

## Задание:

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

- Установите Docker
- Выполните базовую настройку

```
❯ docker --version
Docker version 25.0.3, build 4debf41
```

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

```
Common Commands:
run      Create and run a new container from an image
exec     Execute a command in a running container
ps       List containers
build    Build an image from a Dockerfile
pull     Download an image from a registry
push     Upload an image to a registry
images   List images
login    Log in to a registry
logout   Log out from a registry
search   Search Docker Hub for images
version  Show the Docker version information
info     Display system-wide information
```

```
❯ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS   NAMES
d82325a3f377   nginx:1.23    "/docker-entrypoint..." 20 hours ago   Exited (0)    20 hours ago   determined_cerf
4a27e33f1038   kcoursedocker/task-2.9 "python3 input.py"      20 hours ago   Exited (0)    20 hours ago   wonderful_kalam
131b505d70b8   kcoursedocker/task-2.9 "python3 input.py"      20 hours ago   Exited (1)    20 hours ago   romantic_mendel
f99c4e7ff7a0   postgres     "docker-entrypoint.s..." 21 hours ago   Exited (1)    21 hours ago   mystifying_kare
```

```
❯ docker ps -q
d82325a3f377
4a27e33f1038
131b505d70b8
f99c4e7ff7a0
```

```
❯ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
custom_ubuntu       3           90c770a91c50 3 hours ago    288MB
custom_ubuntu       2           d7d88f0d1cbd 3 hours ago    72.8MB
custom_ubuntu       1           f54d9e421b08 3 hours ago    72.8MB
new_ubuntu          1           4f692c315c77 22 hours ago   77.9MB
postgres            latest      eb634efa7ee4 13 days ago    431MB
node                latest      1b9d5f3b36bf 2 weeks ago    1.1GB
nginx               latest      e4720093a3c1 2 weeks ago    187MB
ubuntu              latest      3db8720ecbf5 3 weeks ago    77.9MB
python              latest      a3aef63c6c10 3 weeks ago    1.02GB
golang              latest      0c4ed86491a4 3 weeks ago    823MB
redis               latest      d1397258b209 7 weeks ago    138MB
node                16         1ddc7e4055fd 5 months ago   909MB
nginx               1.23       a7be6198544f 9 months ago   142MB
hello-world         latest      d2c94e258dcb 10 months ago  13.3kB
kcoursedocker/task-2.9 latest      6975591fdaef 19 months ago  48.7MB
yandex/clickhouse-server latest      c739327b5607 2 years ago    826MB
```

```
❏ > ❏ ~
❏ docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
custom_ubuntu        3            90c770a91c50  3 hours ago  288MB
custom_ubuntu        2            d7d88f0d1cbd  3 hours ago  72.8MB
custom_ubuntu        1            f54d9e421b08  3 hours ago  72.8MB
new_ubuntu           1            4f692c315c77  22 hours ago 77.9MB
postgres             latest       eb634efa7ee4  13 days ago  431MB
node                  latest       1b9d5f3b36bf  2 weeks ago  1.1GB
nginx                 latest       e4720093a3c1  2 weeks ago  187MB
ubuntu                latest       3db8720ecbf5  3 weeks ago  77.9MB
python                latest       a3aef63c6c10  3 weeks ago  1.02GB
golang                latest       0c4ed86491a4  3 weeks ago  823MB
redis                 latest       d1397258b209  7 weeks ago  138MB
node                  16          1ddc7e4055fd  5 months ago 909MB
nginx                 1.23        a7be6198544f  9 months ago 142MB
hello-world           latest       d2c94e258dcb  10 months ago 13.3kB
kcoursedocker/task-2.9 latest       6975591fdaef  19 months ago 48.7MB
yandex/clickhouse-server latest       c739327b5607  2 years ago  826MB
```

```
❏ > ❏ ~
❏ docker history 90c770a91c50
IMAGE      CREATED      CREATED BY          SIZE      COMMENT
90c770a91c50 3 hours ago  RUN /bin/sh -c echo "Hello" # buildkit  0B        buildkit.dockerfile.v0
<missing>    3 hours ago  RUN /bin/sh -c apt-get update && apt-get ins... 215MB     buildkit.dockerfile.v0
<missing>    3 hours ago  COPY script.sh script.sh # buildkit      36B       buildkit.dockerfile.v0
<missing>    3 hours ago  WORKDIR /home/docker                      0B        buildkit.dockerfile.v0
<missing>    6 weeks ago  /bin/sh -c #(nop)  CMD ["/bin/bash"]      0B
<missing>    6 weeks ago  /bin/sh -c #(nop)  ADD file:4b4e122c96445546e... 72.8MB
<missing>    6 weeks ago  /bin/sh -c #(nop)  LABEL org.opencontainers. .... 0B
<missing>    6 weeks ago  /bin/sh -c #(nop)  LABEL org.opencontainers. .... 0B
<missing>    6 weeks ago  /bin/sh -c #(nop)  ARG LAUNCHPAD_BUILD_ARCH    0B
<missing>    6 weeks ago  /bin/sh -c #(nop)  ARG RELEASE                0B
```

```
❏ > ❏ ~
❏ docker run --rm -it custom_ubuntu:3
root@de293e17f7e8:/home/docker#
root@de293e17f7e8:/home/docker# exit
exit

❏ > ❏ ~
❏ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
d82325a3f377   nginx:1.23     "/docker-entrypoint. ...." 20 hours ago   Exited (0)    20 hours ago   determined_cerf
4a27e33f1038   kcoursedocker/task-2.9 "python3 input.py"      20 hours ago   Exited (0)    20 hours ago   wonderful_kalam
131b505d70b8   kcoursedocker/task-2.9 "python3 input.py"      21 hours ago   Exited (1)    20 hours ago   romantic_mendel
f99c4e7ff7a0   postgres      "docker-entrypoint.s ..." 21 hours ago   Exited (1)    21 hours ago   mystifying_kare
```

```
❏ > ❏ ~
❏ docker run -d nginx:1.23
d5580e2a30a823f30e9d852ea2ebe19a08cc24d862a66dd1be7143221b7a8d2d

❏ > ❏ ~
❏ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
d5580e2a30a8   nginx:1.23     "/docker-entrypoint. ...." 5 seconds ago   Up 4 seconds   80/tcp         festive_heyrovsky

❏ > ❏ ~
❏ docker exec -it festive_heyrovsky bash
root@d5580e2a30a8:/#
```

```
❏ > ❏ ~
❏ docker run -d nginx:1.23
d5580e2a30a823f30e9d852ea2ebe19a08cc24d862a66dd1be7143221b7a8d2d

❏ > ❏ ~
❏ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
d5580e2a30a8   nginx:1.23     "/docker-entrypoint. ...." 5 seconds ago   Up 4 seconds   80/tcp         festive_heyrovsky

❏ > ❏ ~
❏ docker exec -it festive_heyrovsky bash
root@d5580e2a30a8:/# exit
exit

❏ > ❏ ~
❏ docker stop $(docker ps -q)
d5580e2a30a8
```

