Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Избранные главы информатики

ОТЧЁТ

к лабораторной работе 2

на тему

Работа с Docker

Выполнил: студент группы 253501
Борисевич Александр Михайлович

Проверила: Жвакина Анна Васильевна

Минск 2024

Оглавление

Вариант 4

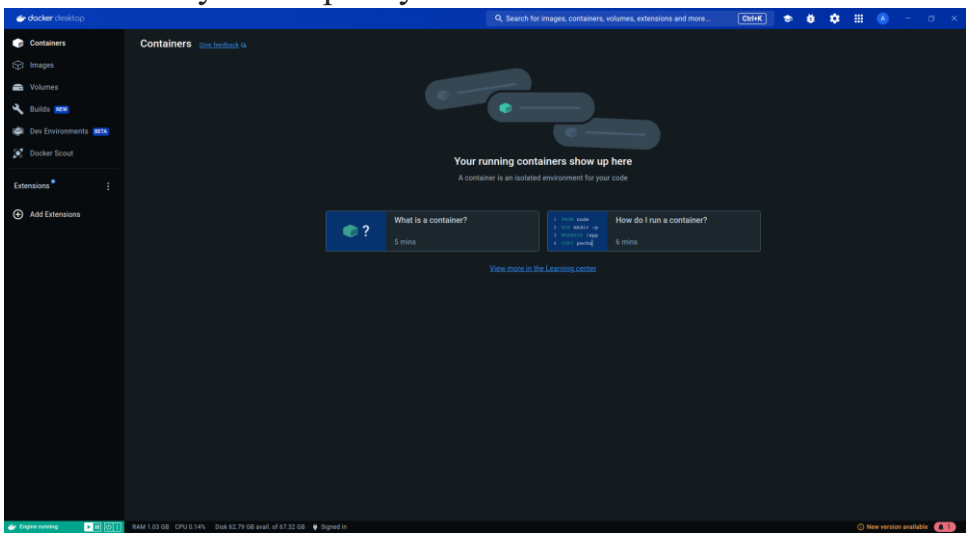
Цель работы

Познакомиться с возможностями и получить практические навыки работы с Docker.

Ход работы

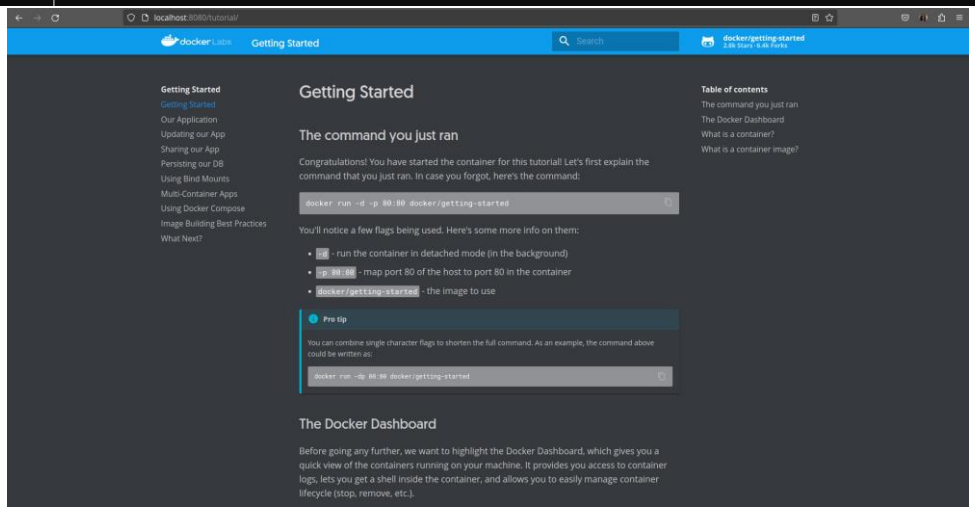
1. Подготовьте рабочее окружение в соответствии с типом вашей операционной системы

