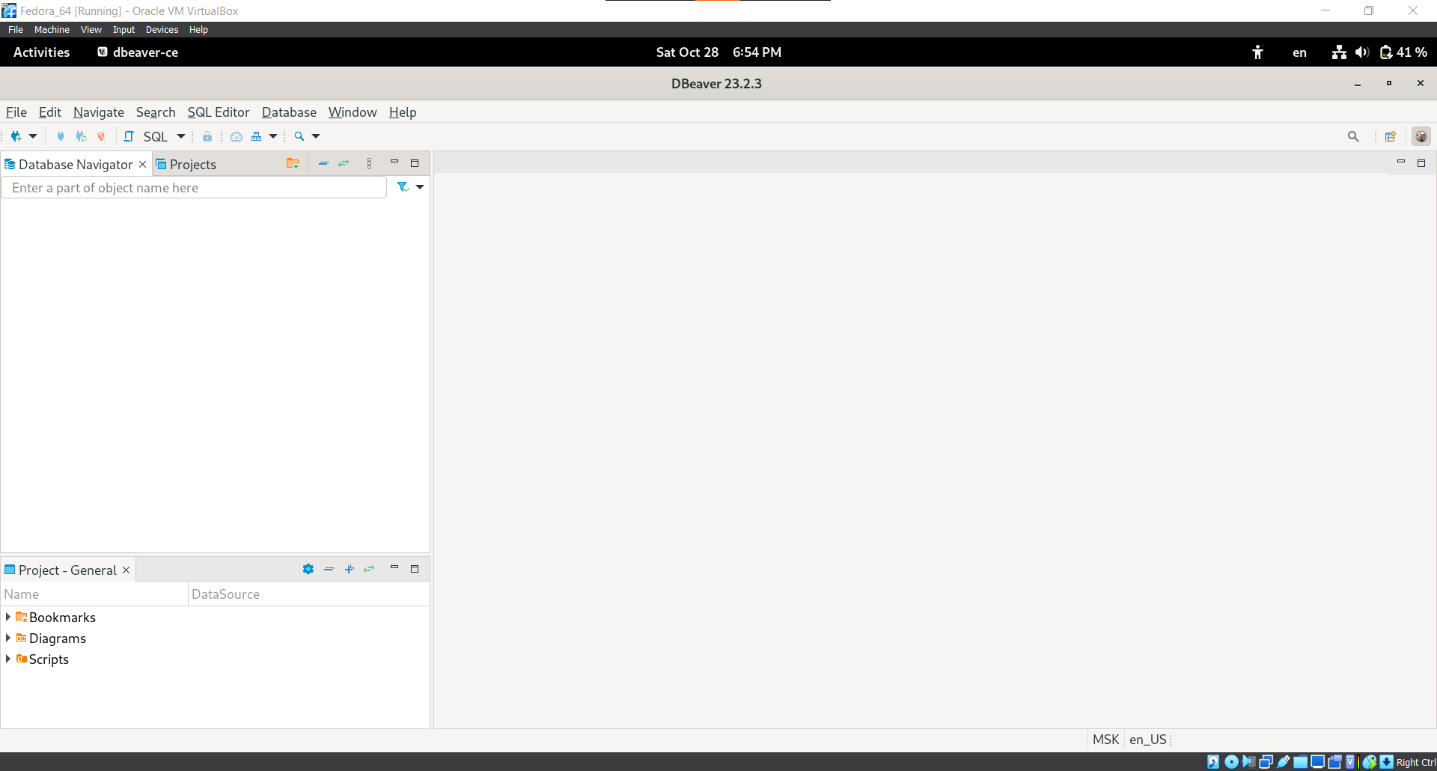


Рисунок - DBeaver успешно установлен

Далее, с помощью команды «docker pull postgres» был скачан Docker-образ СУБД PostgreSQL. Далее, был создан Dockerfile с инструкциями по созданию Docker-контейнера (рисунок 6).



Рисунок – Dockerfile с инструкциями по созданию Docker-образа

Первая строка инструкций (рисунок 6) описывает базовый образ, который будет использоваться для Docker-контейнера. Следующие три строки – это, соответственно, команды для создания переменных среды пароля, имя пользователя и названия будущей базы данных. Последняя строка – это команда копирования скрипта инициализации (init.sql) из локальной директории «init\_scripts», внутрь контейнера в специальное место, где PostgreSQL автоматически выполняет скрипты при первом запуске  
контейнера.

Содержимое файта «init.sql» изображено на рисунке 7. Файл содержит две команды. Первая команда «CREATE TABLE IF NOT EXISTS public.index\_mass user\_id BIGINT, weight BIGINT, height BIGINT);» создаст таблицу с именем index\_mass в схеме public, если такая таблица ещё не существует. Данная таблица будет содержать три столбца user\_id, weight и height типа BIGINT. Вторая команда «INSERT INTO public.index\_mass (user\_id, weight, height) VALUES (1, 75, 175), (2, 60, 182),  
(3, 93, 181);» вставит три строки данных в, созданную ранее первой командой,  
SQL-таблицу.



Рисунок - Содержимое скрипта «init.sql»

Далее, с помощью команды «docker build -t d\_img:latest .» был создан Docker-образ «d\_img» на основе данного Dockerfile (рисунок 8, рисунок 9). Опция «-t» используется для указания тега (имени) образа, а через двоеточие указывается версия этого образа, либо «latest» по-умолчанию. Точка в конце команды указывает на то, что Dockerfile находится в данной директории.