The image shows a terminal window at the top with the following output:

```
docker run -d -p 80:80 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
1bf30fb56aa9e54f4bb251a4110d5851cb645a39581bdce2642ea6eef286531a
```

Below the terminal is a web browser window showing the Docker Labs 'Getting Started' page. The page has a sidebar with links: Getting Started, Getting Started, Our Application, Updating our App, Sharing our App, Persisting our DB, Using Bind Mounts, Multi-Container Apps, Using Docker Compose, Image Building Best Practices, and What Next?. The main content area is titled 'Getting Started' and includes a section 'The command you just ran' with the command `docker run -d -p 80:80 docker/getting-started`. It also has a 'Pro tip' box explaining how to shorten the command using single character flags. The page ends with a section 'The Docker Dashboard'.

3. Создайте docker image, который запускает скрипт с использованием функций из [https://github.com/smartikaorg/geometric\\_lib](https://github.com/smartikaorg/geometric_lib).
  - a. 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
  - b. Создать Dockerfile для реализации сборки собственных Docker образов

```
FROM python

WORKDIR /home/geometric-lib

COPY main.py main.py

COPY square.py square.py

COPY circle.py circle.py

ENTRYPOINT ["python"]
CMD ["main.py"]
```

- с. Использовать его для создания контейнера. Протестировать использование контейнера

перенаправление ввода:

```
~/docker/geometric-lib > on main ↑2 ?2
ls
docs/ circle.py Dockerfile input.txt main.py square.py
```

```
~/docker/geometric-lib > on main ↑2 ?2
docker build -t custom_python:1 .
[+] Building 0.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 251B
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:latest
=> [internal] load build context
=> => transferring context: 262B
=> CACHED [2/5] WORKDIR /home/geometric-lib
=> CACHED [3/5] COPY main.py main.py
=> CACHED [4/5] COPY square.py square.py
=> CACHED [5/5] COPY circle.py circle.py
=> exporting to image
=> => exporting layers
=> => writing image sha256:a0f2f1efc3c26200f001b6895a71cddf5c413a5208c50746667f880512f38ba6
=> => naming to docker.io/library/custom_python:1
```

```
~/docker/geometric-lib > on main ↑2 ?2
cat input.txt | docker run --rm -i custom_python:1 bash
Square:
Perimeter: 20.0
Area: 25.0
Circle:
Perimeter: 62.83185307179586
Area: 314.1592653589793

~/docker/geometric-lib > on main ↑2 ?2
cat input.txt
10
5
```

