**Московский государственный технический университет им. Н. Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Автоматизация разработки и эксплуатации ПО»

# Отчет по рубежному контролю №1

Выполнил:

Студентка группы ИУ5-71Б

Ноздрова Валентина

Дата: 01.10.2022

Подпись:

Проверил:

Дата:

Подпись:

Москва, 2022 г.

**План и задачи лабораторной работы:**

Часть 1. Базовые команды Docker

1. Подготовка рабочего окружения
2. Образа - docker pull , docker images
3. Метки и удаление образа - docker tag
4. Запускаем контейнер - docker run , docker logs
5. Списки контейнеров - docker ps
6. Подключаемся к контейнеру - docker exec
7. Список изменений - docker diff
8. Завершаем контейнер - docker stop , docker kill , docker rm
9. Не теряем данные - docker volume
10. Контейнер Adminer
11. Сети - docker network

Часть 2. Продвинутая работа с Docker

1. Настройка базы данных
2. Запускаем Adminer
3. Запускаем свой сервис
4. Подробнее про сборку образа
5. Оптимизируем сборку
6. Многоэтапная сборка
7. Делимся образом docker push

**Ход выполнения работы:**

1.1 Подготовка рабочего окружения

Запустим hello-world:

```
user@devopsiu5:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

1.2. Образа - docker pull , docker images

Загрузим образ mysql с помощью команды docker pull mysql:

```
user@devopsiu5:~$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
051f419db9dd: Pull complete
7627573fa82a: Pull complete
a44b358d7796: Pull complete
95753aff4b95: Pull complete
a1fa3bee53f4: Pull complete
f5227e0d612c: Pull complete
b4b4368b1983: Pull complete
f26212810c32: Pull complete
d803d4215f95: Pull complete
d5358a7f7d07: Pull complete
435e8908cd69: Pull complete
Digest: sha256:b9532bledea72b6cee12d9f5a78547bd3812ea5db842566e17f8b3329led2921
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
user@devopsiu5:~$
```

Проверим с помощью команды docker image ls:

```
user@devopsiu5:~$ docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
mysql         latest    43fcfca0776d   2 weeks ago     449MB
hello-world   latest    feb5d9fea6a5   12 months ago   13.3kB
```

Установим mysql версии 5.7.39 и убедимся, что теперь есть два образа с разными метками:

```
user@devopsiu5:~$ docker pull mysql:5.7.39
5.7.39: Pulling from library/mysql
a2a0026033lc: Pull complete
6d8167f2fcbe: Pull complete
32454e9854ca: Pull complete
473e2917b0d5: Pull complete
5173f8104ec8: Pull complete
32e218351f9a: Pull complete
fc9ela82359a: Pull complete
c602a3ea2ce7: Pull complete
3c9ea9927039: Pull complete
dfblb236c7fc: Pull complete
e2ad62bd72a7: Pull complete
Digest: sha256:94fe67a0400le9841f68fll4c8e9b5231c1d012e6b00d3b8ade42c0c5e239a0f
Status: Downloaded newer image for mysql:5.7.39
docker.io/library/mysql:5.7.39
user@devopsiu5:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
mysql         5.7.39    aa803eda0f25   7 days ago      433MB
mysql         latest    43fcfca0776d   2 weeks ago     449MB
hello-world   latest    feb5d9fea6a5   12 months ago   13.3kB
```

1.3. Метки и удаление образа - docker tag

Назначим одному из образов тэг с помощью команды docker tag:

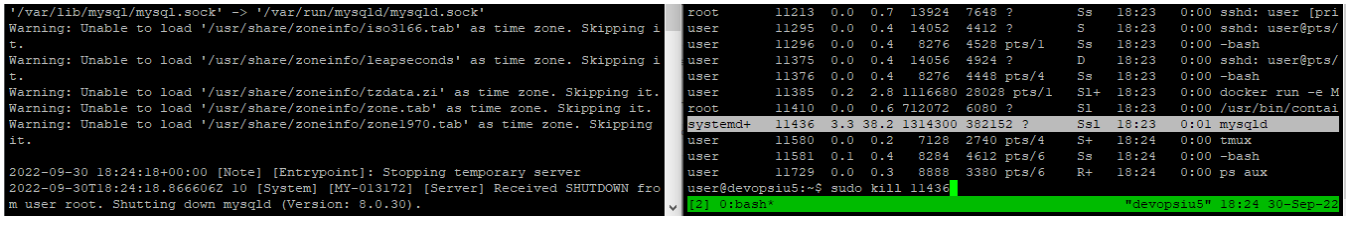
```
user@devopsiu5:~$ docker tag mysql mysql:8.0-iu5
user@devopsiu5:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
mysql         5.7.39    aa803eda0f25   7 days ago      433MB
mysql         8.0-iu5   43fcfca0776d   2 weeks ago     449MB
mysql         latest    43fcfca0776d   2 weeks ago     449MB
hello-world   latest    feb5d9fea6a5   12 months ago   13.3kB
```

Уберем тэг с помощью команды docker rmi. Ненужный образ удалим с помощью команды docker image rm:

```
user@devopsiu5:~$ docker rmi mysql:8.0-iu5
Untagged: mysql:8.0-iu5
user@devopsiu5:~$ docker image rm mysql:5.7.39
Untagged: mysql:5.7.39
Untagged: mysql@sha256:94fe67a04001e9841f68f114c8e9b5231c1d012e6b00d3b8ade42c0c5e2
39a0f
Deleted: sha256:aa803eda0f25baa8886e951f76473a28bec9938508849bc70a0ef340c0077956
Deleted: sha256:c907abd0bf6a56405e7e821daca8c88fe6da6676f25d59d5795abffc7a72c4f3
Deleted: sha256:9a4f84f3d44b3d0a172654d63d8905c2b4e68877c9954ef7c2fda6969c8868a1
Deleted: sha256:2f2d333bb03073be6aa6fb6726a1c1e424e86659d75005a750261c2b770b3027
Deleted: sha256:9558fa5aladf6454eb1a98cc56da437378134c92f8c2aea924993aeac08dfb35
Deleted: sha256:1c92550f4b1883bb5b6bd010da276cdf4a62bed911ee50f396eedeffe5390c8f
Deleted: sha256:be90241cd04b52993762d16bcc7b5a23feece7f4d892aa83e6176f1dc0dbbb30
Deleted: sha256:e6b068a8b2f99bc0e86decded2418309392dd0f4deedceae00157b6a4dd327d7
Deleted: sha256:3348aaa016098f8fa8b5e81960346d97033a27ade742d96563085632 44fbb59e
Deleted: sha256:0e03380d21da0e2f79826bc7257959db267797a9859569cbb972a7bdbe655e43
Deleted: sha256:1cc48a29c4dbbf4050113a26eb212a2b4107e6b9499580a3c7711452268f6759
Deleted: sha256:ac0d5e75f97f0db819766a55b8b100271de48bed81fc50f05c0bd2c220c6b2c5
```

1.4. Запускаем контейнер - docker run , docker logs

Запустим контейнер с СУБД командой docker run –e MYSQL_ROOT_PASSWORD=password mysql. Поднимем еще одно окно и прервем процесс контейнера, который подвесил консоль без ее завершения.




Теперь попробуем запустить в фоновом режиме с аргументом -d , кроме того, дадим ему вменяемое имя, с помощью команды docker run -d -e MYSQL_ROOT_PASSWORD=password --name db1 mysql. В ответ Docker выдал нам хэш-ID контейнера.

Посмотрим логи контейнера по его имени командой docker logs db1:

```
user@devopsiu5:~$ docker logs db1
2022-09-30 18:31:41+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server
8.0.30-1.el8 started.
2022-09-30 18:31:42+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2022-09-30 18:31:42+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server
8.0.30-1.el8 started.
2022-09-30 18:31:42+00:00 [Note] [Entrypoint]: Initializing database files
2022-09-30T18:31:42.653952Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-ho
st-cache' is deprecated and will be removed in a future release. Please use SET GL
OBAL host_cache_size=0 instead.
2022-09-30T18:31:42.654146Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysq
ld 8.0.30) initializing of server in progress as process 78
2022-09-30T18:31:42.666018Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization
has started.
2022-09-30T18:31:43.451918Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization
has ended.
2022-09-30T18:31:45.246611Z 6 [Warning] [MY-010453] [Server] root@localhost is cre
ated with an empty password ! Please consider switching off the --initialize-insec
ure option.
2022-09-30 18:31:49+00:00 [Note] [Entrypoint]: Database files initialized
2022-09-30 18:31:49+00:00 [Note] [Entrypoint]: Starting temporary server
2022-09-30T18:31:49.982749Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-ho
st-cache' is deprecated and will be removed in a future release. Please use SET GL
OBAL host_cache_size=0 instead.
2022-09-30T18:31:49.984593Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysq
ld 8.0.30) starting as process 125
2022-09-30T18:31:50.222357Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization
has started.
2022-09-30T18:31:50.987968Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization
has ended.
2022-09-30T18:31:51.629871Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem
 is self signed.
```

## 1.5. Списки контейнеров - docker ps

Посмотрим списки контейнеров с помощью команды docker ps:

```
user@devopsiu5:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
 PORTS                 NAMES
fe9d2837d610   mysql     "docker-entrypoint.s…"   11 minutes ago   Up 11 minutes
 3306/tcp, 33060/tcp   db1
8fefbf47be27   mysql     "docker-entrypoint.s…"   2 hours ago      Up 2 hours
 3306/tcp, 33060/tcp   priceless_turing
```

## 1.6. Подключаемся к работающему контейнеру - docker exec

Подключимся (запустим еще один процесс внутри контейнера) с помощью команды docker exec:

```
user@devopsiu5:~$ docker exec -it db1 /bin/bash
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show schemas;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.20 sec)

mysql>
```

1.7. Список изменений - docker diff

Просмотрим список изменений в слое на ФС с помощью команды docker diff:

```
user@devopsiu5:~$ docker diff db1
C /run
C /run/mysqld
A /run/mysqld/mysqlx.sock.lock
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
A /run/mysqld/mysqld.sock.lock
A /run/mysqld/mysqlx.sock
C /root
A /root/.bash_history
A /root/.mysql_history
```

1.8. Завершаем контейнер - docker stop, docker kill, docker rm

Завершим контейнер нежно с помощью stop, сразу завершим с помощью kill и удалим остатки (из спика завершенных, включая логи контейнера) с помощью rm:

```
user@devopsiu5:~$ docker stop db1
db1
```

```
2022-09-30T18:50:36.890412Z 0 [System] [MY-013172] [Server] Received SHUTDOWN from
 user <via user signal>. Shutting down mysqld (Version: 8.0.30).
```

```
user@devopsiu5:~$ docker start db1
db1
user@devopsiu5:~$ docker kill db1
db1
```

```
user@devopsiu5:~$ docker rm db1
db1
user@devopsiu5:~$ docker logs db1
Error: No such container: db1
```

1.9. Не теряем данные - docker volume

Запустим контейнер с томом и внесем изменения:

```
user@devopsiu5:~$ docker run --rm -d -v mysql:/var/lib/mysql -v mysql_config:/etc/
mysql --name db1 -e MYSQL_ROOT_PASSWORD=password  mysql
aebe7b20ba854b48cee3e25e5e39786933fe96d425a22797bc4207aeeca8f7f8
user@devopsiu5:~$ docker exec -it db1 mysql -ppassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> create database testdb;
Query OK, 1 row affected (0.08 sec)

mysql> create database blog;
Query OK, 1 row affected (0.10 sec)

mysql> show schemas;
+--------------------+
| Database           |
+--------------------+
| blog               |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testdb             |
+--------------------+
6 rows in set (0.07 sec)
```

Завершим контейнер, перезапустим и убедимся, что базы testdb и blog не исчезли.

```
Bye
user@devopsiu5:~$ docker stop db1
db1
user@devopsiu5:~$ docker run --rm -d -v mysql:/var/lib/mysql -v mysql_config:/etc/
mysql --name db1 -e MYSQL_ROOT_PASSWORD=password  mysql
c3f876adbaf95f5a3a518e515394cebe3974168ae0488508078443eb3456eaf3
user@devopsiu5:~$ docker exec -it db1 mysql -ppassword -e "show schemas;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+--------------------+
| Database           |
+--------------------+
| blog               |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testdb             |
+--------------------+
```

Просмотрим список томов:

```
user@devopsiu5:~$ docker volume ls
DRIVER    VOLUME NAME
local     0e92f0d056db816c0183lcef0efc3a6040770722227c83cc11cc8193cf094faa3
local     8ba4f9354b5e798bd78f389172308e10c6b0532961a1b1cc2877d5de2b8aa5c0
local     26da6ab6d743fe8887c5ebe0f1532b97ced0fa2cf5c23c8d162cdede274484aa
local     a4f42029d295c5f4fd45a9250f6d56875716b1e36b220015134fbcff93427caf
local     a82e19f1e757f7a96a31ec217106ca03d1d1440c9184be93d4d341ce22cdcd2a
local     ad9f7fadc0e7182a444684b24a739a9afcfebea87155300a857201abe87c26be
local     e9f0432402359534545153ebfc2d46da3ae85cdcf080b90403ca40952ef46485
local     mysql
local     mysql_config
```

Создадим новый том и просмотрим информацию о нем:

```
user@devopsiu5:~$ docker volume create test
test
```

```
user@devopsiu5:~$ docker volume inspect test
[
    {
        "CreatedAt": "2022-09-30T21:12:11Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/test/_data",
        "Name": "test",
        "Options": {},
        "Scope": "local"
    }
]
```

Удалим том:

```
user@devopsiu5:~$ docker volume rm test
test
```

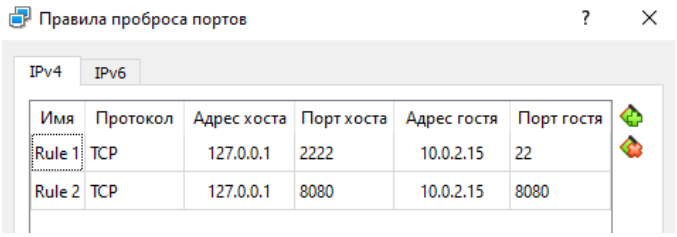
Очистим лишние тома:

```
user@devopsiu5:~$ docker volume prune
WARNING! This will remove all local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Deleted Volumes:
ad9f7fadc0e7182a444684b24a739a9afcfebea87155300a857201abe87c26be
0e92f0d056db816c01831cef0efc3a604077072227c83cc11cc8193cf094faa3
e9f043240235953454515 3ebfc2d46da3ae85cdcf080b90403ca40952ef46485

Total reclaimed space: 405.1MB
```

1.10. Контейнер Adminer

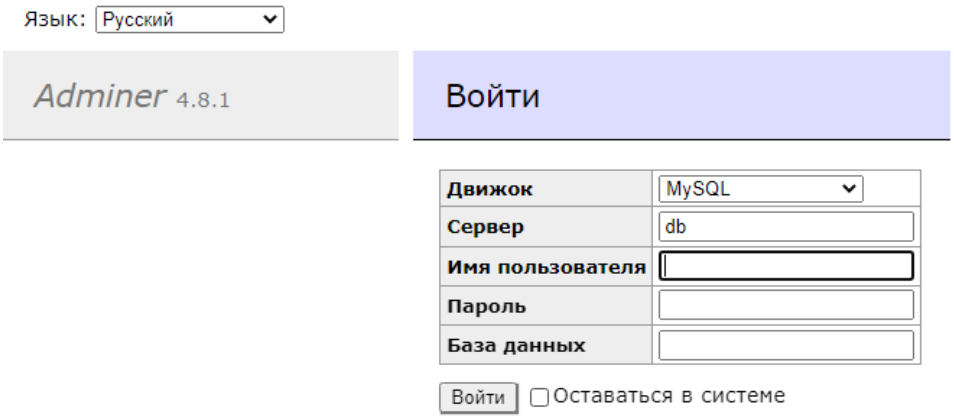
Настроим дополнительный проброс портов




Запустим образ Adminer. Для того, чтобы попасть из виртуалки внутрь контейнера на порт 8080 укажем ключ -p HostPort:ContainerPort:

```
Status: Downloaded newer image for adminer:latest
```

Подключимся в браузере на хосте к http://127.0.0.1:8080/:

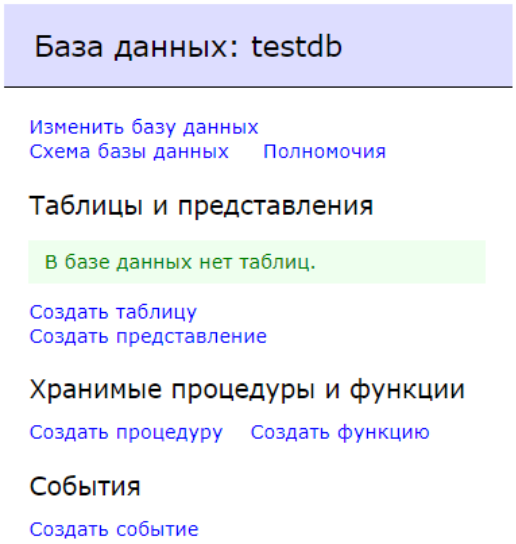



1.11. Сети - docker network

Завершим предыдущий контейнер Adminer и запустим новый с параметром --link Container:AliasName

```
user@devopsiu5:~$ docker rm -f adminer
adminer
user@devopsiu5:~$ docker run -d -p 8080:8080 --link db1:mysql --name adminer admin
er
a138695bff8dfa0aa48be1009158415fd82777fd1f67bf43db6e78fb6b8b6cbc
```

Теперь подключимся к базе по ее Alias из контейнера adminer:




Создадим новую сеть:

```
user@devopsiu5:~$ docker network create cluster
520b1d410b4b3de44686e8b36757ee92377634d3e5eff92cdbef5935768a0954
```

Проверим как создалась сеть с параметрами по умолчанию:

```
user@devopsiu5:~$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
832666c7f07a   bridge    bridge    local
520b1d410b4b   cluster   bridge    local
75e1d4cb28e4   host      host      local
2d8fce9075e3   none      null      local
user@devopsiu5:~$ docker network inspect cluster
[
    {
        "Name": "cluster",
        "Id": "520b1d410b4b3de44686e8b36757ee92377634d3e5eff92cdbef5935768a0954",
        "Created": "2022-10-01T08:45:38.9643447Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Проверим в какой сети работают сейчас Adminer и MySQL:

```
user@devopsiu5:~$ docker inspect db1 | egrep "IPAddress|Gateway|IPPrefixLen"
            "SecondaryIPAddresses": null,
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.3",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.3",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
```

```
user@devopsiu5:~$ docker inspect adminer | egrep "IPAddress|Gateway|IPPrefixLen"
            "SecondaryIPAddresses": null,
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.4",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.4",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
```

Теперь пересоздадим контейнеры СУБД и Adminer в этой сети:

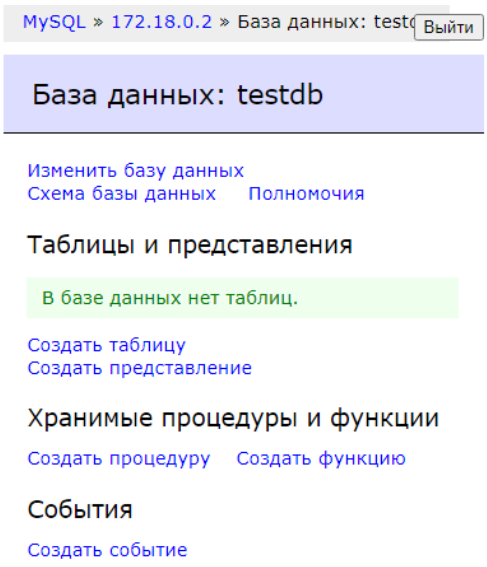
```
user@devopsiu5:~$ docker run --rm -d -v mysql:/var/lib/mysql -v mysql_config:/etc/
mysql --name db1 -e MYSQL_ROOT_PASSWORD=password --net cluster mysql
3219beaa8ebe7a01d1ca8e91bcd3989484fa0c42ca68dc7bce18faabd9672fec
user@devopsiu5:~$ docker run -d -p 8080:8080 --net cluster --name adminer adminer
fc3bd71969514ccc593f73d7a98ec3cbb19100e743fcecc36c77362be18e5045
user@devopsiu5:~$
```

Проверим их IP-адреса:

```
user@devopsiu5:~$ docker inspect db1 | egrep "IPAddress|Gateway|IPPrefixLen"
            "SecondaryIPAddresses": null,
            "Gateway": "",
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
```

```
user@devopsiu5:~$ docker inspect adminer | egrep "IPAddress|Gateway|IPPrefixLen"
            "SecondaryIPAddresses": null,
            "Gateway": "",
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.3",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
```

Пробуем подключиться к базе по IP-адресу:



Попробуем запустить образ для диагностики сети в интерактивном режиме и проверить сеть контейнеров с помощью nmap.

```
                    dP            dP                           dP
                    88            88                           88
88d888b. .d8888b. d8888P .d8888b. 88d888b. .d8888b. .d8888b. d8888P
88'  `88 88ooood8   88   Y8ooooo. 88'  `88 88'  `88 88'  `88   88
88    88 88.  ...   88         88 88    88 88.  .88 88.  .88   88
dP    dP `88888P'   dP   `88888P' dP    dP `88888P' `88888P'   dP

Welcome to Netshoot! (github.com/nicolaka/netshoot)



 ca4be19649ed ~

 ca4be19649ed ~ nmap db1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-01 09:06 UTC
Nmap scan report for db1 (172.18.0.2)
Host is up (0.000011s latency).
rDNS record for 172.18.0.2: db1.cluster
Not shown: 999 closed tcp ports (reset)
PORT     STATE SERVICE
3306/tcp open  mysql
MAC Address: 02:42:AC:12:00:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.42 seconds

 ca4be19649ed ~ nmap adminer
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-01 09:06 UTC
Nmap scan report for adminer (172.18.0.3)
Host is up (0.000024s latency).
rDNS record for 172.18.0.3: adminer.cluster
Not shown: 999 closed tcp ports (reset)
PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 02:42:AC:12:00:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

Теперь выйдем и грохнем все контейнеры:

```
user@devopsiu5:~$ docker stop $(docker ps -a -q)
ca4be19649ed
fc3bd7196951
3219beaa8ebe
3568c12e3216
bfc6b0dc2568
0ad59e06cld1
8fefbf47be27
5728f3c31046
02c646f253a0
08c5902e25cd
```

## 2.1. Настройка базы данных

```
user@devopsiu5:~$ docker run --rm -d \
>   -v mysql:/var/lib/mysql \
>   -v mysql_config:/etc/mysql \
>   --name mysql \
>   -e MYSQL_ROOT_PASSWORD=password \
>   -e MYSQL_DATABASE=blog \
>   --net cluster \
>   mysql
47b6e612574b508ea0581c0afc1a1531c1f250cbc9f54197214d3df437a2a5bb
```

Подключимся к рабочей базе и создадим таблицу posts в базе blog

```
user@devopsiu5:~$ docker exec -it mysql mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE TABLE blog.posts (
    -> id INT NOT NULL AUTO_INCREMENT,
    -> title varchar(255),
    -> created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    -> PRIMARY KEY (id)
    -> );
Query OK, 0 rows affected (0.28 sec)
```

Выйдем из контейнера с помощью ctrl + D

## 2.2. Запускаем Adminer

Запустим Adminer в той же сети. Подключимся к http://127.0.0.1:8080/. В открывшемся интерфейсе подключимся к базе.


Перейдем в нее и добавим несколько записей с помощью "Новая запись".

| Изменить | id | title | created |
|---|---|---|---|
| редактировать | 1 | title1 | 2022-10-01 09:19:08 |
| редактировать | 2 | title1 | 2022-10-01 09:19:09 |
| редактировать | 3 | title2 | 2022-10-01 09:19:13 |
| редактировать | 4 | *NULL* | 2022-10-01 09:19:15 |

## 2.3. Запускаем свой сервис

Загрузим себе материалы лабораторной и соберем образ с помощью команды docker build.

Запустим наш сервис (собранный ранее образ step1), передадим ему параметры подключения к базе через переменные окружения (с помощью опции -e), а также подключим его к контейнеру mysql:

```
user@devopsiu5:~$ docker run --net cluster -d -p 8000:8000 -e MYSQL_HOST=mysql -e
MYSQL_USER=root -e MYSQL_PASS=password -e MYSQL_DB=blog --name step1 step1
4ca532022b8fc5f829ce44707le37a3flfce0a66acd7147875813ee48c5281e0
```

Проверим работу нашего приложения выполнив команду curl:

```
user@devopsiu5:~$ curl localhost:8000/posts
[{"id":"1","title":"title1","created":"2022-10-01 09:19:08"},{"id":"2","title":"ti
tle1","created":"2022-10-01 09:19:09"},{"id":"3","title":"title2","created":"2022-
10-01 09:19:13"}]
```

## 2.4. Подробнее про сборку образа

Взглянем на получившиеся слои у образа:

```
user@devopsiu5:~$ docker history step1
IMAGE          CREATED        CREATED BY                                      SIZE
      COMMENT
50eb9258b24f   25 hours ago   /bin/sh -c #(nop)  CMD ["/lab2"]                0B

3533ff9107a7   25 hours ago   /bin/sh -c #(nop)  EXPOSE 8000                  0B

bcc8583625cf   25 hours ago   /bin/sh -c go build -o /lab2                    10MB

aac59d0558f5   25 hours ago   /bin/sh -c #(nop) COPY file:e58d9cac775e9ae8…   3.51
kB
e07714fefddc   25 hours ago   /bin/sh -c go mod download                      794k
B
206b1a569la4   25 hours ago   /bin/sh -c #(nop) COPY file:4e586lc9lallb8b6…   342B

5bb72a69edle   25 hours ago   /bin/sh -c #(nop) COPY file:c5211cbd55fc94ld…   111B

d230084c6c2b   25 hours ago   /bin/sh -c #(nop) WORKDIR /app                  0B

5dd973625d31   3 weeks ago    /bin/sh -c #(nop) WORKDIR /go                   0B

<missing>      3 weeks ago    /bin/sh -c mkdir -p "$GOPATH/src" "$GOPATH/b…   0B

<missing>      3 weeks ago    /bin/sh -c #(nop)  ENV PATH=/go/bin:/usr/loc…   0B

<missing>      3 weeks ago    /bin/sh -c #(nop)  ENV GOPATH=/go               0B

<missing>      3 weeks ago    /bin/sh -c set -eux;  apk add --no-cache --v…   346M
B
<missing>      3 weeks ago    /bin/sh -c #(nop)  ENV GOLANG_VERSION=1.19.1    0B

<missing>      7 weeks ago    /bin/sh -c #(nop)  ENV PATH=/usr/local/go/bi…   0B

<missing>      7 weeks ago    /bin/sh -c [ ! -e /etc/nsswitch.conf ] && ec…   17B

<missing>      7 weeks ago    /bin/sh -c apk add --no-cache ca-certificates   519k
B
<missing>      7 weeks ago    /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B

<missing>      7 weeks ago    /bin/sh -c #(nop) ADD file:2a949686d9886ac7c…   5.54
MB
```

## 2.5. Оптимизируем сборку

Для удобства вынесем переменные окружения в отдельный файл, например config.env:

```
  GNU nano 4.8                          config.env
MYSQL_DB=blog
MYSQL_HOST=mysql
MYSQL_USER=root
MYSQL_PASS=password
```

Соберем образ step2, запустим и проверим, что он работает, аналогично предыдущему:

```
user@devopsiu5:~/devops-lab2$ docker run --net cluster -d -p 8000:8000 --env-file=
config.env step2
85d8d3ae599188d202143d3b7fe5bf0862bcc499033b868b5838b2e4b03b9839
```

```
user@devopsiu5:~/devops-lab2$ curl localhost:8000/posts
[{"id":"1","title":"title1","created":"2022-10-01 09:19:08"},{"id":"2","title":"ti
tle1","created":"2022-10-01 09:19:09"},{"id":"3","title":"title2","created":"2022-
10-01 09:19:13"}]
```

Сравним Dockerfile образов step1 и step2:

```
user@devopsiu5:~/devops-lab2$ cat step2.Dockerfile
FROM golang:alpine

WORKDIR /app

COPY . .
RUN go mod download && go build -o /lab2

EXPOSE 8000
CMD [ "/lab2" ]user@devopsiu5:~/devops-lab2$ cat step1.Dockerfile
FROM golang:alpine

WORKDIR /app

COPY go.mod ./
COPY go.sum ./
RUN go mod download

COPY *.go ./

RUN go build -o /lab2

EXPOSE 8000

CMD [ "/lab2" ]user@devopsiu5:~/devops-lab2$
```

Таким образом, step1 имеет больше слоев.

## 2.6. Многоэтапная сборка

Соберем, запустим и протестируем образ step3:

```
user@devopsiu5:~/devops-lab2$ docker run --net cluster -d -p 8000:8000 --env-file=
config.env --name step3 step3
172b7711a2f700f1958e9dc22b93d147fe37922bd05a589778fe4eale73b58da
user@devopsiu5:~/devops-lab2$ curl localhost:8000/posts
[{"id":"1","title":"title1","created":"2022-10-01 09:19:08"},{"id":"2","title":"ti
tle1","created":"2022-10-01 09:19:09"},{"id":"3","title":"title2","created":"2022-
10-01 09:19:13"}]
```

Dockerfile и history step3:

```
user@devopsiu5:~/devops-lab2$ cat step3.Dockerfile
FROM golang:alpine as builder

WORKDIR /app
COPY . .
RUN go mod download && go build -o /lab2

FROM alpine
COPY --from=builder /lab2 /lab2
EXPOSE 8000
CMD [ "/lab2" ]user@devopsiu5:~/devops-lab2$
```

```
user@devopsiu5:~/devops-lab2$ docker history step3
IMAGE          CREATED         CREATED BY                                      SIZ
E      COMMENT
17f1bd9d73e5   8 minutes ago   /bin/sh -c #(nop)  CMD ["/lab2"]                0B

bda4e66c9b53   8 minutes ago   /bin/sh -c #(nop)  EXPOSE 8000                  0B

6432ea72466c   8 minutes ago   /bin/sh -c #(nop) COPY file:f1e9d3422234e311…   7.5
9MB
9c6f07244728   7 weeks ago     /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B

<missing>      7 weeks ago     /bin/sh -c #(nop) ADD file:2a949686d9886ac7c…   5.5
4MB
```

Принципиальным отличием step3 от предыдущих докерфайлов является копирование исполняемых файлов из другого образа с помощью команды COPY –from=builder.

3. Контрольные вопросы

- Что такое и зачем нужен Docker? Альтернативные системы?

ПО для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации

- Как получить Docker-образ, что это такое?

Образ – шаблон для создания контейнеров.

- Как запустить контейнер? Как получить доступ к его портам?

Запустить контейнер можно с помощью комнады docker run <имя образа>. Узнать, какой порт используется контейнером можно с помощью команды docker port <имя контейнера>.

- Как просмотреть логи контейнера?

С помощью команды docker logs <ID> или docker logs <имя контейнера>

- Как сохранить данные внутри контейнера между его перезапусками?

Для сохранения данных внутри контейнера между перезапусками контейнера используются тома.

- Как подключить контейнеры к одной сети? Какие есть альтернативные варианты?

С помощью указания –net <имя сети> в команде docker run. Альтернативным способом является связь контейнеров через --link Container:AliasName

- Почему контейнеры могут обращаться между собой по имени (хэшу, если его нет)?

Имя (или хэш) и порты связываемого контейнера добавляется в переменные окружения основного контейнера.

- Что такое метки (docker tag)?

Идентификатор, описывающий состояние образа.

- Как удалить ненужные образа и контейнеры?

Удалить образ можно с помощью docker image rm, удалить все образы, на которых не запущены контейнеры можно с помощью docker image prune. Удалить контейнер можно с помощью docker rm.

- Как запустить что-то внутри работающего контейнера?

С помощью команды docker exec.

- Как узнать, какие файлы изменяет программа внутри контейнера?

С помощью команды docker diff.

- Когда происходит завершение контейнера? Как сделать?

Завершение контейнера происходит тогда, когда основной процесс внутри него завершается. Завершить контейнер можно с помощью docker stop (мягкое завершение) или docker kill (принудительное завершение).

- Перезапустите сборку собранного образа, оцените время пересборки, объясните причины.

- К какому число слоев стремиться в образе, правила оптимизации?

Если образ часто скачивается и редко меняется, имеет смысл уменьшать число слоев. Если образ постоянно пересобирается – нужно выделить слои, которые изменяются редко, и добавлять изменения в последнем слое.

- Опишите базовые команды Dockerfile, что они делают, где смотреть документацию?

FROM – указывает базовый образ
COPY /что /куда – скопировать файл из контекста

RUN – запустить команду в образе
ENV – определяет переменные окружения для будущего контейнера

Документация: https://docs.docker.com/engine/reference/builder/

- Что такое контекст сборки, как его оптимизировать?

Контекст – набор локальных файлов и каталогов, на которые можно ссылаться в инструкциях ADD и COPY. Оптимизировать можно кэшированием или многоступенчатой сборкой.