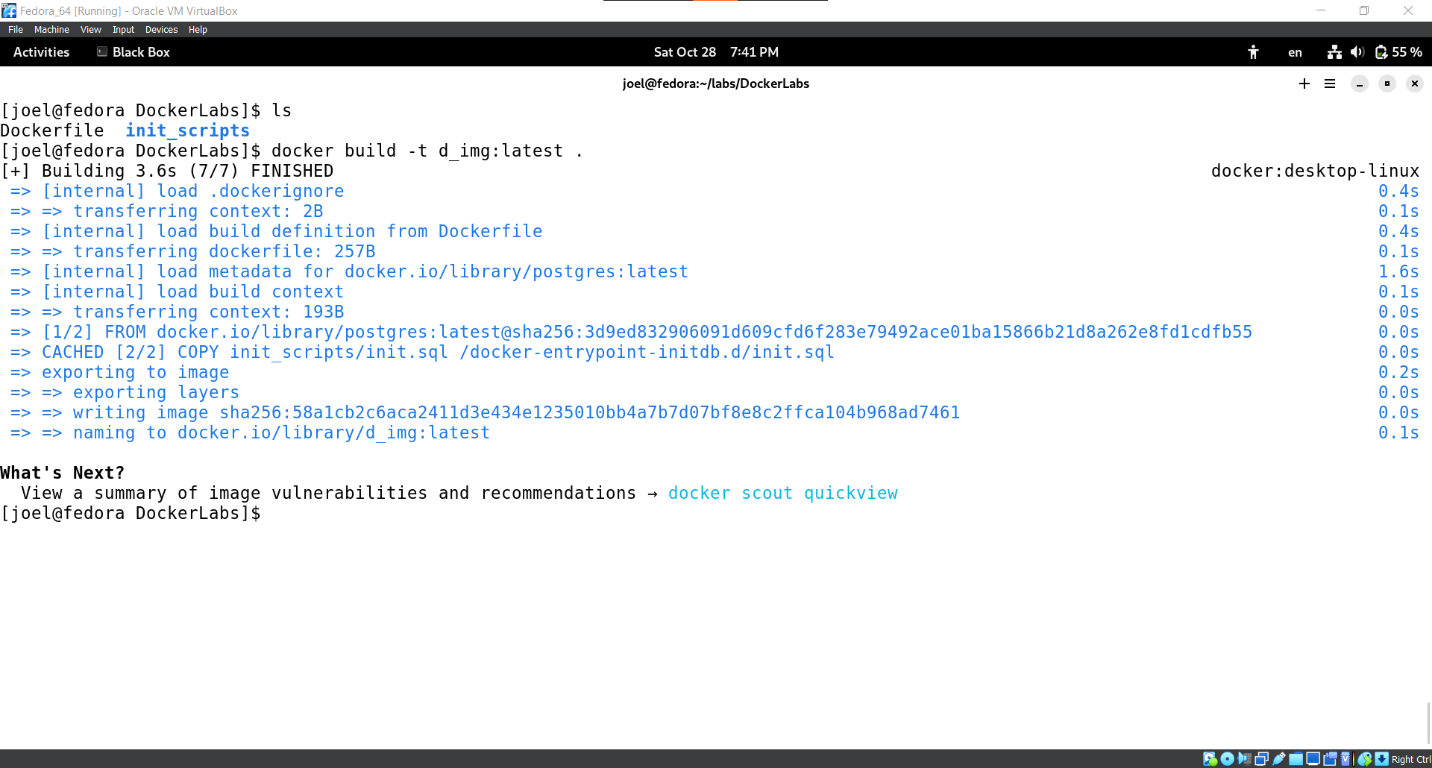


Рисунок - Создание образа «d\_img» в терминале

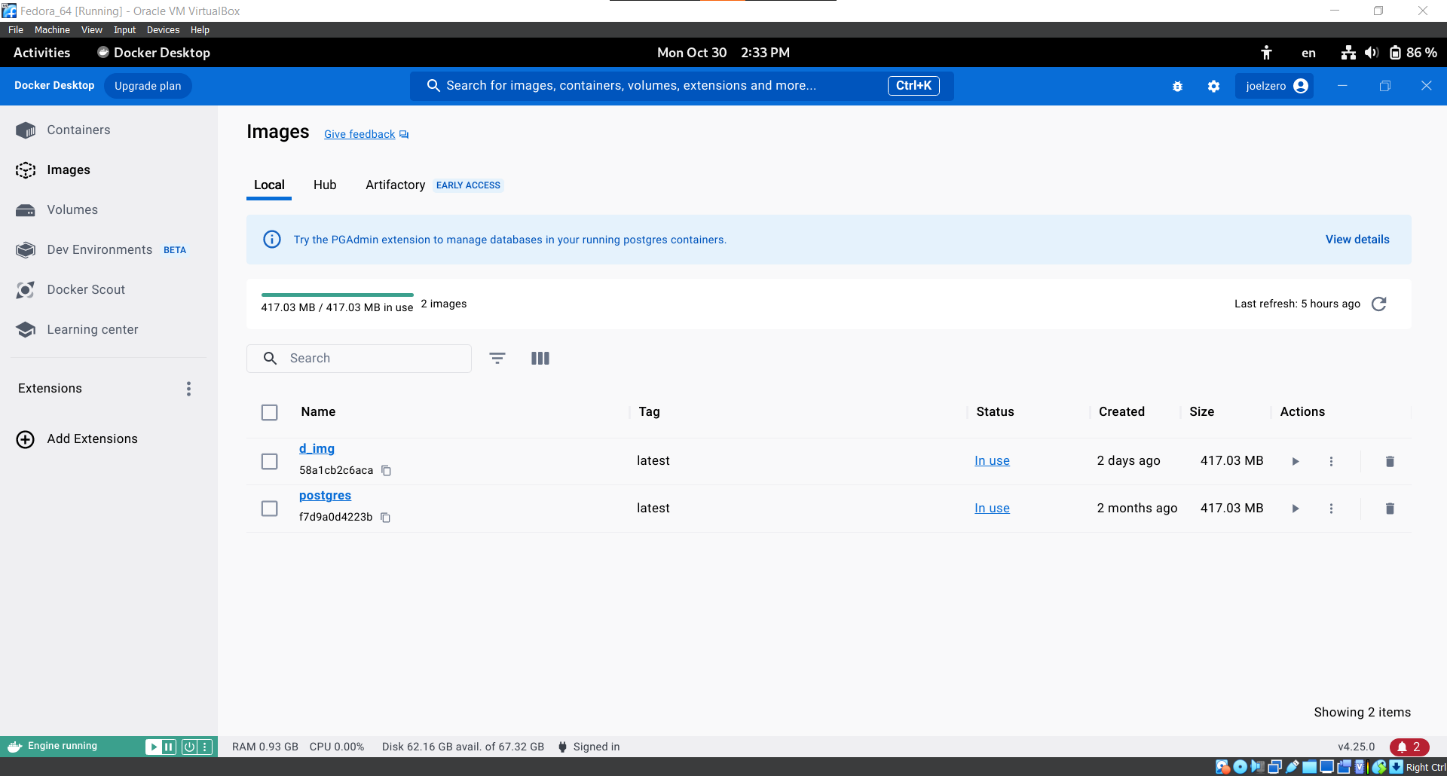


Рисунок - Образ «d\_img» отображается в Docker Desktop

Далее, был запущен контейнер «d\_cont» на основе образа «d\_img», с помощью команды «docker run -d -p 5432:5432 --name d\_cont d\_img:latest» (рисунок 10,  
рисунок 11). Опция «-d» используется для запуска контейнера в фоновом режиме (detached mode), т.е. пользователь снова получит доступ к командной строке. Опция «-p» используется для связки портов хоста и контейнера. Опция «--name» используется для указания имени контейнера (d\_cont). Аргумент «d\_img:latest» - тег образа (его имя), на основе которого будет создан контейнер.



