

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе 2
По дисциплине "Избранные главы информатики"
Работа с Docker

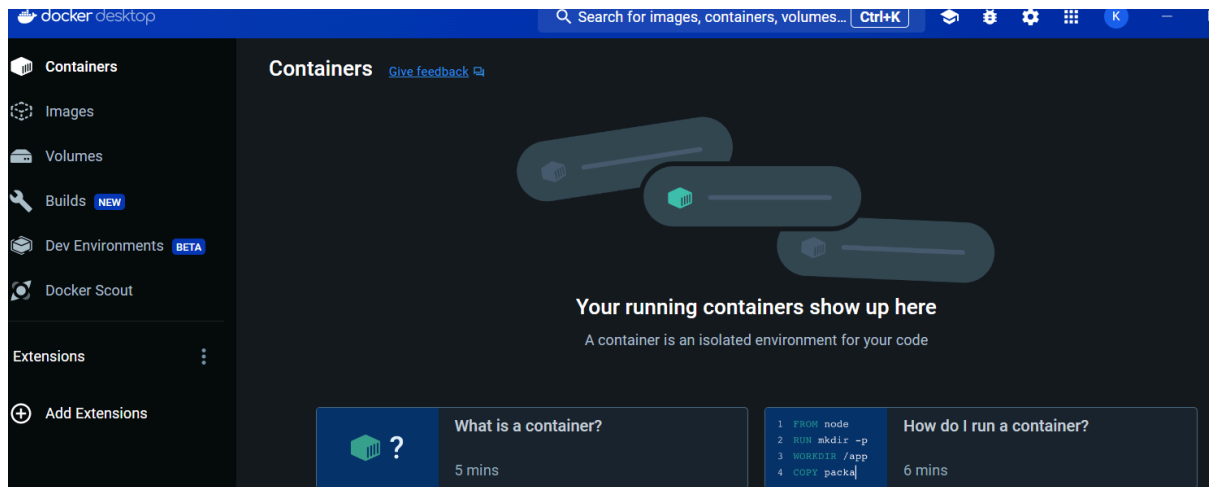
Выполнил:
студент группы 253503
Телего Е.А.

Руководитель:
доцент
Жвакина А.В.

Минск 2024

Задание 1

Подготовьте рабочее окружение в соответствии с типом вашей операционной системы



Задание 2

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

```
D:\>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

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

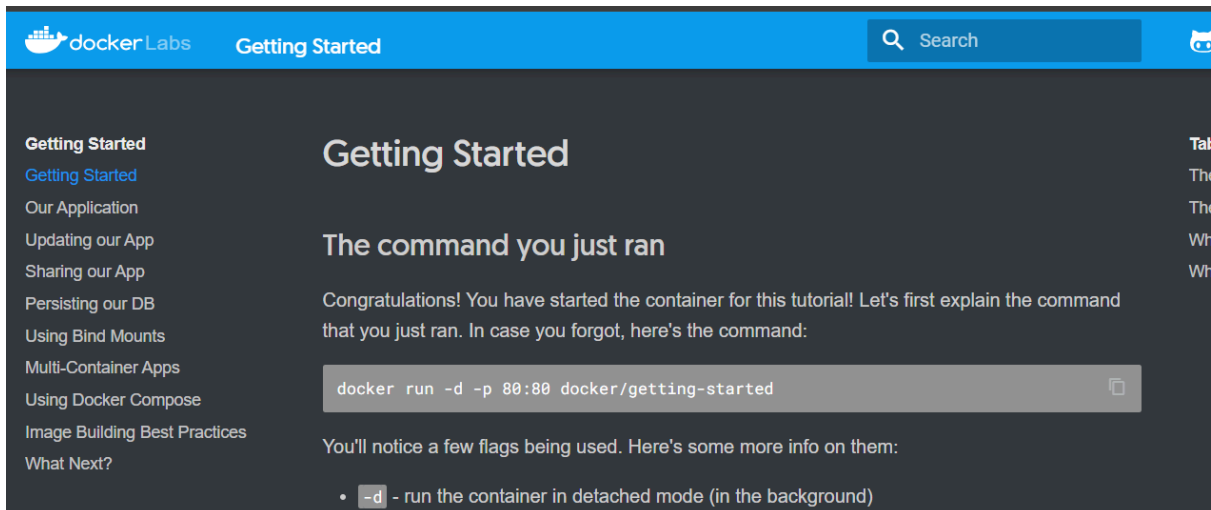
Management Commands:
  builder    Manage builds
  buildx*   Docker Buildx (Docker Inc., v0.12.1-desktop.4)
  compose*  Docker Compose (Docker Inc., v2.24.5-desktop.1)
  container  Manage containers
  context    Manage contexts
  debug*    Get a shell into any image or container. (Docker Inc., 0.0.24)
  dev*      Docker Dev Environments (Docker Inc., v0.1.0)
```

```
D:\>docker pull docker/getting-started
Using default tag: latest
latest: Pulling from docker/getting-started
c158987b0551: Pull complete
```

```
D:\>docker run -d -p 80:80 docker/getting-started
694e9b3651684ae5032c1e06029b87b0c3686f7a63c4aa23200f369fa461017f

D:\>docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS        PORTS                    NAMES
694e9b365168   docker/getting-started  "/docker-entrypoint..."  12 seconds ago  Up 11 seconds  0.0.0.0:80->80/tcp      compassionate_fermi
```

