

5.7 Практическая работа

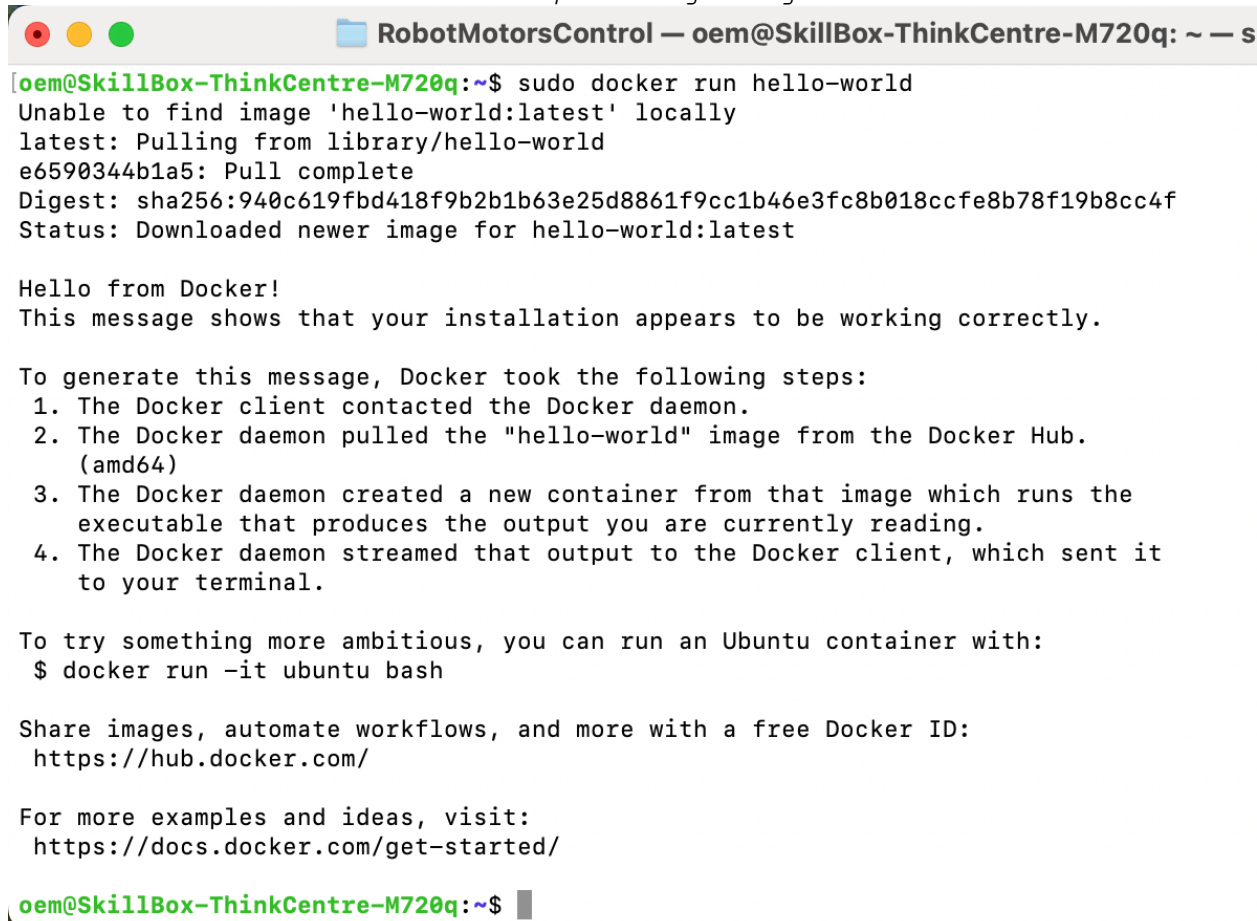
Цели практической работы

- Развить навыки профессионального развёртывания простых программ с использованием Docker.
- Научиться оформлять и представлять свой проект.

Для выполнения практической работы следуйте алгоритму, описанному ниже.

1. Установка Docker и Docker Compose:

а. Установите Docker и Docker Compose на вашу систему.