переменные окружения:

```
F1 > F0 ~/docker/geometric_lib > on F1 F2 main ↑2 !1 ?2
docker run -it --rm -e RADIUS=5 -e SIDE=10 task_image:2
Square:
Perimeter: 40.0
Area: 100.0
aCircle:
Perimeter: 31.41592653589793
Area: 78.53981633974483

F2 > F0 ~/docker/geometric_lib > on F1 F2 main ↑2 !1 ?2
cat main.py
from circle import area as area_circle, perimeter as perimeter_circle
from square import area as area_square, perimeter as perimeter_square
import os

if __name__ == '__main__':
    r = float(os.environ.get("RADIUS"))
    a = float(os.environ.get("SIDE"))

    print(f"Square:\nPerimeter: {perimeter_square(a)}\nArea: {area_square(a)}")
    print(f"Circle:\nPerimeter: {perimeter_circle(r)}\nArea: {area_circle(r)}")
```

ИЛИ

```
F1 > F0 ~/docker/geometric_lib > on F1 F2 main ↑2 !1 ?2
export RADIUS=10 SIDE=5

F2 > F0 ~/docker/geometric_lib > on F1 F2 main ↑2 !1 ?2
docker run -it --rm -e RADIUS=$RADIUS -e SIDE=$SIDE task_image:2
Square:
Perimeter: 20.0
Area: 25.0
aCircle:
Perimeter: 62.83185307179586
Area: 314.1592653589793
```

ИЛИ

```

[7] > [7] ~/docker/geometric_lib > on [1] [2]main ↑2 !1 ?3
docker run --rm --env-file .env -it task_image:2
Square:
Perimeter: 20.0
Area: 25.0
Circle:
Perimeter: 62.83185307179586
Area: 314.1592653589793

[7] > [7] ~/docker/geometric_lib > on [1] [2]main ↑2 !1 ?3
cat .env
RADIUS=10
SIDE=5

```

4. Скачать любой доступный проект с GitHub с произвольным стеком технологий 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/_/phpmyadmin),  
[https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql), [https://hub.docker.com/\\_/postgres](https://hub.docker.com/_/postgres))

```

[7] > [7] ~/docker/compose
tree
.
├── client
│   ├── client.py
│   └── Dockerfile
├── docker-compose.yaml
└── server
    ├── Dockerfile
    ├── index.html
    └── server.py

3 directories, 6 files

```

```

vim Dockerfile

FROM python:latest

ADD client.py /client/

WORKDIR /client/
~
~
~
~
~

```

```
vim client.py

import urllib.request

# Эта переменная содержит запрос к 'http://localhost:1234/'.
# Возможно, сейчас вы задаётесь вопросом о том, что такое 'http://localhost:1234'.
# localhost указывает на то, что программа работает с локальным сервером.
# 1234 - это номер порта, который вам предлагалось запомнить при настройке серверного кода.

fp = urllib.request.urlopen("http://localhost:1234/")

# 'encodedContent' соответствует закодированному ответу сервера ('index.html').
# 'decodedContent' соответствует декодированному ответу сервера (тут будет то, что мы хотим вывести на экран).

encodedContent = fp.read()
decodedContent = encodedContent.decode("utf8")

# Выводим содержимое файла, полученного с сервера ('index.html').

print(decodedContent)

# Закрываем соединение с сервером.

fp.close()
```

```
vim Dockerfile

FROM python:latest

ADD server.py /server/
ADD index.html /server/

WORKDIR /server/
```

```
vim server.py

import http.server
import socketserver

# Эта переменная нужна для обработки запросов клиента к серверу.

handler = http.server.SimpleHTTPRequestHandler

# Тут мы указываем, что сервер мы хотим запустить на порте 1234.
# Постарайтесь запомнить эти сведения, так как они нам очень пригодятся в дальнейшем, при работе с docker-compose.

with socketserver.TCPServer(("", 1234), handler) as httpd:

    # Благодаря этой команде сервер будет выполняться постоянно, ожидая запросов от клиента.

    httpd.serve_forever()
```

```
vim index.html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    Docker-Compose is magic!
</head>
<body>

</body>
</html>
```

```
vim docker-compose.yml

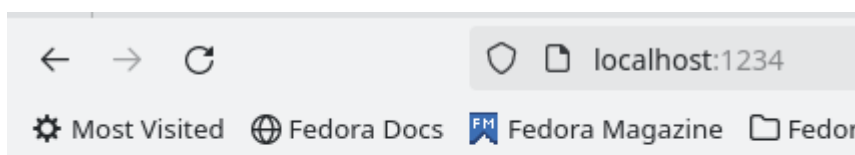
version: '3'

services:

  server:
    build: server/
    command: python ./server.py
    ports:
      - 1234:1234

  client:
    build: client/
    command: python ./client.py
    network_mode: host
    depends_on:
      server:
        condition: service_started
```

```
~/docker/compose
docker compose up -d
[+] Running 2/3
  .: Network compose_default      Created
  ✓ Container compose-server-1    Started
  ✓ Container compose-client-1    Started
```