- Установите Docker
- Выполните базовую настройку



2. Изучите простейшие консольные команды и возможности Docker Desktop (см. лекцию), создать собственный контейнер docker/getting-started, открыть в браузере и изучить tutorial

```
C:\Users\ac1dl>docker run -dp 8080:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
c158987b0551: Pull complete
1e35f6679fab: Pull complete
cb9626c74200: Pull complete
b6334b6ace34: Pull complete
f1d1c9928c82: Pull complete
9b6f639ec6ea: Pull complete
ee68d3549ec8: Pull complete
33e0cbbb4673: Pull complete
4f7e34c2de10: Pull complete
Digest: sha256:d79336f4812b6547a53e735480dde67f8f8f7071b414fbd9297609ffb989abc1
Status: Downloaded newer image for docker/getting-started:latest
0312cb1bb1490feb21ebfb6cf22631a6dac6eba2852a2bcaafa27aba5030f092

C:\Users\ac1dl>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                    NAMES
0312cb1bb149   docker/getting-started   "/docker-entrypoint.…"   9 seconds ago   Up 7 seconds   0.0.0.0:8080->80/tcp   hopeful_wright
```



3. Создайте docker image, который запускает скрипт с использованием функций из https://github.com/smartiqaorg/geometric_lib.

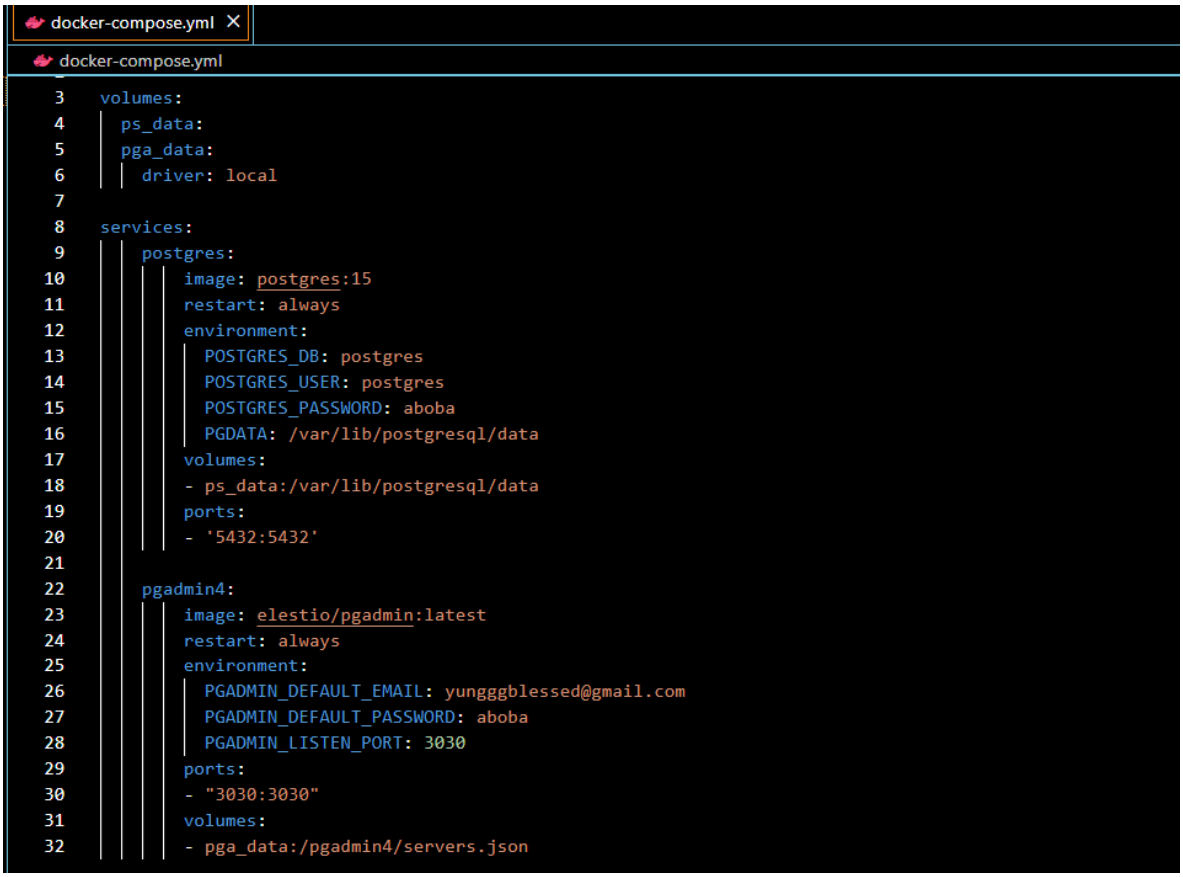
- Данные необходимые для работы скрипта передайте любым удобным способом (например: конфиг файл через docker volume, переменные окружения, перенаправление ввода). Изучите простейшие консольные команды для работы с docker(см. лекцию). Зарегистрируйтесь на DockerHub и выберите необходимые для проекта образы
- Создать Dockerfile для реализации сборки собственных Docker образов
- Использовать его для создания контейнера. Протестировать использование контейнера

