

Избранные главы информатики

Лабораторная работа №2

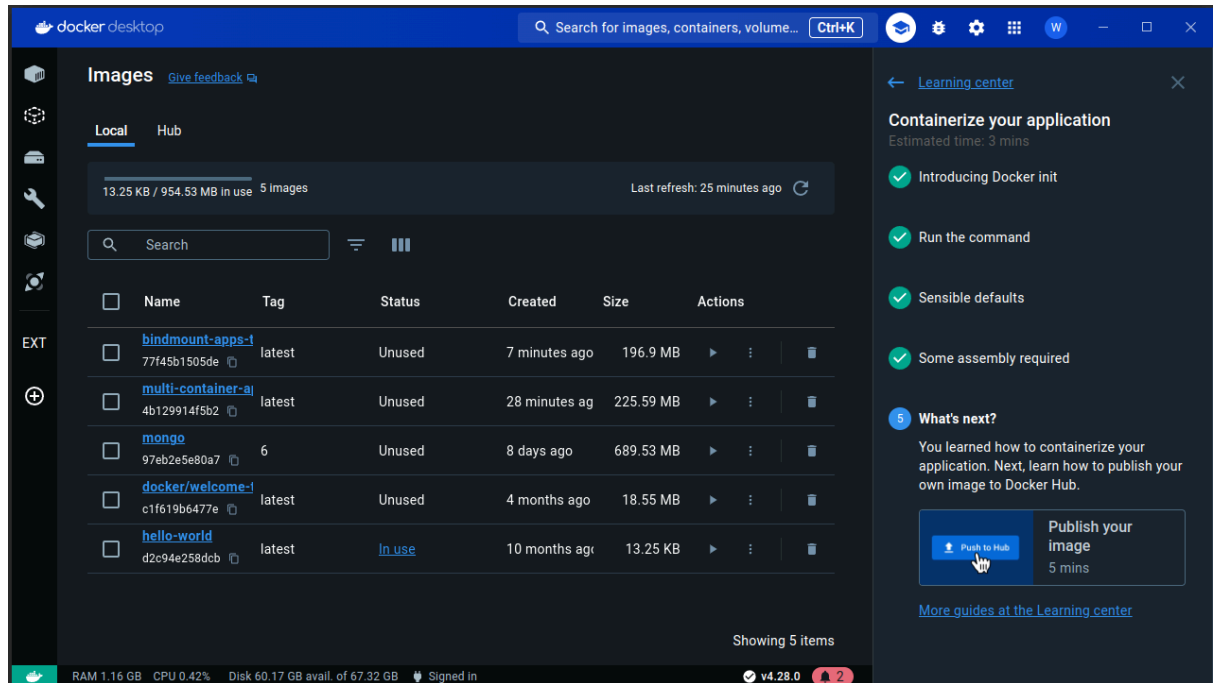
подготовил:

Макаров Алексей

студент группы 253503

Задание 1

Был установлен docker. Я провел все настройки и подготовил его к работе. Скачивание на Linux-систему было проведено с помощью утилиты apt-get.