Docker-Compose is magic!

```
~/docker/compose
docker compose ls
NAME                STATUS              CONFIG FILES
compose             running(1)          /home/ksu_kotovaa/docker/compose/docker-compose.yml
```

```
~/docker/compose
docker compose down
[+] Running 3/3
  ✓ Container compose-client-1    Removed
  ✓ Container compose-server-1    Removed
  ✓ Network compose_default      Removed
```



на 3 контейнера:  
Dockerfile database

```
vim Dockerfile

FROM postgres:14
COPY ./init_db.sh /docker-entrypoint-initdb.d/init_db.sh
~
~
~
~
```

Dockerfile backend

```
vim Dockerfile

FROM python:3.8

COPY ./requirements.txt ./requirements.txt
RUN python -m pip install --upgrade pip && pip install -r requirements.txt

WORKDIR /app

EXPOSE 8000

COPY ./app.py ./app.py
COPY ./wsgi.py ./wsgi.py

CMD ["gunicorn", "--bind", "0.0.0.0:8000", "wsgi:app"]
```

Dockerfile frontend

```
vim Dockerfile

FROM node:17 AS BUILD

WORKDIR /app

COPY ./todo-list/package.json ./package.json

RUN npm i

COPY ./todo-list ./
RUN npm run build

FROM nginx

COPY --from=BUILD /app/dist/index.html /nginx/static/index.html
COPY --from=BUILD /app/dist/static/css /nginx/static
COPY --from=BUILD /app/dist/static/js /nginx/static
```

Images:

```
❏ > ❏ ~/Downloads/Docker_course-main/lesson_7/frontend
docker images
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
7_nginx              latest       ed247bd8a0b1     3 hours ago      187MB
7_database           latest       84c43553ee08     3 hours ago      422MB
7_back               latest       2add46136dc5     3 hours ago      1.05GB
postgres_client     latest       3118e0d1f5ad     10 hours ago     1.02GB
```

docker-compose.yml

```
vim docker-compose.yml
version: '3'
services:
  db:
    image: 7_database
    container_name: database
    networks:
      - todo_net
    environment:
      POSTGRES_DB: docker_app_db
      POSTGRES_USER: docker_app
      POSTGRES_PASSWORD: docker_app
    volumes:
      - todo_db:/var/lib/postgresql/data
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "pg_isready", "-U", "docker_app"]
      interval: 5s
      timeout: 5s
      retries: 3
```

```
backend:
  image: 7_back
  container_name: backend
  networks:
    - todo_net
  environment:
    HOST: database
    PORT: 5432
    DB: docker_app_db
    DB_USERNAME: docker_app
    DB_PASSWORD: docker_app
  healthcheck:
    test: ["CMD", "curl", "--fail", "localhost:8000/test"]
    interval: 5s
    timeout: 5s
    retries: 3
  depends_on:
    db:
      condition: service_healthy
```

```
nginx:
  image: 7_nginx
  container_name: frontend
  networks:
    - todo_net
  ports:
    - 80:80
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
  depends_on:
    db:
      condition: service_healthy
    backend:
      condition: service_healthy

volumes:
  todo_db:
    name: todo_db

networks:
  todo_net:
    name: todo_net
```

Start:

```
~/Downloads/Docker_course-main/lesson_7
$ docker compose up -d
[+] Running 3/5
  Network todo_net      Created                                11.8s
  Volume "todo_db"      Created                                11.7s
  Container database     Healthy                               5.8s
  Container backend      Healthy                               11.4s
  Container frontend     Started                               11.6s

~/Downloads/Docker_course-main/lesson_7
$ docker compose ps
NAME                IMAGE             COMMAND                  SERVICE    CREATED         STATUS               PORTS
backend             7_back           "gunicorn --bind 0.0..." backend    About a minute ago Up About a minute (healthy) 8000/tcp
database            7_database       "docker-entrypoint.s..." db         About a minute ago Up About a minute (healthy) 5432/tcp
frontend            7_nginx         "/docker-entrypoint..." nginx      About a minute ago Up About a minute      0.0.0.0:80->80/tcp, :::80->80/tcp
```

Result:

localhost/tutorial/ 🔍 ⭐

Fedora Docs Fedora Magazine Fedora Project User Communities Red Hat Free Content

ToDo Лист

▶ Новая задача

Нет задач

Всего задач: 0 Все Активные Выполненные

```

[7] > [7] ~/Downloads/Docker_course-main/lesson_7
docker compose down -v
[+] Running 5/5
✓ Container frontend    Removed
✓ Container backend     Removed
✓ Container database    Removed
✓ Volume todo_db        Removed
✓ Network todo_net      Removed