Container ID	Image	Command	Created	Status	Ports	Names
694e9b36516	compassior	docker/getting-star	Running	0%	80:80	23 seconds ago



Задание 3

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

```
FROM python:3.12-slim
WORKDIR /app
COPY . .
ENTRYPOINT ["python", "lab2.py"]
CMD ["1"]
```
















```
import circle
import square
import sys
args = sys.argv[1:]
if args:
    val = int(args[0])
    print(f"Square area of square with side = {val} equals {square.area(val)} ")
    print(f"Perimeter of square with side = {val} equals {square.perimeter(val)} ")
    print(f"Cirlce area of circle with radius = {val} equals {circle.area(val)} ")
    print(f"Circle area of side = {val} equals {circle.perimeter(val)} ")
else:
    print("INVALID ARGUMENTS")
```

```
PS D:\IGI\LAB2> docker build -t tuckercarlson .
```

```
PS D:\IGI\LAB2> docker run tuckercarlson 2
Square area of square with side = 2 equals 4
Perimeter of square with side = 2 equals 8
Circle area of circle with radius = 2 equals 12.566370614359172
Circle area of side = 2 equals 12.566370614359172
PS D:\IGI\LAB2>
```

Задание 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.

<input type="checkbox"/>	 lab2_compose	Running (2/2)	0%	3 seconds ago	  
<input type="checkbox"/>	 mongo-db-explorer-container 94e2d7e95db3  mongo-express	Running	0% 8081:8081 	3 seconds ago	  
<input type="checkbox"/>	 mongo-db-container 3187e4408493  mongo	Running	0% 27017:27017 	3 seconds ago	  

 View	aboba	 Del
 View	admin	 Del
 View	config	 Del
 View	local	 Del

Server Status			
Hostname	e1fb17c9d70d	MongoDB Version	7.0.6
Uptime	24 seconds	Node Version	18.19.1
Server Time	Tue, 12 Mar 2024 19:09:30 GMT	V8 Version	10.2.154.26-node.28

```

version: '3.9'
name: lab2_compose

services:
  mongo-db:
    container_name: mongo-db_container
    image: mongo
    restart: always
    volumes:
      - dbdata:/data/db
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
    networks:
      - db-network

```

```

  mongo-db-explorer:
    container_name: mongo-db-explorer-container
    image: mongo-express
    restart: always
    depends_on:
      - mongo-db
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_BASICAUTH_USERNAME: admin
      ME_CONFIG_BASICAUTH_PASSWORD: admin
      ME_CONFIG_MONGODB_URL: mongodb://root:example@mongo-db:27017/
    networks:
      - db-network

volumes:
  dbdata:

networks:
  db-network:

```

Задание 5

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

```

volumes:
  - dbdata:/data/db
ports:

```

```
volumes:
  dbdata:
```

 dbdata

3 days ago

8 kB

Задание 6

Разместите результат в созданный репозиторий в DockerHub

```
D:\>docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have
over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants l
is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: karasinius
Password:
Login Succeeded

D:\>docker tag mongo karasinius/lr_2

D:\>docker push karasinius/lr_2
Using default tag: latest
The push refers to repository [docker.io/karasinius/lr_2]
dbedb1e17615: Mounted from library/mongo
7718c1a89816: Mounted from library/mongo
c181b6726969: Mounted from library/mongo
307be78e0018: Mounted from library/mongo
ae9250d8ab87: Mounted from library/mongo
67d9b164d3bb: Mounted from library/mongo
8832d958e48b: Mounted from library/mongo
5498e8c22f69: Mounted from library/mongo
latest: digest: sha256:d377cc142c913bc0d9110441468424bafa8832b133c1611d0e507eec9dcf7707 size: 1994
D:\>
```

karasinius / lr_2_express

Contains: Image • Last pushed: 17 minutes ago

Security unknown

☆ 0

↓ 0

Public

karasinius / lr_2

Contains: Image • Last pushed: 18 minutes ago

Security unknown

☆ 0

↓ 0

Public

Задание 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:

```
D:\>docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
e3d16ca578d8	bridge	bridge	local
6f446a8630b6	host	host	local
811e79ed6184	lab2_compose_db-network	bridge	local
2b72e2dc0c5b	none	null	local

```
D:\>docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "e3d16ca578d80650f4b44a70279144d4683a4cef10750b9318bc9ecf9aed8e4c",
    "Created": "2024-03-14T15:44:21.1294088Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {}
  }
]
```

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

Driver bridge по умолчанию

Если надо указать, то `docker network create -d bridge aboba`


```
D:\>docker network create aboba
d22e51987148857612c3e6bd57d9e3d9ed18805ab8974bfd38f9a4754b9fd22e
```

```
D:\>docker network ls
```
















NETWORK ID	NAME	DRIVER	SCOPE
d22e51987148	aboba	bridge	local
e3d16ca578d8	bridge	bridge	local
6f446a8630b6	host	host	local
811e79ed6184	lab2_compose_db-network	bridge	local
2b72e2dc0c5b	none	null	local

```
D:\>docker network inspect aboba
[
  {
    "Name": "aboba",
    "Id": "d22e51987148857612c3e6bd57d9e3d9ed18805ab8974bfd38f9a4754b9fd22e",
    "Created": "2024-03-14T16:28:00.8660986Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    }
  }
]
```

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

```
D:\>docker network create mynetwork
11e01f512617bdfccbe4c66519a9fb86222a7724d2c5a2b6ca222fef90cdc705
```

```
D:\>docker run -it -d --name web3 --network mynetwork alpine ash
a11ff955be3bc9c5b1cd96e4467f4b7aa5ccca0b368f4cb21fee0146c624f4a7
```

<input type="checkbox"/>	 web1 7e0c3aea87ab 	alpine	Running	0%	23 seconds ago	  
<input type="checkbox"/>	 web2 d1e8481b0d7b 	alpine	Running	0%	14 seconds ago	  
<input type="checkbox"/>	 web3 a11ff955be3b 	alpine	Running	0%	6 seconds ago	  

```

"Containers": {
  "7e0c3aea87abc3a66e1a0e538b99c5fd4eb0c57464d30612f621040e606278": {
    "Name": "web1",
    "EndpointID": "35607c2e52cc0fe499da6e1d6ee1296545a2eaad8f1ba50bef6c10d7192bb4b9",
    "MacAddress": "02:42:ac:13:00:02",
    "IPv4Address": "172.19.0.2/16",
    "IPv6Address": ""
  },
  "a11ff955be3bc9c5b1cd96e4467f4b7aa5ccca0b368f4cb21fee0146c624f4a7": {
    "Name": "web3",
    "EndpointID": "cf9f23cc8e2f9d0e1d16ab1791bde09076c8befa66de8fa2b217bff9cd4f57e7",
    "MacAddress": "02:42:ac:13:00:04",
    "IPv4Address": "172.19.0.4/16",
    "IPv6Address": ""
  },
  "d1e8481b0d7b9a9b58c69d6182d3c00c0953c0699e2c76bec3752548d5217433": {
    "Name": "web2",
    "EndpointID": "c0608f9a1420f389f00cdc45b872b14bc4e03a4820f0f3e61ec4022e78f1cf59",
    "MacAddress": "02:42:ac:13:00:03",
    "IPv4Address": "172.19.0.3/16",
    "IPv6Address": ""
  }
}

```

