Лабораторная работа.

Использование базы данных MySQL в Docker

Цель работы: изучение особенности создания томов в Docker и создания контейнера с сервером базы данных.

Методические указания.

Особенностью работы контейнера является то, что по умолчанию результаты рабо- ты процесса в контейнере не сохраняются. После остановки работы контейнера и запуска программы из образа создаётся новый контейнер. Созданный контейнер не видит резуль- татов работы предыдущего контейнера.

Для того чтобы результаты работы процесса в контейнере сохранились, необходи- мо часть файловой системы операционной системы отобразить в файловую систему кон- тейнера. Для этого надо специальной командой создать том (**volume**). В этом томе можно сохранять результаты работы контейнера. При следующем запуске образа процесс в кон- тейнере сможет использовать данные сохранённые в томе. Фактически определяется пап- ка на компьютере, с которой будет работать запущенный контейнер. Эта папка и её со-

держимое сохраняется после завершения работы контейнера.

Для создания тома используется команда **docker volume** с различными опциями.

Создание тома: **docker volume create <имя тома>**

Получение списка существующих томов: **docker volume ls**

Получение подробной информации о томе: **docker volume inspect <имя тома>**

Удаление тома **docker volume rm <имя тома>**

Удаление томов, которые не использует ни один контейнер: **docker volume prune**

Если монтируется *пустой* том в каталог контейнера, где уже есть файлы, то эти файлы не удалятся, а будут скопированы в том. Этим можно пользоваться, когда нужно сохранить данные контейнера.

Если вы монтируется *непустой* том или каталог с хоста в каталог, где уже есть

файлы, то эти файлы не удалятся, а просто будут скрыты. Видно будет только то, что есть в томе или каталоге на хосте. Это похоже на простое монтирование в Linux.

Docker также позволяет монтировать каталог хоста в контейнер. Эта технология работает для разных операционных систем и для разных файловых систем. Например, можно запустить контейнер Docker в Windows и примонтировать каталог на каком-нибудь диске. Для монтирования каталогов используется команда

# docker run -- mount <параметры монтирования>

При работе с **--mount** параметры представлены как пары вида ключ-значение: **key=value**.

Эти пары разделяют запятыми. Вот часто используемые параметры **--mount**:

**type** — тип монтирования. Значением для соответствующего ключа могут выступать bind, volume или tmpfs.

* bind – монтирование каталога хоста;
* volume – монтирование тома;
* tmpfs – монтирование во временное файловое хранилище. Это временное храни- лище располагается в оперативной памяти. При закрытии контейнера данные из tmpfs удаляются.

**source** — источник монтирования. Для именованных томов это — имя тома. Для неиме- нованных томов этот ключ не указывают. При монтировании каталога хоста указывается полный путь к каталогу. Этот ключ может быть сокращён до src.

**destination** — путь, к которому файл или папка монтируется в контейнере. Этот ключ мо- жет быть сокращён до dst или target.

**readonly** — монтирует том, который предназначен только для чтения. Использовать этот ключ необязательно, значение ему не назначают.

Сервер базы данных можно запустить в контейнере из соответствующего образа.

Файлы базы данных необходимо хранить на томе. Контейнерная реализация обеспечивает лёгкую и быструю переносимость небольшой базы данных и независимость её от опера- ционной системы хоста.

Указания по выполнению лабораторной работы

1. Работа с томом. Для примера запишем текстовый файл в том.
   1. Для создания и работы с томами используются специальные команды. Справку по командам работы с томом можно получить, используя команду

docker volume –help

Изображение выглядит как текст

Автоматически созданное описание

* 1. Для создания тома используйте команду

docker volume create six6 six6 – это имя тома.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Используя соответствующую команду, посмотрите список томов.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Используя команду inspect, посмотрите информацию о созданном томе.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Запустите контейнер, подключив к нему том. В контейнере том примонтируем как каталог **/data**

docker run --rm -v six6:/data -it ubuntu bash

Ключ --rm указывает, что контейнер удаляется после завершения работы.

Изображение выглядит как текст, внутренний, экран, набор

Автоматически созданное описание

* 1. Создайте в каталоге **/data** файл, text.txt и занесите туда какую-нибудь информа- цию.