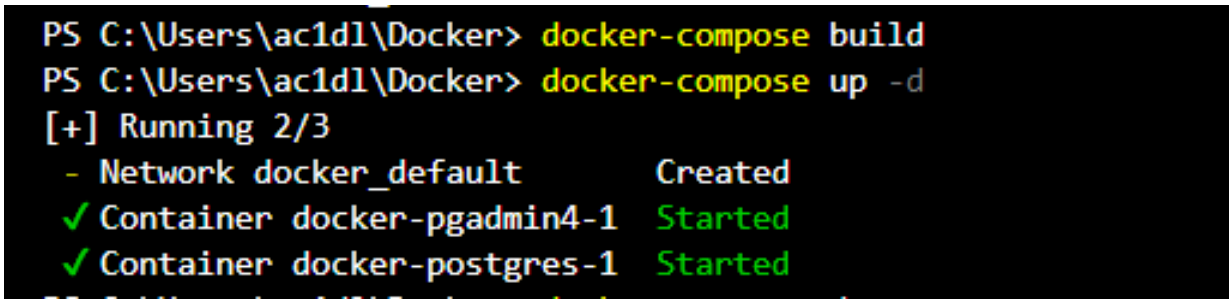
```python
import os
import circle

def main():
    key = 'Radius'
    r = float(os.getenv(key))

    print(f'Perimeter of Circle With Radius={r} Equals {circle.perimeter(r)}')
    print(f'Area of Circle With Radius={r} Equals {circle.area(r)}')

if __name__ == '__main__':
    main()
```

```
PS C:\Users\ac1dl\geometric_lib> docker build -t script .
2024/03/18 18:01:27 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 0.9s (8/8) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                         0.0s
 => => transferring dockerfile: 121B                                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:latest                                                             0.7s
 => [internal] load .dockerignore                                                                                            0.0s
 => => transferring context: 2B                                                                                              0.0s
 => [1/3] FROM docker.io/library/python:latest@sha256:336461f63f4eb1100e178d5acbfea3d1a5b2a53dea88aa0f9b8482d4d02e981c        0.0s
 => [internal] load build context                                                                                            0.0s
 => => transferring context: 2.22kB                                                                                          0.0s
 => CACHED [2/3] WORKDIR /geometric_lib                                                                                      0.0s
 => [3/3] COPY . .                                                                                                           0.0s
 => exporting to image                                                                                                       0.0s
 => => exporting layers                                                                                                      0.0s
 => => writing image sha256:82b224c2fbad83062b157586d5094f6ba22a82b1973a7ff4474a4412d82c9078                                 0.0s
 => => naming to docker.io/library/script                                                                                    0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ac1dl\geometric_lib> docker run -it script
Perimeter of Circle With Radius=5.0 Equals 31.41592653589793
Area of Circle With Radius=5.0 Equals 78.53981633974483
```