```

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

```

[7] > [7] ~/Downloads/Docker_course-main/lesson_7/frontend
docker volume create todo_vol
todo_vol

[7] > [7] ~/Downloads/Docker_course-main/lesson_7/frontend
docker network create todo_net
bf766fb0ce18c21be0582839ffbd16fd64902f844dbdcaf6e4ce64261d62ef7d

```

```

[7] > [7] ~/Downloads/Docker_course-main/lesson_7
docker run --rm -d \
--name database \
--net=todo_net \
-v todo_db:/var/lib/postgresql/data \
-e POSTGRES_DB=docker_app_db \
-e POSTGRES_USER=docker_app \
-e POSTGRES_PASSWORD=docker_app \
7_database
bd20719ab77f0d082fc5578809ed05a2519c68b10e3bf879fc2129486a08dddc

```

```

[7] > [7] ~/Downloads/Docker_course-main/lesson_7
docker run --rm -d \
--name backend \
--net=todo_net \
-e HOST=database \
-e PORT=5432 \
-e DB=docker_app_db \
-e DB_USERNAME=docker_app \
-e DB_PASSWORD=docker_app \
7_back
5556447cbcc8b4b7a48929291695e183b651ed234afd01221b482f0da0a995c3

```

```
F1 > F2 ~/Downloads/Docker_course-main/lesson_7
docker run --rm -d \
--name frontend \
--net=todo_net \
-p 80:80 \
-v $(pwd)/nginx/nginx.conf:/etc/nginx/nginx.conf:ro \
7_nginx
1221bf8c125122e7721a8fdb66eaf149667a4f7f0afed2be62deef8cdfb5d939
```

localhost/tutorial/

Most Visited Fedora Docs Fedora Magazine Fedora Project User Communities Red Hat Free Content

## ToDo Лист

Нет задач

Всего задач: 0

Все

Активные

Выполненные

6. Разместите результат в созданный репозиторий в DockerHub

```
F1 > F2 ~
docker login
Log in with your Docker ID or
//hub.docker.com/ to create or
You can log in with your passw
r organizations using SSO. Lea

Username: ksukot
Password:
WARNING! Your password will be
Configure a credential helper
https://docs.docker.com/engine







Login Succeeded
```

```
❯ docker tag 7_database:latest ksukot/docker-lab:database
❯ docker push ksukot/docker-lab:database
The push refers to repository [docker.io/ksukot/docker-lab]
```

```
❯ docker tag 7_back:latest ksukot/docker-lab:backend
❯ docker push ksukot/docker-lab:backend
The push refers to repository [docker.io/ksukot/docker-lab]
```

```
❯ docker tag 7_nginx:latest ksukot/docker-lab:frontend
❯ docker push ksukot/docker-lab:frontend
The push refers to repository [docker.io/ksukot/docker-lab]
41bc924d4620: Preparing
```

This repository contains 3 tag(s).

Tag	OS	Type	Pulled	Pushed
 frontend		Image	---	3 minutes ago
 backend		Image	---	4 minutes ago
 database		Image	---	5 minutes ago

[See all](#)

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

```
root@bf2ef3c1c8ac:~/docker# docker network inspect $(docker network ls -q)
[
  {
    "Name": "back_net",
    "Id": "a94969e55272172586a608c074023ee67c3ad9bbfaa70f9bae6c4b984c3eabe2",
    "Created": "2024-03-09T10:29:02.408470917+03:00",
    "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,
```

- 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

```
root@bf2ef3c1c8ac:~/docker# docker network create my_bridge
d46352482d819e269ddf05fd804f21cc1ade4425b6fea79fe98566cde4868df1
```

```
root@bf2ef3c1c8ac:~/docker# docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
a94969e55272	back_net	bridge	local
07a3856ad7c6	bridge	bridge	local
c3c57f637b79	host	host	local
d46352482d81	my_bridge	bridge	local
5feb3e46dff9	none	null	local
aa74e022652f	task_6.4_net	bridge	local
cbb934a8865d	task_7.1_net	bridge	local

```
root@bf2ef3c1c8ac:~/docker# docker run -it --rm --name my_ubuntu --net my_bridge custom_ubuntu:0
→ docker
```

```
❏ > ❏ ~/docker
docker network inspect my_bridge
[
  {
    "Name": "my_bridge",
    "Id": "d46352482d819e269ddf05fd804f21cc1ade4425b6fea79fe98566cde4868df1",
    "Created": "2024-03-09T21:36:47.905531412+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.26.0.0/16",
          "Gateway": "172.26.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
  },
]
```