Изображение выглядит как текст

Автоматически созданное описание

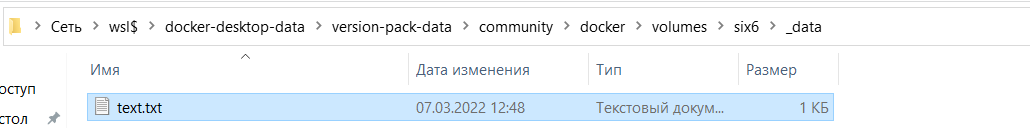
* 1. Закройте контейнер ubuntu.
  2. Запустите на выполнение другой образ, подключив к нему том six6. Проверьте наличие файла и содержимое файла text.txt. Поясните получившийся результат.

Изображение выглядит как текст

Автоматически созданное описание

Так как мы монтируем папку, а не копируем ее содержимое, то мы, по сути, работаем с содержимым папки, которая находится где-то у нас на компьютере

* 1. Покажите, в каком месте на компьютере хранится том six6.



1. Подключение каталога хоста.
   1. Подключите к контейнеру каталог, находящейся на хосте, используя команду наподобие следующей

docker run --rm --mount type=bind,source=c:/avatars,target=/ava -it ubuntu bash



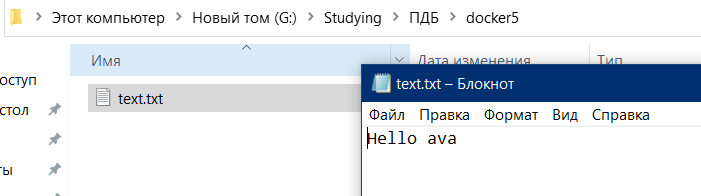
В контейнер монтируется каталог c:\avatars. В контейнере этот каталог называется

/ava.

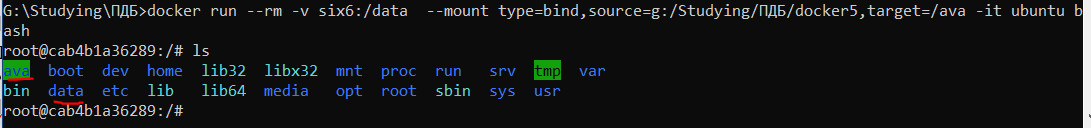
* 1. Используя контейнер, внесите изменения в содержимое примонтированного ката- лога. Определите, как изменилось содержимое каталога после закрытия контейне- ра.

Изображение выглядит как текст

Автоматически созданное описание



* 1. Используя команду --mount подключите к контейнеру том **six6** и какой-нибудь каталог хоста.



* 1. Подключите один том или каталог к двум разным контейнерам. Покажите, что из- менения, сделанные в одном контейнере, видны из другого контейнера.

Изображение выглядит как текст, снимок экрана, монитор, ноутбук

Автоматически созданное описание

1. Работа с MySQL
   1. Установите MySQL с официального репозитория командой

docker pull mysql/mysql-server:latest

Параметр latest позволяет установить последнюю доступную версию MySQL.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Разверните контейнер с MySQL с помощью команды docker run --name=mysqltest -d mysql/mysql-server:latest mysqltest – это имя создаваемого контейнера;

флаг -d – это сокращение для --detach.

Данная команда запускает контейнер в фоновом режиме. Это позволяет использо- вать терминал, из которого запущен контейнер, для выполнения других команд во время работы контейнера.

* 1. Убедитесь в том, что контейнер запущен с помощью команды

docker ps

Команда docker ps выводит список контейнеров, она аналогична команде

docker container ls

Изображение выглядит как текст

Автоматически созданное описание

* 1. При создании контейнера с базой MySQL автоматически создаётся пароль для подключения к серверу MySQL. Для просмотра пароля надо воспользоваться ко- мандой

docker logs <имя контейнера>

В пункте GENERATED ROOT PASSWORD приведён пароль. Он понадобится для входа на сервер MySQL.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Для подключения к запущенному контейнеру и выполнения на нём команд ис- пользуется команда docker exec. Для работы на контейнере mysqltest запустим на нём командный интерпретатор bash.

docker exec -it mysqltest bash

* 1. Для подключения к SQL серверу, используется команда mysql. Ознакомьтесь с параметрами данной команды, используя справку

mysql --help

Нас, прежде всего, интересует вход под учётной записью root с паролем.

mysql -u root -p

После задания этой команды сервер предложит ввести пароль. Пароль надо скопи- ровать, вставить и нажать клавишу **Enter**. Учтите, что после вставки пароль не отображается, изображение на экране не меняется.