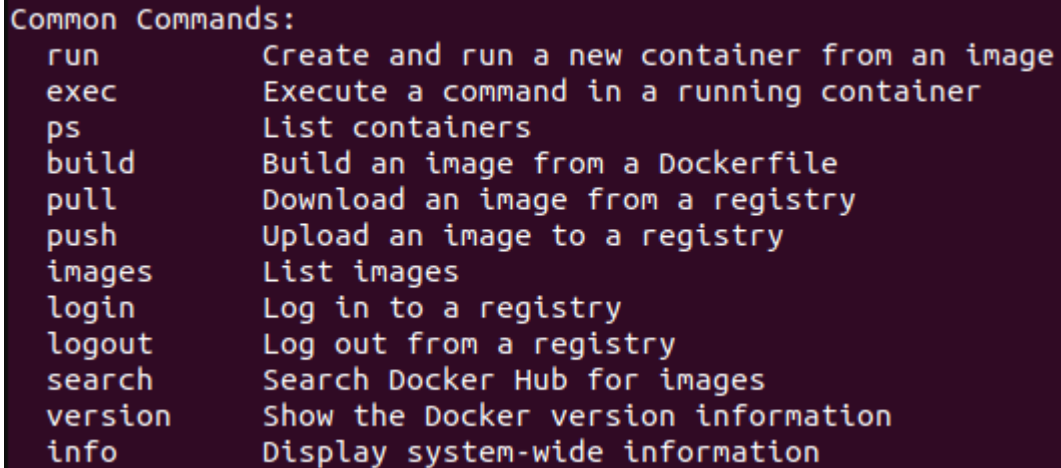


Внешний вид Docker desktop

Данное приложение содержит в себе две основные вещи для работы с Docker-ом. Это Docker Compose и Docker Engine. Я все сделал.

Задание 2

Все базовые команды можно просмотреть с помощью команды `docker`.

A screenshot of a terminal window with a dark purple background. It displays a list of common Docker commands and their descriptions. The text is as follows:

```
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
```

Слева у нас находятся сами команды, а справа описание действий.

Запустим контейнер `getting-started`.

При переходе по ссылке мы попадаем на данный сайт

Getting Started

The command you just ran

Congratulations! You have started the container for this tutorial! Let's first explain the command that you just ran. In case you forgot, here's the command:

```
docker run -d -p 80:80 docker/getting-started
```

You'll notice a few flags being used. Here's some more info on them:

- `-d` - run the container in detached mode (in the background)
- `-p 80:80` - map port 80 of the host to port 80 in the container
- `docker/getting-started` - the image to use

Pro tip

You can combine single character flags to shorten the full command. As an example, the command above could be written as:

```
docker run -dp 80:80 docker/getting-started
```

The Docker Dashboard

Before going any further, we want to highlight the Docker Dashboard, which gives you a quick view of the containers running on your machine. It provides you access to container logs, lets you get a shell inside the container, and allows you to easily manage container lifecycle (stop,

Тutorial был выполнен

	What is a container? ✓ Completed
<pre>1 FROM node 2 RUN mkdir -p 3 WORKDIR /app 4 COPY packa</pre>	How do I run a container? ✓ Completed
	Run Docker Hub images ✓ Completed
	Multi-container applications ✓ Completed
	Persist your data between containers ✓ Completed
	Access your local folder from a container ✓ Completed
<code>\$ docker init</code>	Containerize your application ✓ Completed
	Publish your image ✓ Completed

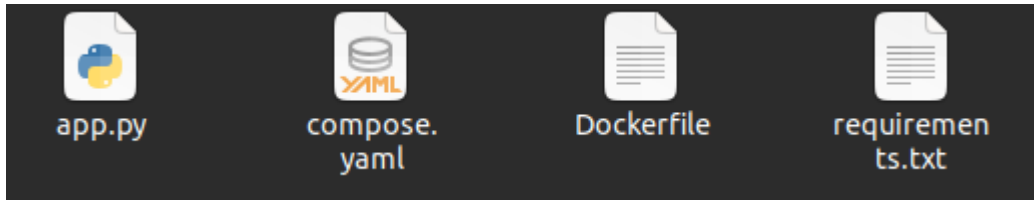
Задание 3

Создадим файл `main.py` там будет располагаться наш скрипт, для подсчета площади и периметра круга. Радиус задается переменной среды. При ее отсутствии радиус полагается 4. Результат выполнения программы

```
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/253503_MAKAROV_16/IGI/LR2$ docker run -e RADUIS=7 first
^[[A^[[ACircle area with radius 7: 153.93804002589985
Circle perimeter with radius 7: 43.982297150257104
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/253503_MAKAROV_16/IGI/LR2$ docker run --rm first
Circle area with radius 4: 50.26548245743669
Circle perimeter with radius 4: 25.132741228718345
```

Задание 4

За основу я взял себе свое веб-приложение, написанное на фласке. Файл app.py является основным, там расположена вся логика приложения.