```
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "bf2ef3c1c8ac487d50d4c3bfea15bd5673d17051d7d9f4353ff03f30c76767c8": {
        "Name": "my_ubuntu",
        "EndpointID": "3be68b7ce96ec0efb3ecf85568cb332c8561a4a1180c8db84d1c0ba497d622ff",
        "MacAddress": "02:42:ac:1a:00:02",
        "IPv4Address": "172.26.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

```
❏ > ❏ ~/docker
docker network disconnect my_bridge my_ubuntu
```

```
❏ > ❏ ~/docker
docker network inspect my_bridge | grep Containers
"Containers": {},
```

- 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



```
ksu_kotovaa@fedora:~/docker

[ ] [ ] > [ ] ~/docker
- docker network create my_bridge2
80ed28a31ca5cfef6fab10c753cd04fa95cf5a3017c563a09452bd374c770e49

[ ] [ ] > [ ] ~/docker
- docker network inspect my_bridge2
[
  {
    "Name": "my_bridge2",
    "Id": "80ed28a31ca5cfef6fab10c753cd04fa95cf5a3017c563a09452bd374c770e49",
    "Created": "2024-03-10T16:07:50.689507061+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.0.0/20",
          "Gateway": "192.168.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
```

```
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```

```
[ ] [ ] > [ ] ~/docker
- ifconfig
br-80ed28a31ca5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 255.255.240.0 broadcast 192.168.15.255
    inet6 fe80::42:ffff:fee6:8805 prefixlen 64 scopeid 0x20<link>
    ether 02:42:ff:e6:88:05 txqueuelen 0 (Ethernet)
    RX packets 43 bytes 1360 (1.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 85 bytes 9107 (8.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[ ] [ ] > [ ] ~/docker
- docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED        STATUS        PORTS          NAMES
db61abc3b18c   custom_ubuntu:0 "zsh"                  About a minute ago Up About a minute          ubuntu_3
a156b10947ef   custom_ubuntu:0 "zsh"                  About a minute ago Up About a minute          ubuntu_2
507fddaf0ee5   custom_ubuntu:0 "zsh"                  About a minute ago Up About a minute          ubuntu_1
```

```

[71] > [72] ~/docker
[73] docker network inspect my_bridge2 | grep -A 30 Containers
"Containers": {
  "4cee1cf424dce18bc44515893b5d2f6cb8d853fe929e87a2927beecf46418f32": {
    "Name": "ubuntu_2",
    "EndpointID": "4f341abd8e48491006264f159d17f7354771e560d9de5034963a851c790a6fb6",
    "MacAddress": "02:42:c0:a8:00:03",
    "IPv4Address": "192.168.0.3/20",
    "IPv6Address": ""
  },
  "5066cdb9b25e8699d2f6fb75f2e0521c4e88fe8688e5bd63e79528f0a23387c78": {
    "Name": "ubuntu_3",
    "EndpointID": "c50023be50af2d796e65d461b026ced3d68e65c77fcbf9a590c84b2e4ce01a0e",
    "MacAddress": "02:42:c0:a8:00:04",
    "IPv4Address": "192.168.0.4/20",
    "IPv6Address": ""
  },
  "e14240883d3ff4a870eb557f093910754d5c180b4d2bf8c4455dbbfc8f96d9bd": {
    "Name": "ubuntu_1",
    "EndpointID": "0e5efa82935d5f203a5d1f549d33d3f7845051744ecec00896d39313525c154",
    "MacAddress": "02:42:c0:a8:00:02",
    "IPv4Address": "192.168.0.2/20",
    "IPv6Address": ""
  }
}

```

```

[71] > [72] ~/docker
[73] ifconfig | grep veth
veth4d54f5f: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
veth63890be: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
vethd5a48c0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

```

ubuntu\_3 -> ubuntu\_2

```

[71] > [72] ~
[73] docker run --rm --name ubuntu_3 --net=my_bridge2 -it custom_ubuntu:0
→ docker ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=64 time=0.179 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=64 time=0.064 ms
64 bytes from 192.168.0.3: icmp_seq=3 ttl=64 time=0.094 ms
64 bytes from 192.168.0.3: icmp_seq=4 ttl=64 time=0.096 ms

```

\*receiver

```

[71] > [72] ~/docker
[73] sudo tcpdump -i veth4d54f5f
[sudo] password for ksu_kotovaa:
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on veth4d54f5f, link-type EN10MB (Ethernet), snapshot length 262144 B
16:49:28.260614 IP 192.168.0.4 > 192.168.0.3: ICMP echo request, id 4, seq 105,
16:49:28.260642 IP 192.168.0.3 > 192.168.0.4: ICMP echo reply, id 4, seq 105,

```

\*sender