Рисунок - Запуск контейнера «d\_cont» на основе образа «d\_img»

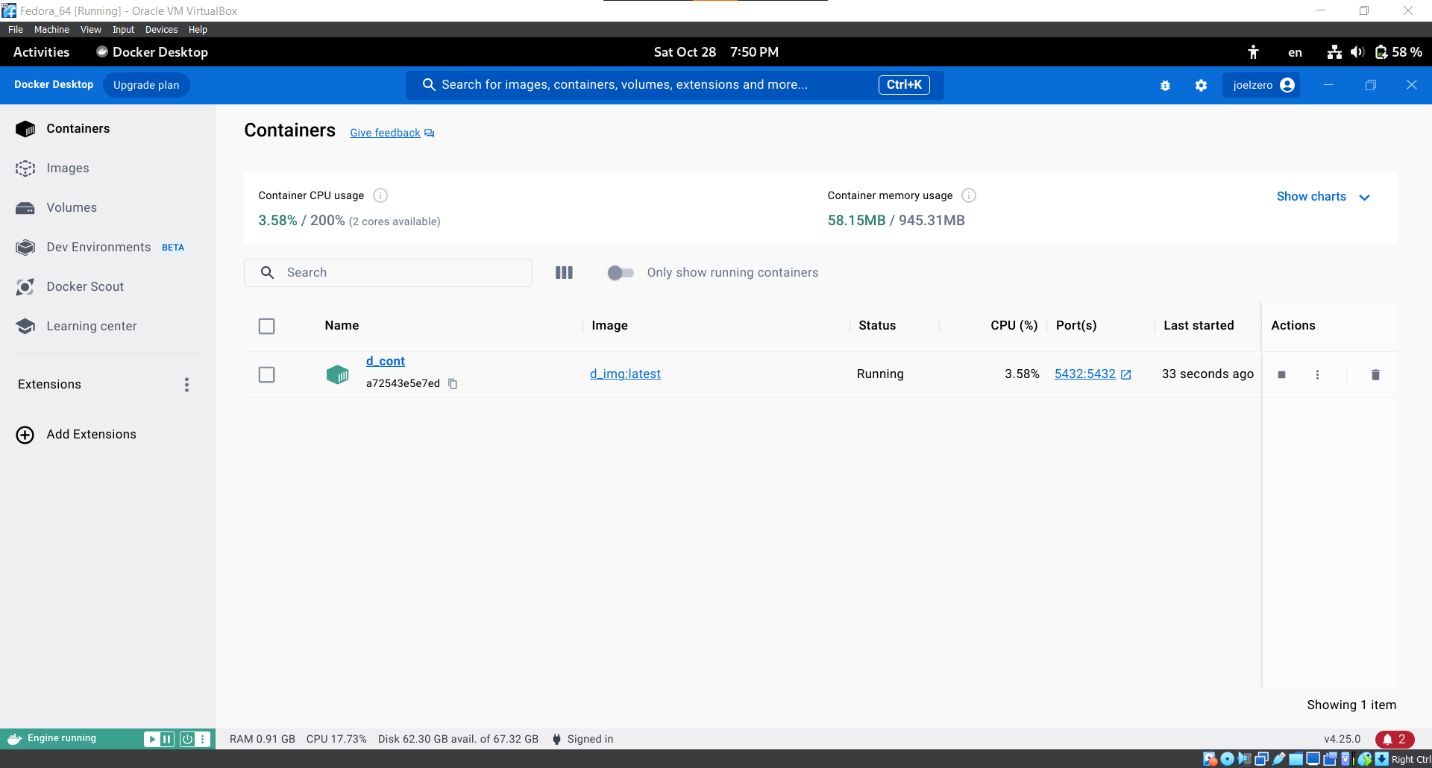


Рисунок – Контейнер «d\_cont» успешно создан и работает

Далее, было выполнено подключение к базе данных из контейнера «d\_cont», с помощью DBeaver (рисунок 12, рисунок 13).