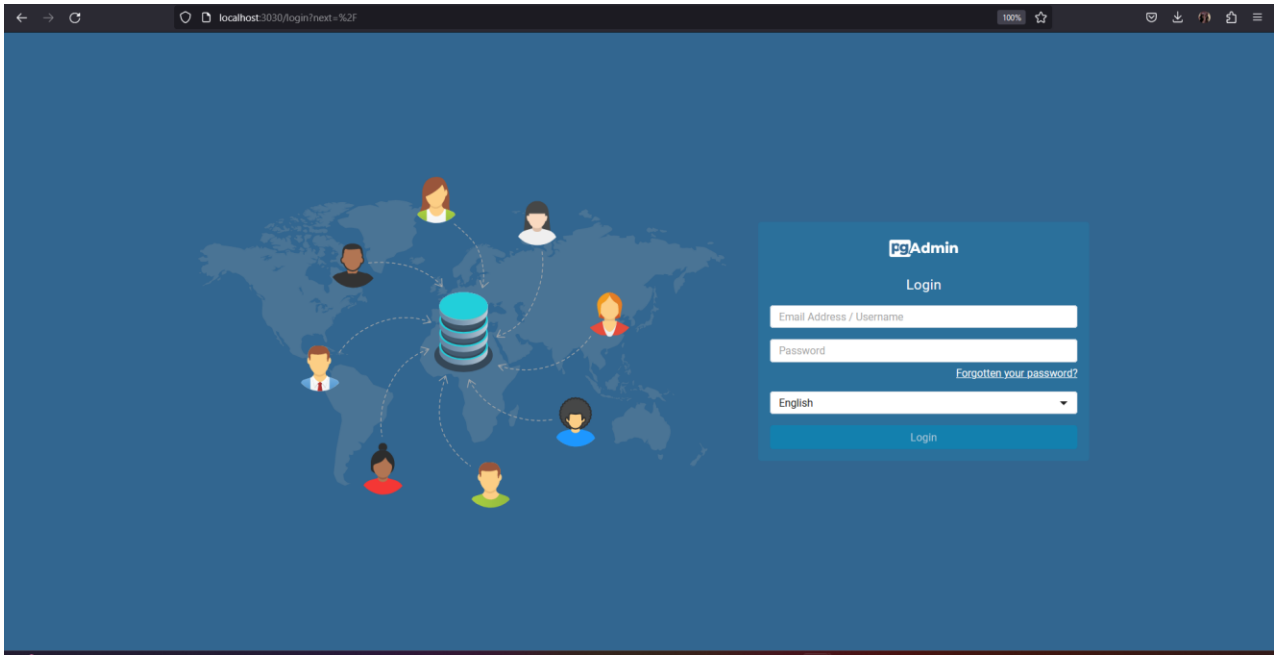
```
FROM python
WORKDIR /geometric_lib
COPY . .
ENV Radius=5
CMD ["python", "script.py"]
```

4. Скачать любой доступный проект с GitHub с произвольным стеком технологий (пример – см. индивидуальное задание) или использовать свой, ранее разработанный. Создать для него необходимый контейнер, используя Docker Compose для управления многоконтейнерными приложениями. Запустить проект в контейнере.(Примеры Images: https://hub.docker.com/_/phpmyadmin, https://hub.docker.com/_/mysql, https://hub.docker.com/_/postgres)