```

[71] > [72] ~/docker
[73] sudo tcpdump -i veth63890be
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on veth63890be, link-type EN10MB (Ethernet), snapshot length 262144 B
16:52:48.964583 IP 192.168.0.4 > 192.168.0.3: ICMP echo request, id 4, seq 301,
16:52:48.964648 IP 192.168.0.3 > 192.168.0.4: ICMP echo reply, id 4, seq 301,
16:52:49.055670 IP fedora.60774 > 192.168.0.4.llmnr: Flags [S], seq 624574681,

```

\*my\_bridge2

```
root@my_bridge2:~# docker exec -it br-80ed28a31ca5 /bin/bash
root@br-80ed28a31ca5:~# sudo tcpdump -i br-80ed28a31ca5
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br-80ed28a31ca5, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:03:36.516422 IP 192.168.0.4 > 192.168.0.3: ICMP echo request, id 6, seq 22, length 64
17:03:36.516485 IP 192.168.0.3 > 192.168.0.4: ICMP echo reply, id 6, seq 22, length 64
17:03:36.805696 IP fedora.56494 > 192.168.0.3: llmnr: Flags [S], seq 3906350614, win 32120, options [mss
r 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
17:03:36.805739 IP 192.168.0.3: llmnr > fedora.56494: Flags [R.], seq 0, ack 3906350615, win 0, length 0
17:03:37.055739 IP fedora.39440 > 192.168.0.4: llmnr: Flags [S], seq 1072667388, win 32120, options [mss
cr 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
17:03:37.055779 IP 192.168.0.4: llmnr > fedora.39440: Flags [R.], seq 0, ack 1072667389, win 0, length 0
17:03:37.540572 IP 192.168.0.4 > 192.168.0.3: ICMP echo request, id 6, seq 23, length 64
17:03:37.540635 IP 192.168.0.3 > 192.168.0.4: ICMP echo reply, id 6, seq 23, length 64
```

ubuntu\_3 -> ubuntu\_1

```
root@ubuntu_3:~# docker exec -it ubuntu_1 ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.235 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.090 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.119 ms
64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.088 ms
```

\*sender

```
root@sender:~# docker exec -it veth63890be /bin/bash
root@veth63890be:~# sudo tcpdump -i veth63890be
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on veth63890be, link-type EN10MB (Ethernet), snapshot length 262144 bytes
16:58:33.540590 IP 192.168.0.4 > 192.168.0.2: ICMP echo request, id 5, seq 62, length 64
16:58:33.540663 IP 192.168.0.2 > 192.168.0.4: ICMP echo reply, id 5, seq 62, length 64
16:58:33.805588 ARP, Request who-has 192.168.0.2 tell fedora, length 28
16:58:34.055601 IP fedora.44388 > 192.168.0.4: llmnr: Flags [S], seq 750373330, win 32120, options [mss
r 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
16:58:34.055635 IP 192.168.0.4: llmnr > fedora.44388: Flags [R.], seq 0, ack 750373331, win 0, length 0
```

\*receiver

```
root@receiver:~# docker exec -it vethd5a48c0 /bin/bash
root@vethd5a48c0:~# sudo tcpdump -i vethd5a48c0
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on vethd5a48c0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:00:37.444644 IP 192.168.0.4 > 192.168.0.2: ICMP echo request, id 5, seq 183, length 64
17:00:37.444669 IP 192.168.0.2 > 192.168.0.4: ICMP echo reply, id 5, seq 183, length 64
17:00:37.555777 IP fedora.45050 > 192.168.0.2: llmnr: Flags [S], seq 3139759947, win 32120, options [mss
cr 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
17:00:37.555811 IP 192.168.0.2: llmnr > fedora.45050: Flags [R.], seq 0, ack 3139759948, win 0, length 0
17:00:38.468625 IP 192.168.0.4 > 192.168.0.2: ICMP echo request, id 5, seq 184, length 64
17:00:38.468651 IP 192.168.0.2 > 192.168.0.4: ICMP echo reply, id 5, seq 184, length 64
```

\*my\_bridge2