Структура рабочей папки

Докерфайл

```
1 # syntax=docker/dockerfile:1
2 FROM python:3.10-alpine
3 WORKDIR /code
4 ENV FLASK_APP=app.py
5 ENV FLASK_RUN_HOST=0.0.0.0
6 RUN apk add --no-cache gcc musl-dev linux-headers
7 COPY requirements.txt requirements.txt
8 RUN pip install -r requirements.txt
9 EXPOSE 5000
10 COPY . .
11 CMD ["flask", "run"]
```

Compose-file

```
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

Задание 5)

Добавим том в докерфайл из прошлого задания

```

1 services:
2   web:
3     build: .
4     ports:
5       - "8000:5000"
6     volumes:
7       - ./app
8   redis:
9     image: "redis:alpine"

```

Вот последовательность команд для создания сети bridge. Затем мы подключаем контейнеры к сети.

```
Task4$ docker network create my-net
```

```
Task4$ docker run --network=my-net -d redis:alpine
```

```
Task4$ docker run --network=my-net -d task4-web
```

Задание 6)

Добавим контейнер из четвертого задания в свой репозиторий.

```

docker tag task4-web worsen/task4-web
docker push worsen/task4-web

```

Задание 7)

просмотрим все сети на нашем устройстве

```

16/IGI/LR2$ docker network ls

```

NETWORK ID	NAME	DRIVER	SCOPE
7b109ae2780c	bridge	bridge	local
73fdfb89fea3	host	host	local
347fa748c287	my-net	bridge	local
9cb073fa2c3d	none	null	local

Просмотрим информацию о нашем хосте

```
16/IGI/LR2$ docker inspect 73fdfb89fea3
[
  {
    "Name": "host",
    "Id": "73fdfb89fea319510cd78880a1356068e41df87ca295fc169182fcf7d8a3bca5",
    "Created": "2024-03-02T18:40:28.365390849Z",
    "Scope": "local",
    "Driver": "host",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": null
    },
  },
]
```

подключим контейнер к сети

```
docker run --network=my-bridge-network -d mongo:6
```

Подключим еще три контейнера к нашей сети

```
]
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/
16/IGI/LR2$ docker run --network=my-bridge-network -d mongo:6
541f6acb8109fc199a0bc788b1faf1bb8dd1309d505068d4cdef0e42fd478507
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/
16/IGI/LR2$ docker run --network=my-bridge-network -d mongo:6
9f630ac4e1741c6021f26b3fcab77ee4d372d6e7add7ae9a9066f9a0607c973d
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/
16/IGI/LR2$ docker run --network=my-bridge-network -d mongo:6
99c374865f9cb0cbf2836743b99b16c69a68554bf946487526982b9231660e64
```

Создадим собственную сеть оверлей

```
I/LR2$ docker network create my-overlay-network --driver overlay
or "docker swarm join" to connect this node to swarm and try again.
I/LR2$ docker swarm init
```

```
90e-adrb7igird51daw8w89q8xnle 192.168.65.9:2377
```

nstructions.

```
I/LR2$ docker network create my-overlay-network --driver overlay
```



```
s/IGI/LR2$ docker network create second-overlay --driver overlay  
s/IGI/LR2$ docker network rv second-overlay
```

```
s/IGI/LR2$ docker network rm second-overlay
```

создали и удалили вторую сеть

МЫ НЕ СОЗДАЛИ НОВЫЙ ХОСТ

```
bass-n-cripp@bassncripp-HP-Pavilion-Laptop-15-eh1xxx:~/university/253503.MAKAROV_16/IGI/LR2$ docker network create my-host --driver host  
Error response from daemon: only one instance of "host" network is allowed
```