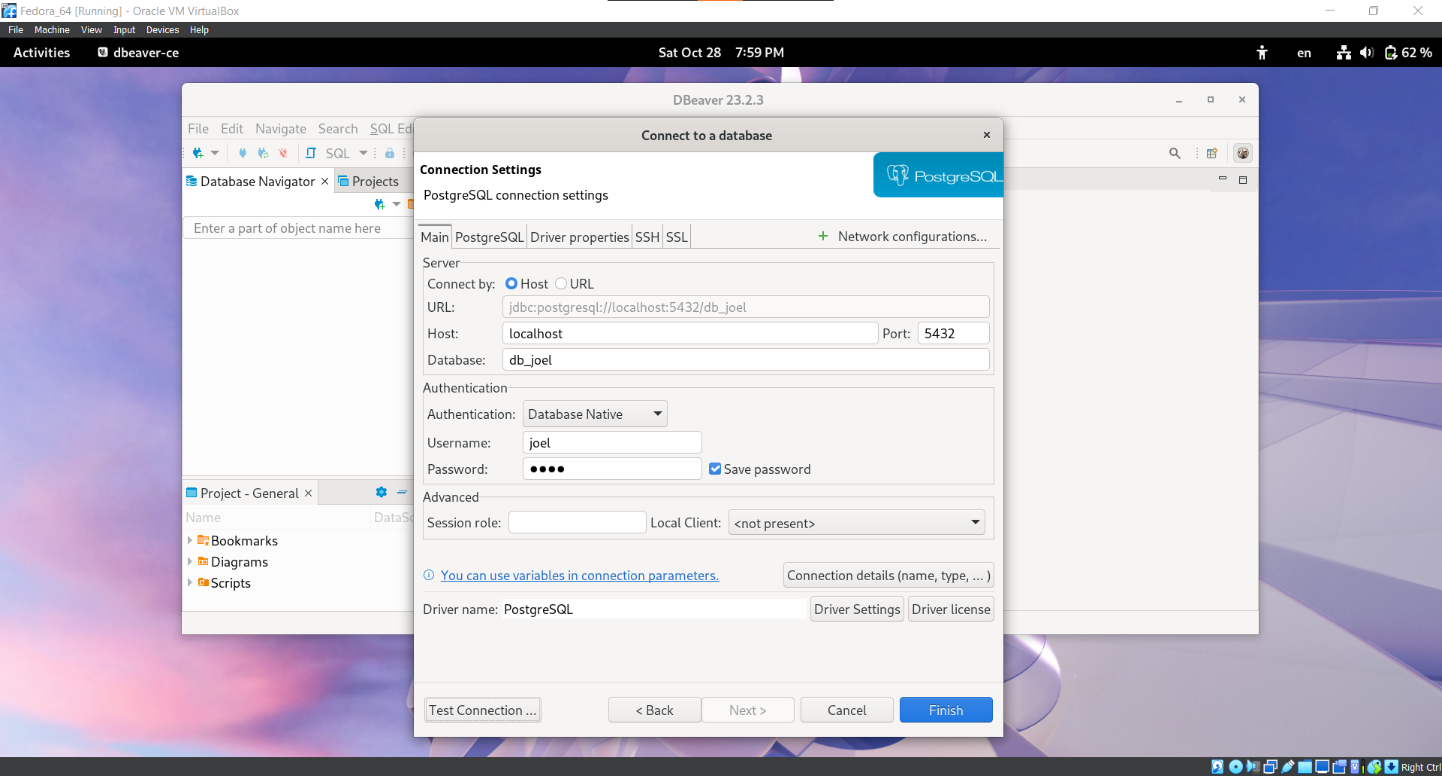


Рисунок - Подключение к базе данных контейнера «d\_cont» через DBeaver

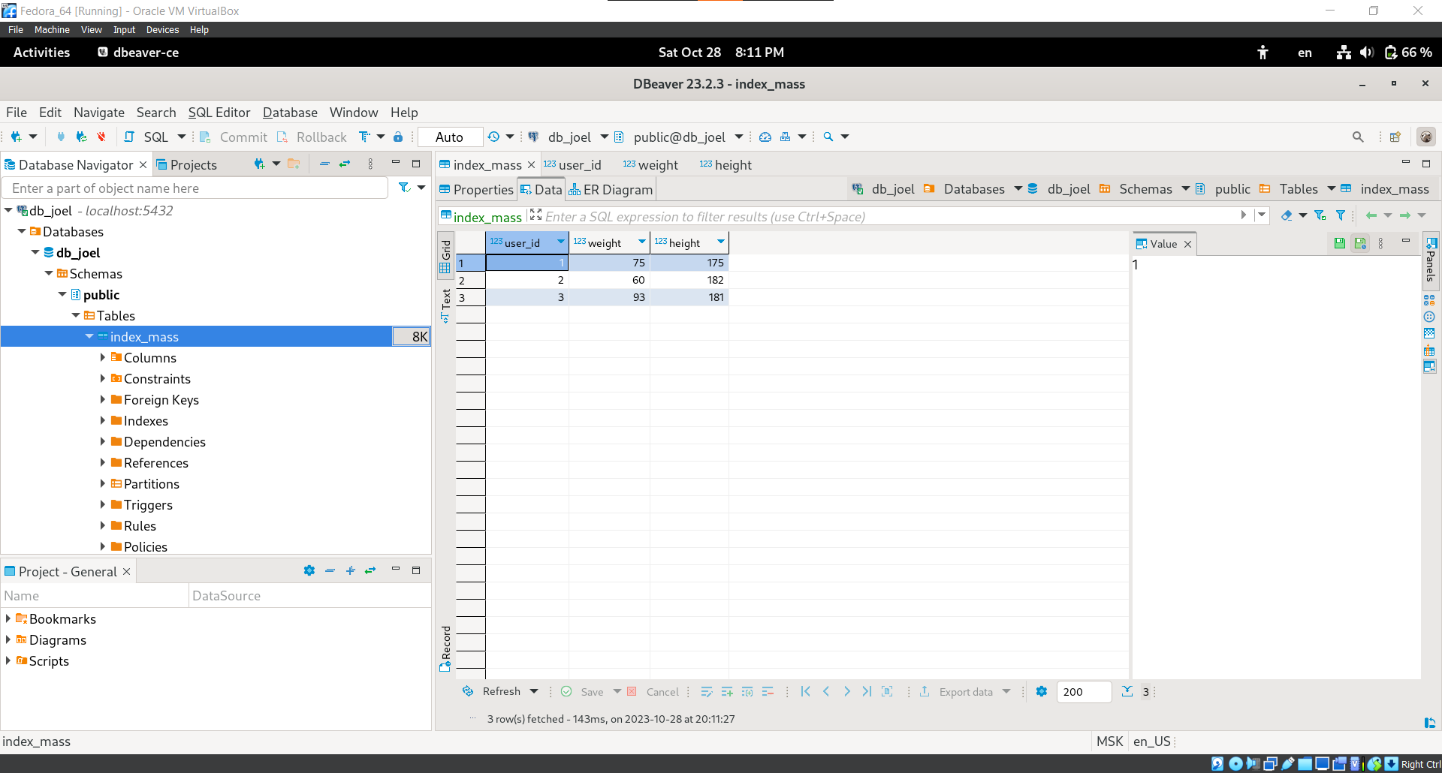


Рисунок - Успешное подключение к БД «db\_joel»  
и просмотр таблицы «index\_mass»

Далее, с помощью команды «docker exec -it d\_cont psql -d db\_joel -U joel», было выполнено подключение к интерфейсу «psql» контейнера «d\_cont». После этого, появится возможность взаимодействовать с СУБД, например, просмотреть список всех имеющихся баз данных (рисунок 14).