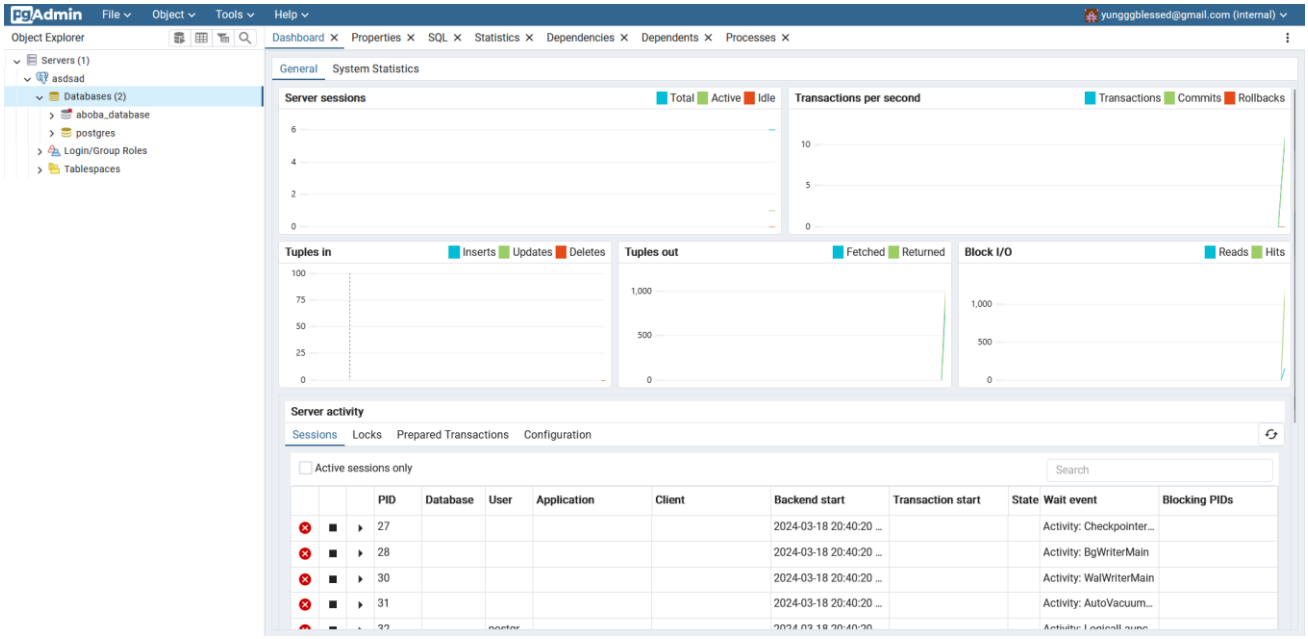
x

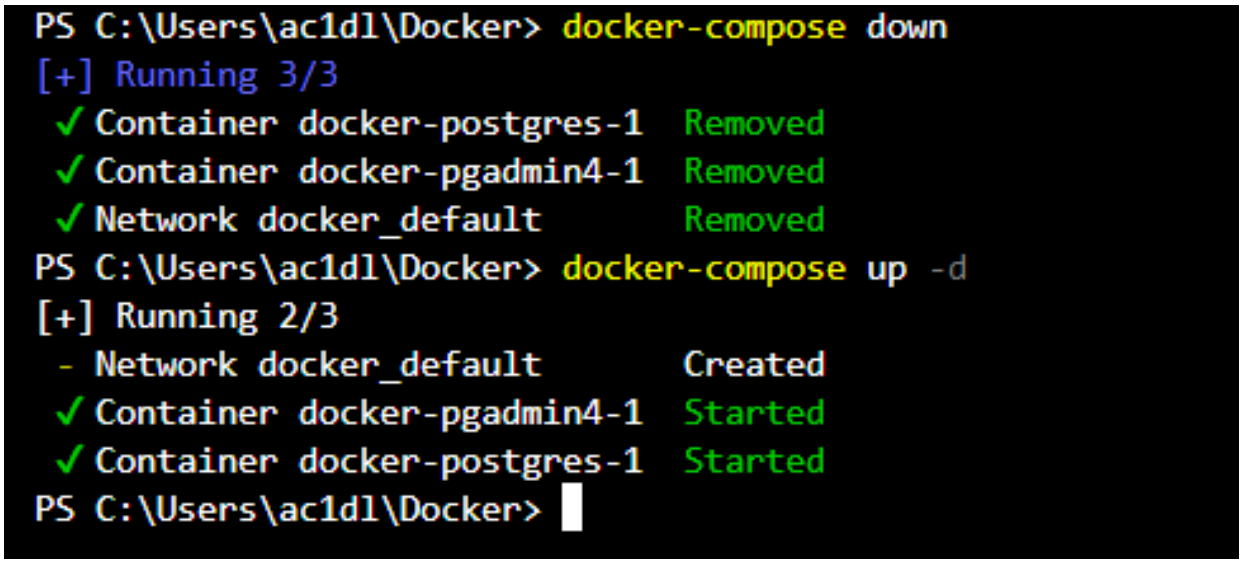


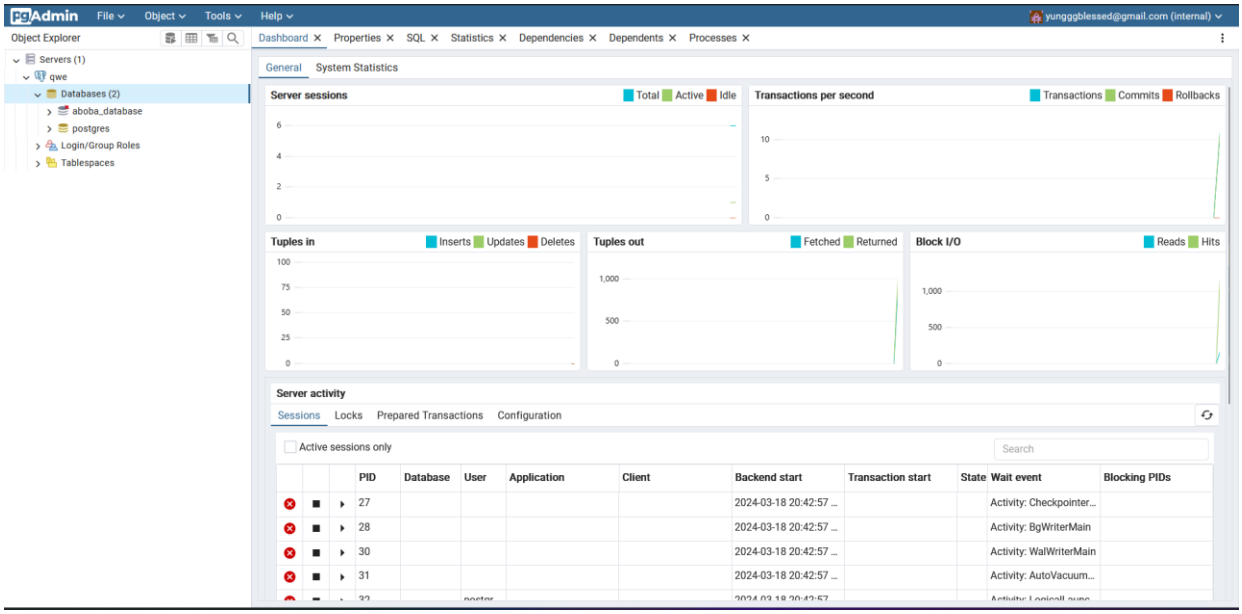
5. Настроить сети и тома для обеспечения связи между контейнерами и сохранения данных (исходные данные, логин, пароль и т.д.)