A terminal window titled "RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~ — s". The prompt is "oem@SkillBox-ThinkCentre-M720q:~\$". The user enters "sudo docker run hello-world". The output shows Docker pulling the "hello-world:latest" image from the library, displaying the digest, and then printing "Hello from Docker!". It explains that the installation is working and lists the steps taken: contacting the daemon, pulling the image, creating a container, and streaming output. It then suggests running an Ubuntu container and provides links for more information and Docker ID registration. The prompt returns to "oem@SkillBox-ThinkCentre-M720q:~\$".

```
oem@SkillBox-ThinkCentre-M720q:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:940c619fbd418f9b2b1b63e25d8861f9cc1b46e3fc8b018ccfe8b78f19b8cc4f
Status: Downloaded newer image for hello-world:latest

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/

oem@SkillBox-ThinkCentre-M720q:~$
```

б. Настройте Docker для работы без прав root (добавление пользователя в группу docker).

```

RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~ — ss
[oem@SkillBox-ThinkCentre-M720q:~]$ sudo usermod -aG docker oem
[oem@SkillBox-ThinkCentre-M720q:~]$ newgrp docker
[oem@SkillBox-ThinkCentre-M720q:~]$ 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/>

```
oem@SkillBox-ThinkCentre-M720q:~$
```

```

RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~
100 56.8M 100 56.8M 0 0 2630k 0 0:00:22 0:00:22 --:--:-- 4112k
[oem@SkillBox-ThinkCentre-M720q:~]$ sudo chmod +x /usr/local/bin/docker-compose
[oem@SkillBox-ThinkCentre-M720q:~]$ docker-compose --version
Docker Compose version v2.23.0
oem@SkillBox-ThinkCentre-M720q:~$

```

2. Разработка простой программы:

- a. Напишите простую программу на Python или Bash. Например, программа может выполнять простую симуляцию работы или даже выводить текущее время и дату или простое приветствие.

Программа на Python

```

RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~/Desktop/do
GNU nano 4.8 test.py
def add(a,b):
    print( a+b )

if __name__ == "__main__":
    add(3, 5)
    print("Успешно выполнено с помощью test.py")

```

```

#!/bin/bash
echo "Hello, $USER!"
echo "Today is $(date)"

```

```

RobotMotorsControl — oem@SkillBox-
[oem@SkillBox-ThinkCentre-M720q:~]$ nano myscript
[oem@SkillBox-ThinkCentre-M720q:~]$ chmod +x ./myscript
[oem@SkillBox-ThinkCentre-M720q:~]$ ./myscript
Hello, oem!
Today is Вт 17 июн 2025 11:31:30 MSK
oem@SkillBox-ThinkCentre-M720q:~$

```

- b. Создайте репозиторий на GitHub для вашего проекта.

3. Создание Docker-образа для программы:

- Создайте Dockerfile для сборки образа, включающего вашу программу и зависимости.
- Соберите Docker-образ из вашего Dockerfile.

```
RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~ — ssh oem@192.168.1.64 — 141x35
oem@SkillBox-ThinkCentre-M720q:~$ docker image build -t python:0.0.1 /home/oem/Desktop/docker_2/docker_assignment
[+] Building 114.7s (8/8) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 153B
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/python:latest@sha256:5f69d22a88dd4cc4ee1576def19aef48c8faa1b566054c44291183831cbad13b
=> => resolve docker.io/library/python:latest@sha256:5f69d22a88dd4cc4ee1576def19aef48c8faa1b566054c44291183831cbad13b
=> => sha256:5f69d22a88dd4cc4ee1576def19aef48c8faa1b566054c44291183831cbad13b 9.72kB / 9.72kB
=> => sha256:1ae8a7a2593de66cd1398659363f8ea92db4a634bffa4db074a7bd53303ada 2.32kB / 2.32kB
=> => sha256:3b29f43b77ff1f92cbea7613c52afdf7725af4fec4821263114389b09b9a86bf 6.32kB / 6.32kB
=> => sha256:0c01110621e0ec1ded421406c9f11777ae5486c8f7b0a0d1a37cc7bc9317226 48.49MB / 48.49MB
=> => sha256:3b1eb73e993990490aa137c00e60ff4ca9d1715baf8e888dbb0986275edb13f 24.02MB / 24.02MB
=> => sha256:b1b8a0660a31403a35d70b276c3c86b1200b8683e83cd77a92ec98744017684a 64.40MB / 64.40MB
=> => sha256:48b8862a18fa961ebfbac8484877dd4894e96ee88177d8c4f1f54d9727262bd7 211.37MB / 211.37MB
=> => sha256:28db4e6a2e6295e295119126082b4f39882f50278611e48580e8fbef730b08a 6.16MB / 6.16MB
=> => extracting sha256:0c01110621e0ec1ded421406c9f11777ae5486c8f7b0a0d1a37cc7bc9317226
=> => sha256:77a6ac598bc154025b4b2e393a3ca959116e0d8af9c31659a857ac05ab34ccbb 27.39MB / 27.39MB
=> => sha256:5cc4a19fbac0d0fb7423535182443188713730a05b7ab1104f2116055472c13e 250B / 250B
=> => extracting sha256:3b1eb73e993990490aa137c00e60ff4ca9d1715baf8e888dbb0986275edb13f
=> => extracting sha256:b1b8a0660a31403a35d70b276c3c86b1200b8683e83cd77a92ec98744017684a
=> => extracting sha256:48b8862a18fa961ebfbac8484877dd4894e96ee88177d8c4f1f54d9727262bd7
=> => extracting sha256:28db4e6a2e6295e295119126082b4f39882f50278611e48580e8fbef730b08a
=> => extracting sha256:77a6ac598bc154025b4b2e393a3ca959116e0d8af9c31659a857ac05ab34ccbb
=> => extracting sha256:5cc4a19fbac0d0fb7423535182443188713730a05b7ab1104f2116055472c13e
=> [internal] load build context
=> => transferring context: 139B
=> [2/3] WORKDIR /usr/app/src
=> [3/3] COPY test.py ./
=> => exporting to image
=> => exporting layers
=> => writing image sha256:d0f726b54338ef64833591b7fe3faac3babe7240db9b05e072d660a75306b51
=> => naming to docker.io/library/python:0.0.1
oem@SkillBox-ThinkCentre-M720q:~$
```

4. Запуск и тестирование Python-приложения в Docker-контейнере:

- Запустите Docker-контейнер из созданного образа.
- Проверьте, что ваша программа работает корректно внутри контейнера.