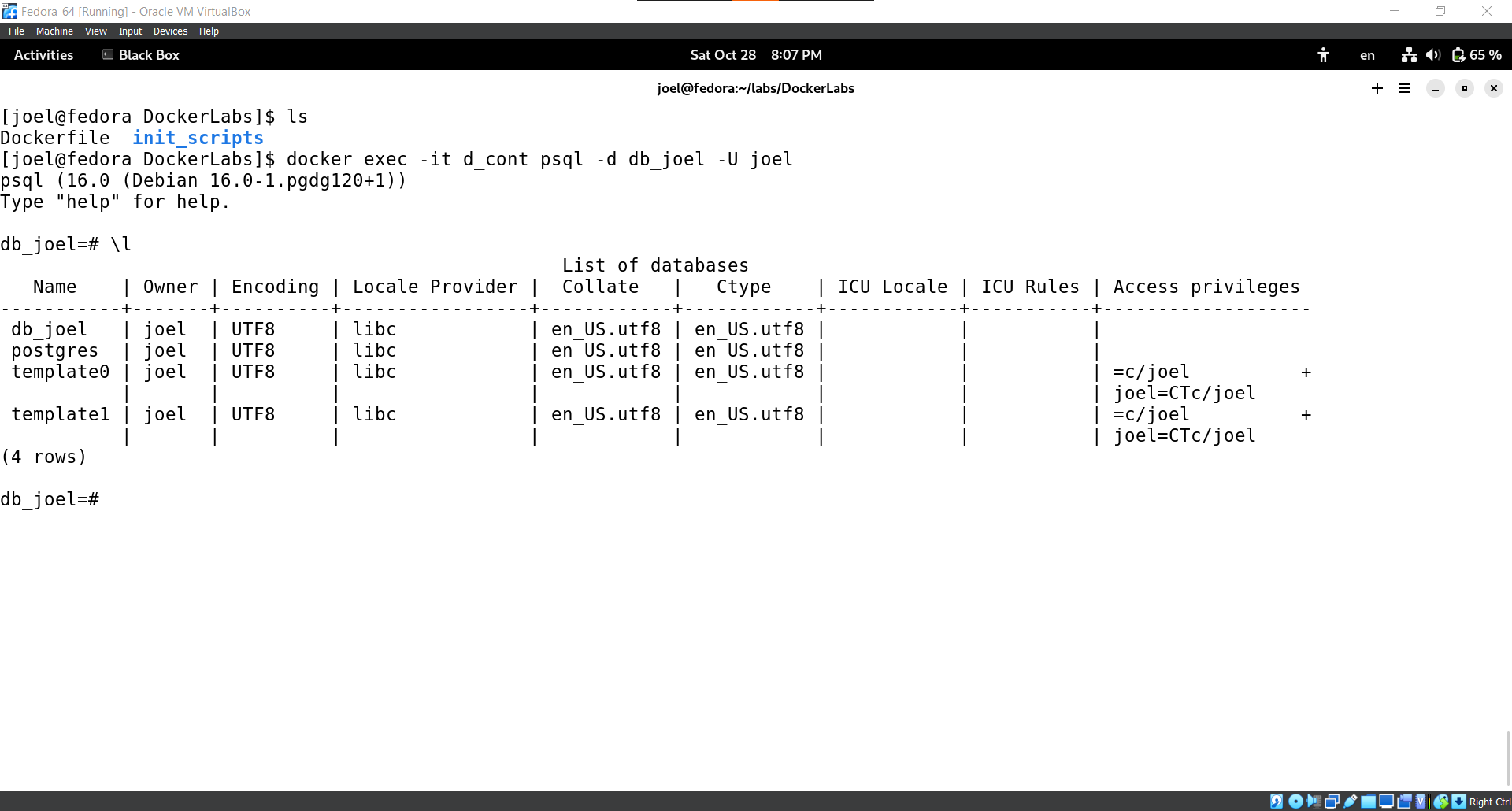


Рисунок - Просмотр всех имеющихся баз данных через интерфейс «psql»

Команда «docker exec -it d\_cont psql -d db\_joel -U joel» осуществляет подключение к интерфейсу «psql» в работающем контейнере «d\_cont». Опция «-it» обеспечивают интерактивный режим и привязывают терминал к стандартному вводу/выводу контейнера, что позволяет вам взаимодействовать с командной оболочкой внутри контейнера. Аргумент «d\_cont» – это имя контейнера, к которому будет выполнено подключение. Опция «psql -d db\_joel -U joel» указывает имя базы данных (db\_joel) и имя пользователя (joel).

Далее, в Dockerfile были добавлены изменения, которые позволят создать том для хранения информации за пределами контейнера (рисунок 15).



Рисунок - В Dockerfile внесена информация для создания тома

Далее, было выполнено удаление имеющегося контейнера «d\_cont» и было выполнено создание нового контейнера «d\_cont» с монтированием, созданного ранее, тома для хранения данных (рисунок 16).



Рисунок - Удаление и создание нового контейнера «d\_cont»  
с монтированием тома

Команда «docker run -d -p 5432:5432 --name d\_cont d\_img:latest -v /d\_data:/var/lib/postgresql/data» уже была описана ранее. Она отличается лишь тем, что в данной команде опцией «-v» выполняется монтирование тома с хоста внутрь контейнера, для возможности сохранять данных за пределы контейнера. Это позволит, в случае удаления контейнера сохранить информацию.

Далее, был создан файл «cmd.txt», где были описаны все команды, которые необходимо использовать для развертывания базы данных PostgreSQL, с помощью Dockerfile. Также там была описана команда для создания контейнера с томом и команда для запуска контейнера с интерфейсом «psql» (рисунок 17).