Изображение выглядит как текст

Автоматически созданное описание

* 1. После успешного ввода логина и пароля запустится интерпретатор команд SQL. Проверьте его работоспособность какой-нибудь командой.
  2. Для смены пароля используйте следующую команду

ALTER USER ‘root’@’localhost’ IDENTIFIED BY ‘New\_Password’;

Изображение выглядит как текст

Автоматически созданное описание

* 1. Определите, где в контейнере хранятся файлы базы данных.

Изображение выглядит как текст

Автоматически созданное описание

1. Создадим простейшую базу данных, хранящую необходимую нам информацию.
   1. В файловой системе хоста создайте папку для хранения файлов скриптов, напри- мер **MySQL\sql-scripts**.
   2. Создадим поля для нашей БД и сохраним их в файл **CreateTable.sql**. Этот файл поместим в папку **sql-scripts**. В качестве полей используем информацию о студен- тах:

CREATE TABLE students ( ID CHAR(5),

first\_name varchar(25), last\_name varchar(25), groupe varchar(20), PRIMARY KEY (ID)

);

Создадим файл с данными **InsertData.sql.** Этот файл также поместим в папку

# sql-scripts.

INSERT INTO students (ID, first\_name, last\_name, groupe) VALUES ('1', 'Ingmar', 'Bergman', '00001');

Создадим файл скрипта students.txt. Этот файл поместим в папку **MySQL**

FROM mysql

ENV MYSQL\_DATABASE students

COPY ./sql-scripts/ /docker-entrypoint-initdb.d/

* 1. Создайте образ, используя файл скрипта и задайте имя новому образу

mysql:students

Изображение выглядит как текст, оранжевый

Автоматически созданное описание

* 1. Создайте том для хранения файлов базы данных с именем db\_students.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Файлы базы данных MySQL в контейнере хранятся в папке /var/lib/mysql. К этой папке надо подключить том при запуске контейнера из образа. Для связи с кон- тейнером назначим внешний порт 3306, для связи с MySQL внутренний порт 3306. Зададим пароль для входа в базу данных 5root5. Также дадим контейнеру имя stu- dents1.

docker run -d -p 3306:3306 -v db\_students:/var/lib/mysql --name students1 -e MYSQL\_ROOT\_PASSWORD=5root5 mysql:students

* 1. Подключитесь к контейнеру, используя команду docker exec.

Проверьте, что файлы скриптов для создания базы данных скопировались. Убеди- тесь, что файлы базы данных находятся в каталоге /var/lib/mysql.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

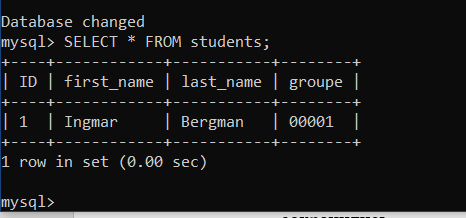
Автоматически созданное описание

* 1. Подключитесь к серверу MySQL. Проверьте, что автоматически создалась база

данных students, и она содержит нужные данные. Объясните механизм создания базы students.

Изображение выглядит как текст

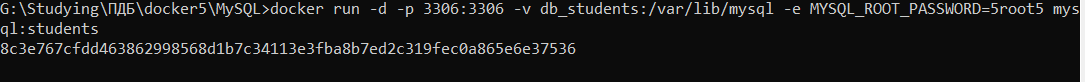
Автоматически созданное описание



Секрет – в инструкции COPY ./sql-scripts/ /docker-entrypoint-initdb.d/. Она скопировала наши скрипты в папку, в которой содержатся скрипты для инициализации бд. Таким образом скрипты запустились и мы имеем то, что имеем.

* 1. Внесите изменения в базу данных, выйдите из контейнера и остановите контейнер. Запустите новый контейнер из образа. Убедитесь, что изменения в базе данных сохранились.





Изображение выглядит как текст

Автоматически созданное описание