```
root@my_bridge2:~# docker exec -it br-80ed28a31ca5 /bin/bash
root@br-80ed28a31ca5:~# sudo tcpdump -i br-80ed28a31ca5
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on br-80ed28a31ca5, link-type EN10MB (Ethernet), snapshot length 262144 bytes
17:06:25.156467 IP 192.168.0.4 > 192.168.0.2: ICMP echo request, id 8, seq 5, length 64
17:06:25.156531 IP 192.168.0.2 > 192.168.0.4: ICMP echo reply, id 8, seq 5, length 64
17:06:25.305761 IP fedora.50266 > 192.168.0.2: llmnr: Flags [S], seq 2651917167, win 32120, options [mss
cr 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
17:06:25.305816 IP 192.168.0.2: llmnr > fedora.50266: Flags [R.], seq 0, ack 2651917168, win 0, length 0
17:06:25.555681 IP fedora.59044 > 192.168.0.4: llmnr: Flags [S], seq 1541381547, win 32120, options [mss
cr 0,nop,wscale 7,tfo cookiereq,nop,nop], length 0
17:06:25.555734 IP 192.168.0.4: llmnr > fedora.59044: Flags [R.], seq 0, ack 1541381548, win 0, length 0
17:06:26.180636 IP 192.168.0.4 > 192.168.0.2: ICMP echo request, id 8, seq 6, length 64
17:06:26.180696 IP 192.168.0.2 > 192.168.0.4: ICMP echo reply, id 8, seq 6, length 64
17:06:26.554384 ARP, Request who-has 192.168.0.4 tell 192.168.0.2, length 28
```

- Создать свою собственную сеть overlay, проверить, создана ли она, вывести о ней всю информацию

```

[1] [2] ~/docker
- docker swarm init
Swarm initialized: current node (i9oky6gvaih4y7hmn8q53bv5m) is now a manager.

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

    docker swarm join --token SWMTKN-1-0mcxbcxpjol2pn6rz41tnxysy7ziawnm4k3pqfn1sn945hk6f-1hcg3imvv055gh3m9liul13gcj 192.168.100.79:2377

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

[1] [2] ~/docker
- docker network create -d overlay my_overlay_network
jpwpg3bvr5e8rrp461vzsb6yi

```

```

[1] [2] ~/docker
- docker network ls

```

NETWORK ID	NAME	DRIVER	SCOPE
a94969e55272	back_net	bridge	local
9af964dbe248	bridge	bridge	local
674a38ccf3a0	docker_gwbridge	bridge	local
c3c57f637b79	host	host	local
73r0xw7n1thu	ingress	overlay	swarm
d46352482d81	my_bridge	bridge	local
80ed28a31ca5	my_bridge2	bridge	local
jpwpg3bvr5e8	my_overlay_network	overlay	swarm
5feb3e46dff9	none	null	local
aa74e022652f	task_6.4_net	bridge	local
cbb934a8865d	task_7.1_net	bridge	local

```

[1] [2] ~/docker
- docker network inspect my_overlay_network
{
  "Name": "my_overlay_network",
  "Id": "jpwpg3bvr5e8rrp461vzsb6yi",
  "Created": "2024-03-10T16:15:44.537761599Z",
  "Scope": "swarm",
  "Driver": "overlay",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": null,
    "Config": [
      {
        "Subnet": "10.0.1.0/24",
        "Gateway": "10.0.1.1"
      }
    ]
  },
  "Internal": false,
  "Attachable": false,
  "Ingress": false,
  "ConfigFrom": {
    "Network": ""
  },
  "ConfigOnly": false,
  "Containers": null,
  "Options": {
    "com.docker.network.driver.overlay.vxlanid_list": "4097"
  },
  "Labels": null
}

```

- Создать еще одну сеть overlay, проверить, создана ли она, вывести о ней всю информацию, удалить сеть

```

[1] [2] ~/docker
- docker network create -d overlay my_overlay_network2
h52geeyrg2oziy8v0gwylu0h9

[1] [2] ~/docker
- docker network ls

```

NETWORK ID	NAME	DRIVER	SCOPE
a94969e55272	back_net	bridge	local
9af964dbe248	bridge	bridge	local
674a38ccf3a0	docker_gwbridge	bridge	local
c3c57f637b79	host	host	local
73r0xw7n1thu	ingress	overlay	swarm
d46352482d81	my_bridge	bridge	local
80ed28a31ca5	my_bridge2	bridge	local
jpwpg3bvr5e8	my_overlay_network	overlay	swarm
h52geeyrg2oz	my_overlay_network2	overlay	swarm
5feb3e46dff9	none	null	local
aa74e022652f	task_6.4_net	bridge	local
cbb934a8865d	task_7.1_net	bridge	local

```
~/docker
docker network inspect my_overlay_network2
[
  {
    "Name": "my_overlay_network2",
    "Id": "h52geeyrg2oziy8v0gwylu0h9",
    "Created": "2024-03-10T16:18:01.554904738Z",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.0.2.0/24",
          "Gateway": "10.0.2.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": null,
    "Options": {
      "com.docker.network.driver.overlay.vxlanid_list": "4098"
    },
    "Labels": null
  }
]

~/docker
docker network rm my_overlay_network2
my_overlay_network2

~/docker
docker network ls | grep my_overlay_network2
```

- Попробовать создать сеть host, сохранить результат в отчет.

```
~/docker
docker network create -d host another_host
Error response from daemon: only one instance of "host" network is allowed
```