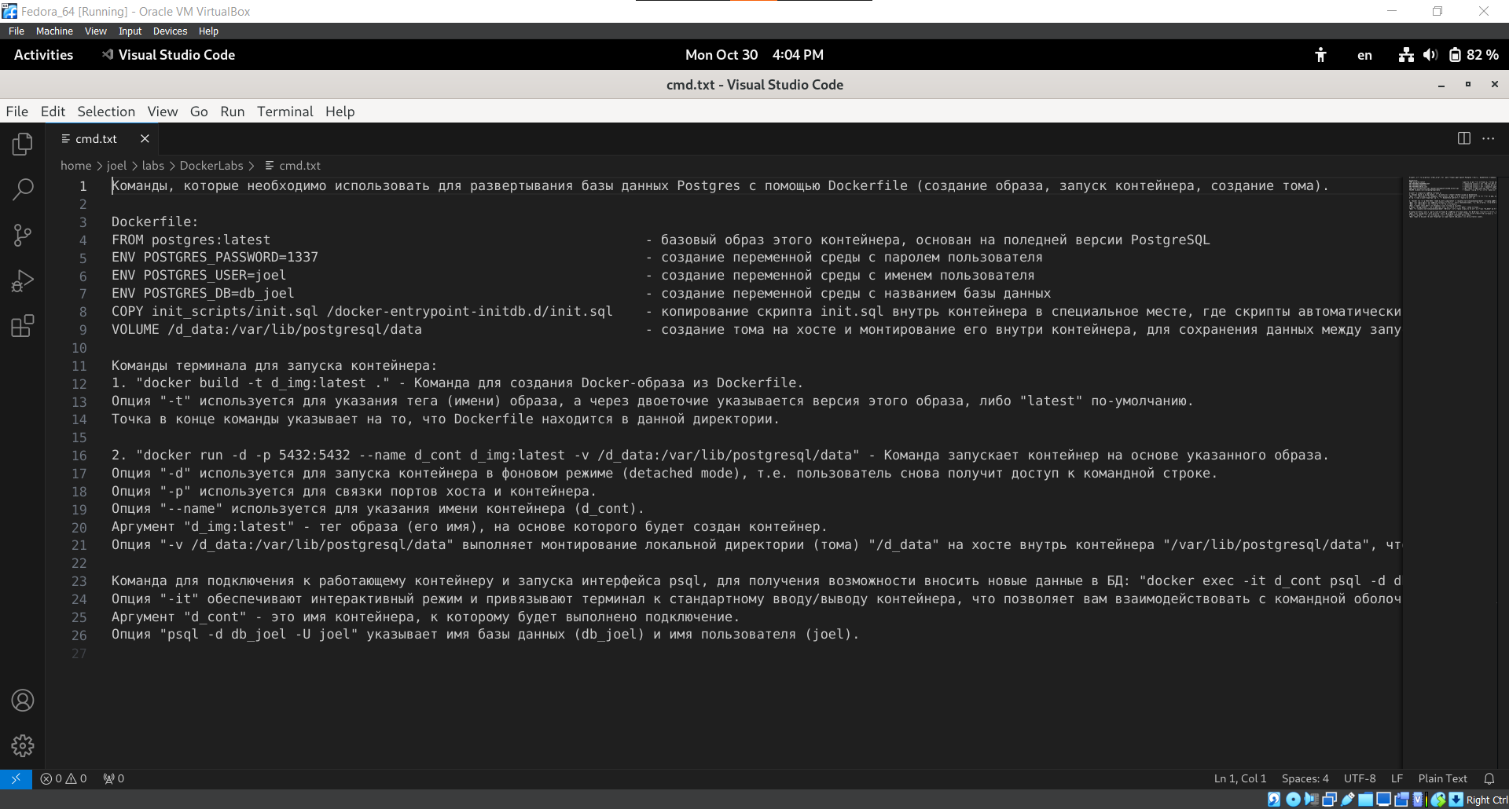


Рисунок - Файл «cmd.txt» с описанием команд

Далее, было выполнено создание контейнера с использованием файла  
«docker-compose.yml» (рисунок 18, рисунок 19, рисунок 20, рисунок 21).

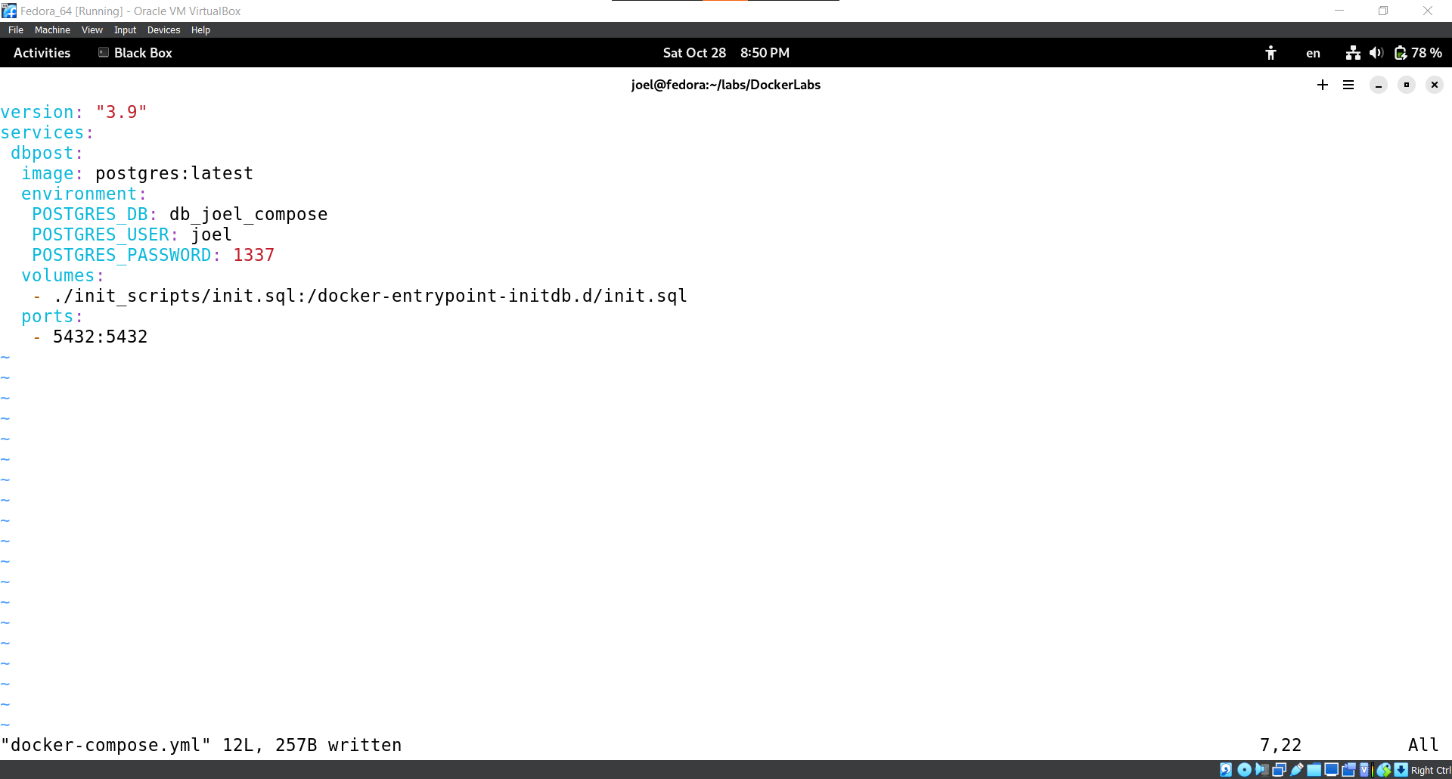


Рисунок - Содержимое файла «docker-compose.yml»