docker-compose.yml

```yaml
volumes:
  ps_data:
  pga_data:
    driver: local

services:
    postgres:
        image: postgres:15
        restart: always
        environment:
          POSTGRES_DB: postgres
          POSTGRES_USER: postgres
          POSTGRES_PASSWORD: aboba
          PGDATA: /var/lib/postgresql/data
        volumes:
        - ps_data:/var/lib/postgresql/data
        ports:
        - '5432:5432'

    pgadmin4:
        image: elestio/pgadmin:latest
        restart: always
        environment:
          PGADMIN_DEFAULT_EMAIL: yungggblessed@gmail.com
          PGADMIN_DEFAULT_PASSWORD: aboba
          PGADMIN_LISTEN_PORT: 3030
        ports:
        - "3030:3030"
        volumes:
        - pga_data:/pgadmin4/servers.json
```

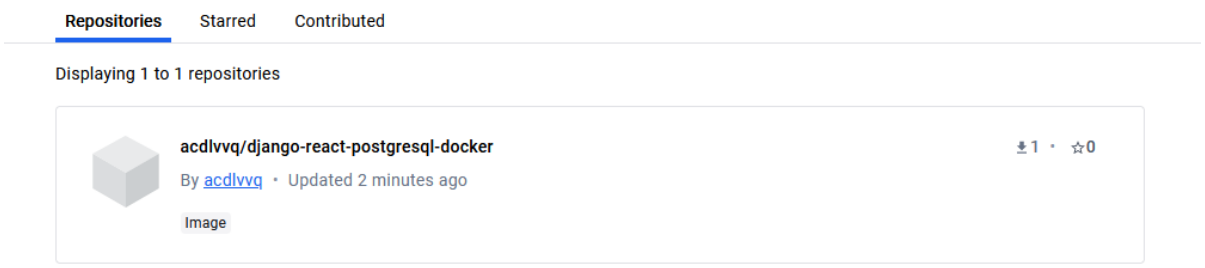






6. Разместите результат в созданный репозиторий в DockerHub

```
PS C:\Users\ac1dl\Docker> docker build -t acdlvvq/django-react-postgresql-docker .
[+] Building 2.0s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 177B
 => [internal] load metadata for docker.io/library/python:latest
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:latest@sha256:336461f63f4eb1100e178d5acbfea3d1a5b2a53dea88aa0f9b8482d4d02e981c
 => [internal] load build context
 => => transferring context: 857.18kB
 => CACHED [2/5] WORKDIR /sm
 => CACHED [3/5] COPY . .
 => CACHED [4/5] RUN pip install -r requirements.txt
 => CACHED [5/5] RUN python ./dj/manage.py makemigrations
 => exporting to image
 => => exporting layers
 => => writing image sha256:cddf1b98ce2829099bd8d399ce74f052b153617961c60a0c805b73607a36a930
 => => naming to docker.io/acdlvvq/django-react-postgresql-docker

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ac1dl\Docker> docker push acdlvvq/django-react-postgresql-docker
Using default tag: latest
The push refers to repository [docker.io/acdlvvq/django-react-postgresql-docker]
c35d09e2273d: Layer already exists
4aa3aa040fbf: Layer already exists
fe79ee76232f: Layer already exists
39bf73c675d3: Layer already exists
9adbc4b1428d: Layer already exists
f52093e4f67d: Layer already exists
1193f41e6b14: Layer already exists
e077e19b6682: Layer already exists
21e1c4948146: Layer already exists
68866beb2ed2: Layer already exists
e6e2ab10dba6: Layer already exists
0238a1790324: Layer already exists
latest: digest: sha256:70cc1eb7d7b5c12b4e658ee0caa240b1b939ded48c3b4ab05ea1a9baae34bfbb size: 2847
```