```
RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~/Desktop/docker_2/docker_assignment — ssh oem@192.168.1.64 — 141x22
oem@SkillBox-ThinkCentre-M720q:~/Desktop/docker_2/docker_assignment$ docker image build -t python:0.0.1 /home/oem/Desktop/docker_2/docker_assignment
[+] Building 6.2s (9/9) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 193B
=> [internal] load metadata for docker.io/library/python:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:latest@sha256:5f69d22a88dd4cc4ee1576def19aef48c8faa1b566054c44291183831cbad13b
=> [internal] load build context
=> => transferring context: 59B
=> CACHED [2/4] WORKDIR /usr/app/src
=> CACHED [3/4] COPY test.py ./
=> CACHED [4/4] COPY myscript.sh ./
=> => exporting to image
=> => exporting layers
=> => writing image sha256:0de2bcdad56bd73a2de8efc6f214d0686c8356ac2e34bbb55ca0119934f7a3ec
=> => naming to docker.io/library/python:0.0.1
oem@SkillBox-ThinkCentre-M720q:~/Desktop/docker_2/docker_assignment$ docker run python:0.0.1
8
Успешно выполнено с помощью test.py
oem@SkillBox-ThinkCentre-M720q:~/Desktop/docker_2/docker_assignment$
```

5. Работа с Docker Compose:

- Создайте docker-compose.yml, который запускает ваш Docker-контейнер с программой.

```
RobotMotorsControl — oem@SkillBox-ThinkCentre-M720q: ~/Desktop/docker_2/docker_assignment — ssh:
GNU nano 4.8 docker-compose.yml
version: '2' # Версия синтаксиса docker-compose, поддерживаемого докером.

services: # Описание всех служб (контейнеров), которые будут запущены.
  python: # Имя службы (можете назвать иначе, если хотите).
    image: python:0.0.1 # данный компонент будет запускаться из образа.

    container_name: my_python_container # Название контейнера, отображаемое в списке контейнеров.
                                         # Необязательно, но полезно для идентификации.
```

- Добавьте комментарии в docker-compose.yml, объясняющие его структуру и команды.

```
oem@SkillBox-ThinkCentre-M720q:~/Desktop/docker_2/docker_assignment$ docker-compose up
[+] Building 0.0s (0/0)
[+] Running 1/1
✔ Container my_python_container Recreated
Attaching to my_python_container
my_python_container | Hello, !
my_python_container | Today is Tue Jun 17 15:07:23 UTC 2025
my_python_container | 8
my_python_container | Успешно выполнено с помощью test.py
my_python_container exited with code 0
oem@SkillBox-ThinkCentre-M720q:~/Desktop/docker_2/docker_assignment$
```

- Убедитесь, что Docker Compose позволяет запустить ваш контейнер.

6. Оформление проекта на GitHub:

- Поместите ваш Dockerfile и docker-compose.yml в репозиторий на GitHub.

`git@github.com:Neznaika76/Doker.git`

- Подготовьте README.md, описывающий ваш проект и процесс запуска программы с помощью Docker и Docker Compose.

"python", "/test.py",