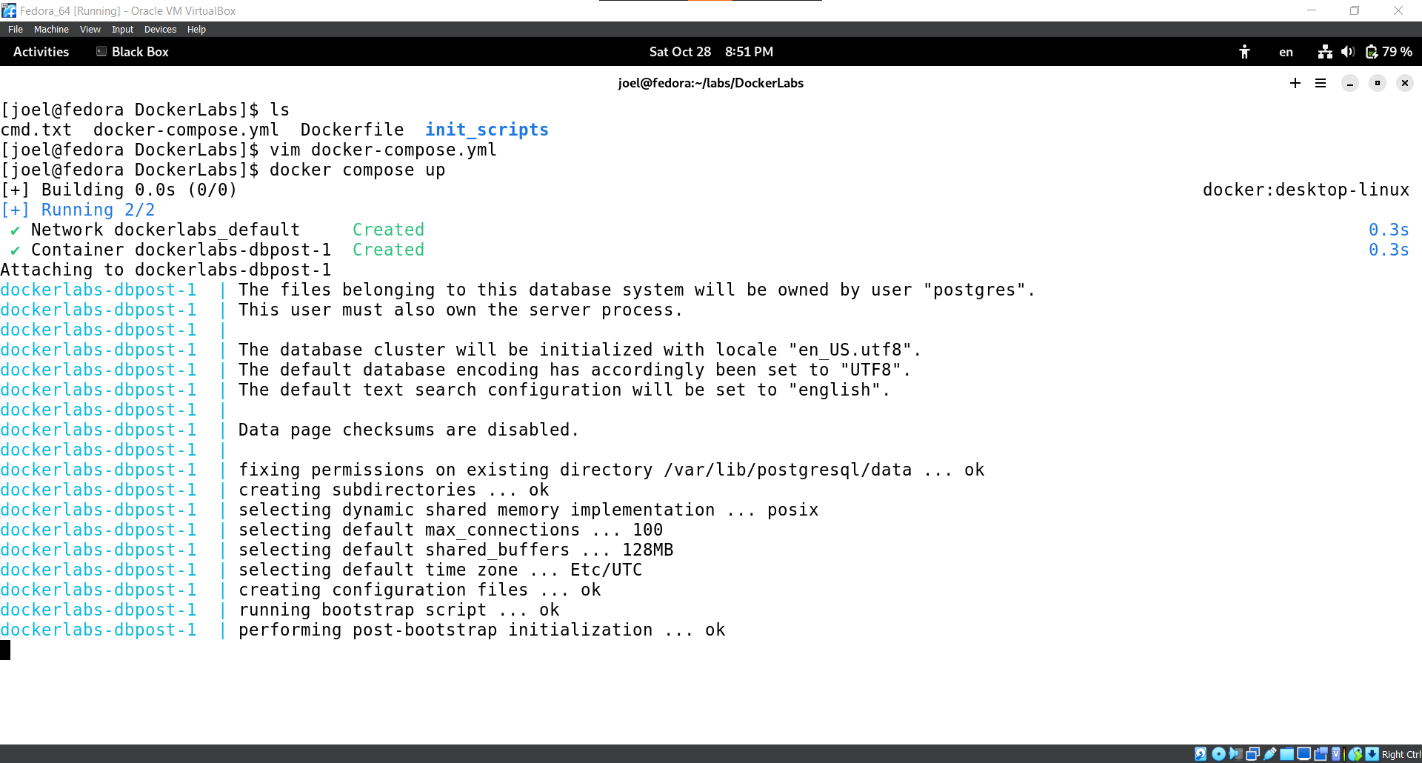


Рисунок - Создание и запуск контейнера,  
с помощью команды «docker compose up»

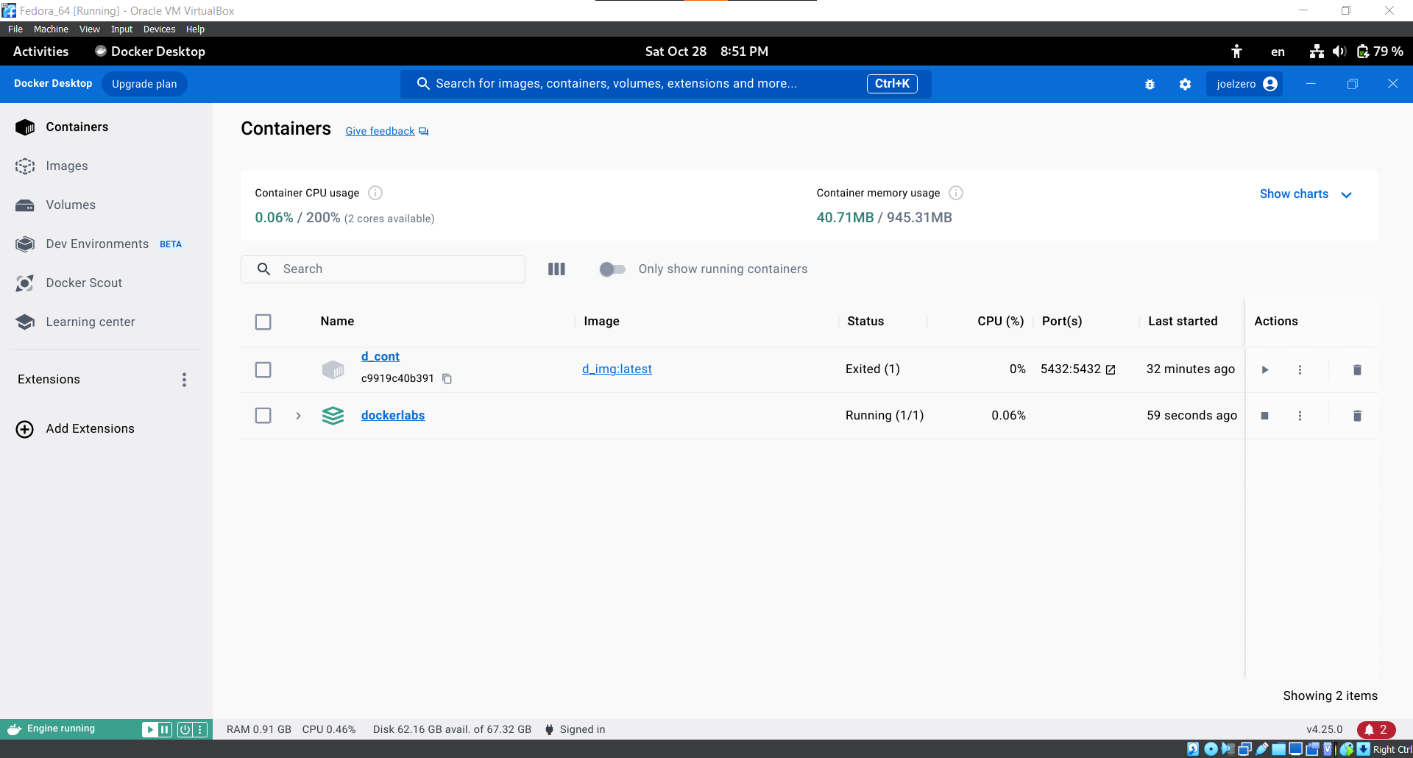


Рисунок - Контейнер «dockerlabs» создан, с помощью команды  
«docker compose up» и работает