7. Выполните следующие действия с целью изучить особенности сетевого взаимодействия:

- Получить информацию о всех сетях, работающих на текущем хосте и подробности о каждом типе сети
- Создать свою собственную сеть bridge, проверить, создана ли она, запустить Docker-контейнер в созданной сети, вывести о ней всю информацию(включая IP-адрес контейнера), отключить сеть от контейнера
- Создать еще одну сеть bridge, вывести о ней всю информацию, запустить в ней три контейнера, подключиться к любому из контейнеров и пропинговать два других из оболочки контейнера, убедиться, что между контейнерами происходит общение по IP-адресу
- Создать свою собственную сеть overlay, проверить, создана ли она, вывести о ней всю информацию
- Создать еще одну сеть overlay, проверить, создана ли она, вывести о ней всю информацию, удалить сеть
- Попробовать создать сеть host, сохранить результат в отчет.

```
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
b5d09553b0aa   bridge    bridge    local
d4262570137c   host      host      local
8d3a7e1089e2   none      null      local
PS C:\Users\ac1dl\Docker> docker network inspect d4262570137c
[
    {
        "Name": "host",
        "Id": "d4262570137c8ce9e2cf4efb003fa02442da4ead210b98d0f9e3c6cc026f05a7",
        "Created": "2024-03-18T14:31:21.207910929Z",
        "Scope": "local",
        "Driver": "host",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": null
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

```
PS C:\Users\ac1dl\Docker> docker network create -d bridge myntwrk
a42ef7de4b6319216436988a11f6b8b27ce44a14976e8f6ddb92aa5c478a7646
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
b5d09553b0aa   bridge    bridge    local
d4262570137c   host      host      local
a42ef7de4b63   myntwrk   bridge    local
8d3a7e1089e2   none      null      local
PS C:\Users\ac1dl\Docker> docker run -dp 8081:8081 -e "port=8081" --name rndm --network=myntwrk docker/getting-started
d60cc0fbd45a698484b6ed8af6b694ab581a276051df2fc281778d003f04b0cc
```

```
PS C:\Users\ac1dl\Docker> docker inspect rndm
```

```
        "MacAddress": "",
        "Networks": {
            "myntwrk": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": [
                    "d60cc0fbd45a"
                ],
                "MacAddress": "02:42:ac:1e:00:02",
                "NetworkID": "a42ef7de4b6319216436988a11f6b8b27ce44a14976e8f6ddb92aa5c478a7646",
                "EndpointID": "3400e0cf764faba688f34f46e54959a6fbe0ed6c350ab960fdec33679f4764a8",
                "Gateway": "172.30.0.1",
                "IPAddress": "172.30.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DriverOpts": null,
                "DNSNames": [
                    "rndm",
                    "d60cc0fbd45a"
                ]
            }
        }
```

```
PS C:\Users\ac1dl\Docker> docker network disconnect myntwrk rndm
PS C:\Users\ac1dl\Docker> docker inspect rndm
```

```
PS C:\Users\ac1dl\Docker> docker network create -d bridge scntwrk
de71153423fb0460270f8b5eeb773987e306ca9a402388b0ba4113bf9cae2099
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
b5d09553b0aa   bridge    bridge    local
d4262570137c   host      host      local
a42ef7de4b63   myntwrk   bridge    local
8d3a7e1089e2   none      null      local
de71153423fb   scntwrk   bridge    local
PS C:\Users\ac1dl\Docker> docker run -dp 8090:8090 -e "port=8090" --name fr --network=scntwrk docker/welcome-to-docker
Unable to find image 'docker/welcome-to-docker:latest' locally
latest: Pulling from docker/welcome-to-docker
96526aa774ef: Pull complete
740091335c74: Pull complete
da9c2e764c5b: Pull complete
ade17ad21ef4: Pull complete
4e6f462c8a69: Pull complete
1324d9977cd2: Pull complete
1b9b96da2c74: Pull complete
5d329b1e101a: Pull complete
Digest: sha256:eedaff45e3c78538087bdd9dc7afafac7e110061bbdd836af4104b10f10ab693
Status: Downloaded newer image for docker/welcome-to-docker:latest
2646b5382b45cf39a35fa765293f3d414ea5a4bf0889c19e4534a86712412290
```