```

D:\>docker container attach web1
/ # ping -c 2 web2
PING web2 (172.19.0.3): 56 data bytes
64 bytes from 172.19.0.3: seq=0 ttl=64 time=0.129 ms
64 bytes from 172.19.0.3: seq=1 ttl=64 time=0.075 ms

--- web2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/0.102/0.129 ms
/ # ping -c 2 web3
PING web3 (172.19.0.4): 56 data bytes
64 bytes from 172.19.0.4: seq=0 ttl=64 time=0.186 ms
64 bytes from 172.19.0.4: seq=1 ttl=64 time=0.223 ms

--- web3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.186/0.204/0.223 ms

```

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

```

D:\>docker swarm init
Swarm initialized: current node (tn0qoyn78l271y055jhe8n8b6) is now a manager.

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

    docker swarm join --token SWMTKN-1-5i3sn3jfqbh7dn3s64fvp19q0qkq6yxai02ggobos4z9fqfh6x-dj96qvhiigcw9x9he2q8oz6ts 192.168.65.3:2377

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

D:\>docker network create --driver overlay myoverlay
tupmiv5wrg68ryougfd9j13

```

```
D:\>docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
d22e51987148	aboba	bridge	local
862ad2a5a547	bridge	bridge	local
383fd537f986	docker_gwbridge	bridge	local
6f446a8630b6	host	host	local
i02w5z8lt0o8	ingress	overlay	swarm
811e79ed6184	lab2_compose_db-network	bridge	local
11e01f512617	mynetwork	bridge	local
tupmiv5wrg68	myoverlay	overlay	swarm
2b72e2dc0c5b	none	null	local

```
D:\>docker network inspect myoverlay
```

```
[
  {
    "Name": "myoverlay",
    "Id": "tupmiv5wrg68ryougfqds9j13",
    "Created": "2024-03-14T17:56:14.0523187Z",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.0.1.0/24",
          "Gateway": "10.0.1.1"
        }
      ]
    }
  }
]
```

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

```
D:\>docker network create --driver overlay myoverlay2
rt964xnt33nscdk2c0grc21oz

D:\>docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
d22e51987148        aboba               bridge              local
862ad2a5a547        bridge              bridge              local
383fd537f986        docker_gwbridge     bridge              local
6f446a8630b6        host                host                local
i02w5z8lt0o8        ingress             overlay             swarm
811e79ed6184        lab2_compose_db-network bridge              local
11e01f512617        mynetwork           bridge              local
tupmiv5wrg68        myoverlay           overlay             swarm
rt964xnt33ns        myoverlay2          overlay             swarm
2b72e2dc0c5b        none                null                local

D:\>docker network inspect myoverlay2
[
  {
    "Name": "myoverlay2",
    "Id": "rt964xnt33nscdk2c0grc21oz",
    "Created": "2024-03-14T17:57:54.0133447Z",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false
```

```
D:\>docker network rm myoverlay2
myoverlay2

D:\>docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
d22e51987148        aboba               bridge              local
862ad2a5a547        bridge              bridge              local
383fd537f986        docker_gwbridge     bridge              local
6f446a8630b6        host                host                local
i02w5z8lt0o8        ingress             overlay             swarm
811e79ed6184        lab2_compose_db-network bridge              local
11e01f512617        mynetwork           bridge              local
tupmiv5wrg68        myoverlay           overlay             swarm
2b72e2dc0c5b        none                null                local
```

Попробовать создать сеть host, сохранить результат в отчет.

```
D:\>docker network create --driver host myhost
Error response from daemon: only one instance of "host" network is allowed
```