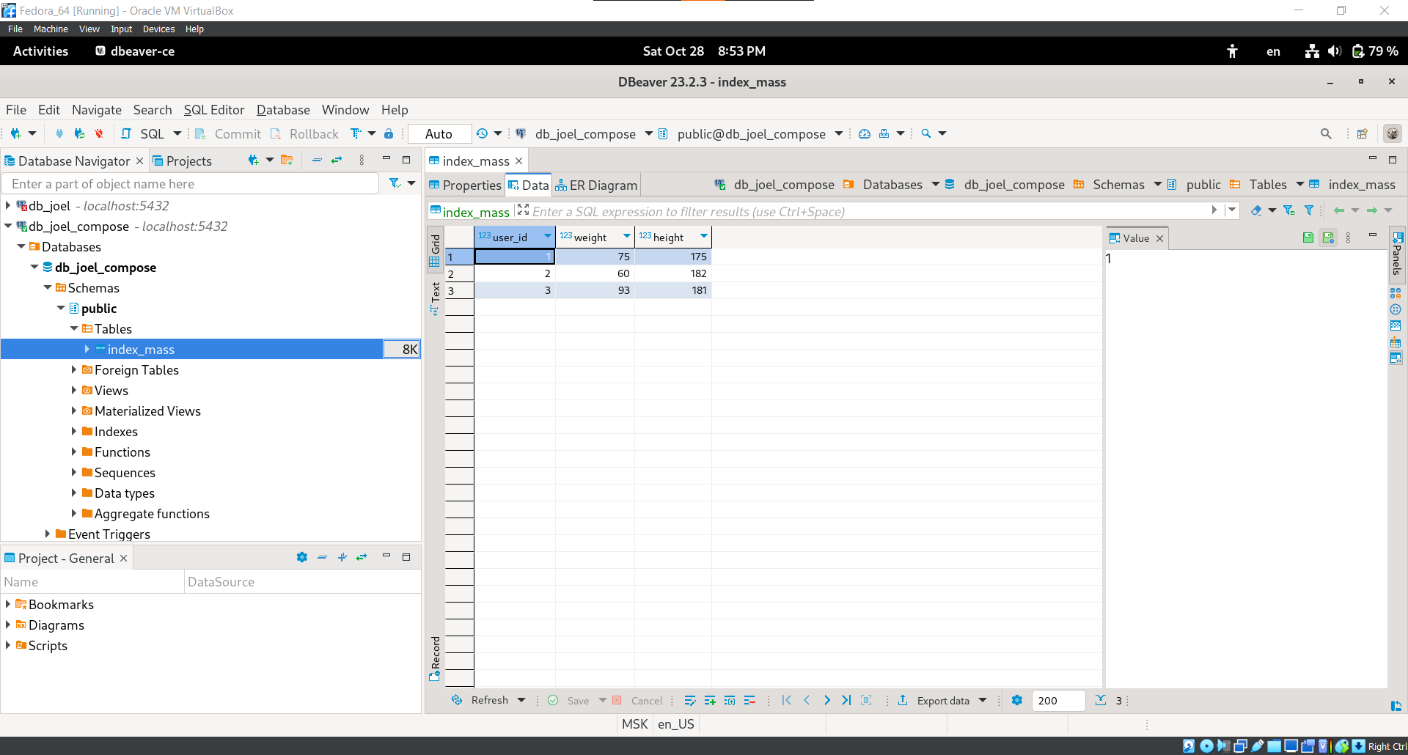


Рисунок - Подключение к базе данных контейнера «dockerlabs»  
выполнено успешно

Описание файла «docker-compose.yml». Строка «version» указывает версию синтаксиса Docker Compose, которую следует использовать при интерпретации файла. Строка «services» определяет сервисы, которые будут запущены. Строка «dbpost» – это имя сервиса, в данном случае, dbpost. Строка «image: postgres:latest» указывает образ PostgreSQL, который будет использоваться для создания контейнера. В данном случае, используется последняя версия образа postgres. Строка «environment» определяет переменные среды для настройки PostgreSQL (как и в Dockerfile, далее три строки перечисляют название БД, имя пользователя и пароль БД). Строка «volumes» описывает монтирование тома (директории) с локальной машины внутрь контейнера, а файл «init.sql», описанные в следующей строке, будет скопирован внутрь контейнера для автоматического исполнения. Строка «ports» описывает порты между хостом и контейнером, это, соответственно, порты 5432:5432, описанные в строке ниже.

**Ответы на контрольные вопросы:**

Вопрос 1: что такое Docker?

Ответ: Docker – это проект с открытым исходным кодом для автоматизации развертывания приложений в виде переносимых автономных контейнеров, выполняемых в облаке или локальной среде. Одновременно с этим, Docker — это компания, которая разрабатывает и продвигает эту технологию в сотрудничестве с поставщиками облачных служб, а также решений Linux и Windows, включая корпорацию Майкрософт.

Вопрос 2: зачем нужен Docker?

Ответ: Docker упрощает развертывание приложений, упаковывая их и все необходимые зависимости в контейнеры, что обеспечивает консистентность среды выполнения. Это позволяет легко масштабировать приложения и уменьшает вероятность конфликтов между зависимостями. Docker также облегчает переносимость приложений, позволяя им работать в различных средах без изменений.

Вопрос 3: что такое docker-образ?

Ответ: Docker-образ представляет собой легковесный, автономный исполняемый пакет, включающий в себя приложение, его зависимости и системные настройки. Этот образ служит основой для создания и запуска контейнеров в среде Docker.

Вопрос 4: что такое docker-контейнер?

Ответ: Docker-контейнер представляет собой запускаемый экземпляр Docker-образа, изолированный от окружающей системы. Он включает в себя приложение, все необходимые зависимости и среду выполнения. Контейнеры обеспечивают легковесную виртуализацию, упрощая развертывание, масштабирование и управление приложениями.

Вопрос 5: что такое volume и зачем он нужен?

Ответ: Volume в Docker представляет собой механизм для постоянного хранения данных вне контейнера. Он используется для разделения данных между контейнерами и хостовой системой, а также для обеспечения устойчивого хранения, не зависящего от жизненного цикла контейнера.

Вопрос 6: что такое docker-compose?

Ответ: Docker Compose — это инструментальное средство, входящее в состав Docker. Оно предназначено для решения задач, связанных с развёртыванием  
проектов.

Вопрос 7: в чём разница между dockerfile и docker-compose?

Ответ: если Dockerfile – это способ описать логику работы одного контейнера, то Docker-compose – это способ описать логику работы сразу нескольких образов.

Вопрос 8: какая команда позволяет отправлять различные задания в запущенный докер-контейнер?

Ответ: для отправки различных заданий в запущенный Docker-контейнер  
используется команда «docker exec -it <container\_id\_or\_name> <command>», где <container\_id\_or\_name> - идентификатор или имя контейнера, а <command>  
- команда, которую необходимо выполнить внутри контейнера. Флаги «-it» обеспечат интерактивный режим и привязку к терминалу контейнера.

Вопрос 9: с помощью какого инструмента можно сохранить важные данные после аварийного отключения контейнера?

Ответ: для сохранения важных данных после аварийного отключения контейнера в Docker, необходимо использовать тома (Volume). Volume обеспечивают постоянное и изолированное хранение данных вне контейнера, что позволяет им сохраняться даже при аварийном отключении или удалении контейнера.

**Вывод:** в ходе лабораторной работы были изучены основы работы программного средства для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации Docker на примере решения задачи развёртывания СУБД Postgres.