```
PS C:\Users\ac1dl\Docker> docker run -dp 8070:8070 -e "port=8070" --name sc --network=scntwrk golang
Unable to find image 'golang:latest' locally
latest: Pulling from library/golang
71215d55680c: Already exists
3cb8f9c23302: Already exists
5f899db30843: Already exists
c29f45468664: Pull complete
05ed43b991ed: Pull complete
0efeec9a2d29: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:0b55ab82ac2a54a6f8f85ec8b943b9e470c39e32c109b766bbc1b801f3fa8d3b
Status: Downloaded newer image for golang:latest
d3b246bd0106bb9f772439760d62346d5e336cf58071b7a3a392b0b88030022e
```

```
PS C:\Users\ac1dl\Docker> docker run -it -p 8060:8060 -e "port=8060" --name thr --network=scntwrk archlinux
Unable to find image 'archlinux:latest' locally
latest: Pulling from library/archlinux
9a82a64c3a84: Pull complete
403a73115b61: Pull complete
Digest: sha256:2dbd72d1e5510e047db7f441bf9069e9c53391b87e04e5bee3f379cd03cec060
Status: Downloaded newer image for archlinux:latest
       [root@8e4d5d592112 /]# ping fr
PING fr (192.168.0.2) 56(84) bytes of data.
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=1 ttl=64 time=0.070 ms
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=3 ttl=64 time=0.076 ms
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=4 ttl=64 time=0.053 ms
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=5 ttl=64 time=0.052 ms
64 bytes from fr.scntwrk (192.168.0.2): icmp_seq=6 ttl=64 time=0.057 ms
```

```
PS C:\Users\ac1dl\Docker> docker swarm init
Swarm initialized: current node (90lrwa5i9ff5aviz5i0ufbrjf) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-0qaspgjvnheixhskmm68h9jxvskvzv2kmv4m3sss9lt4zm1p42-1b5bph4rz9rteptwf4eoh0lof 192.168.65.3:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

PS C:\Users\ac1dl\Docker> docker network create -d overlay ovrly
xjnanvn5inil196hgprzhj5op
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
b5d09553b0aa   bridge            bridge    local
6349b887c624   docker_gwbridge   bridge    local
d4262570137c   host              host      local
q7wmdvlm5crb   ingress           overlay   swarm
a42ef7de4b63   myntwrk           bridge    local
8d3a7e1089e2   none              null      local
xjnanvn5inil   ovrly             overlay   swarm
de71153423fb   scntwrk           bridge    local
```

```
PS C:\Users\ac1dl\Docker> docker network create -d overlay scovrly
hxota7isas459p40pdus9qlx2
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
b5d09553b0aa   bridge            bridge    local
6349b887c624   docker_gwbridge   bridge    local
d4262570137c   host              host      local
q7wmdvlm5crb   ingress           overlay   swarm
a42ef7de4b63   myntwrk           bridge    local
8d3a7e1089e2   none              null      local
xjnanvn5inil   ovrly             overlay   swarm
de71153423fb   scntwrk           bridge    local
hxota7isas45   scovrly           overlay   swarm
```

```
PS C:\Users\ac1dl\Docker> docker network rm -f scovrly
scovrly
PS C:\Users\ac1dl\Docker> docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
b5d09553b0aa   bridge            bridge    local
6349b887c624   docker_gwbridge   bridge    local
d4262570137c   host              host      local
q7wmdvlm5crb   ingress           overlay   swarm
a42ef7de4b63   myntwrk           bridge    local
8d3a7e1089e2   none              null      local
xjnanvn5inil   ovrly             overlay   swarm
de71153423fb   scntwrk           bridge    local
```

```
PS C:\Users\ac1dl\Docker> docker network create -d host hst
Error response from daemon: only one instance of "host" network is allowed
```

Вывод

В ходе выполнения данной лабораторной работы я познакомился с возможностями системы контейнизации Docker, изучил основные команды для работы с контейнерами и образами, получил практические навыки, работая с локальными Docker-образами и контейнерами. Я также научился создавать и управлять Docker-сетями, что позволило мне лучше понять, как контейнеры взаимодействуют друг с другом.