Оглавление

[1. ХОД РАБОТЫ 2](#_Toc133712041)

[1.1 Образы 2](#_Toc133712042)

[1.2 Изоляция 3](#_Toc133712043)

[1.3 Работа с портами 4](#_Toc133712044)

[1.4 Именованные контейнеры, остановка и удаление 5](#_Toc133712045)

[1.5 Постоянное хранение данных 6](#_Toc133712046)

[1.5.1 Тома 6](#_Toc133712047)

[1.5.2 Монтирование директорий и файлов 7](#_Toc133712048)

[1.6 Переменные окружения 7](#_Toc133712049)

[1.7 Dockerfile 8](#_Toc133712050)

[1.8 Индивидуальные задания 9](#_Toc133712051)

[Вывод 11](#_Toc133712052)

[Список информационных источников 12](#_Toc133712053)

1. ХОД РАБОТЫ

1.1 Образы

После установки докера необходимо посмотреть на все образы с помощью команды docker image (рис. 1).



Рисунок 1 – Просмотр образов

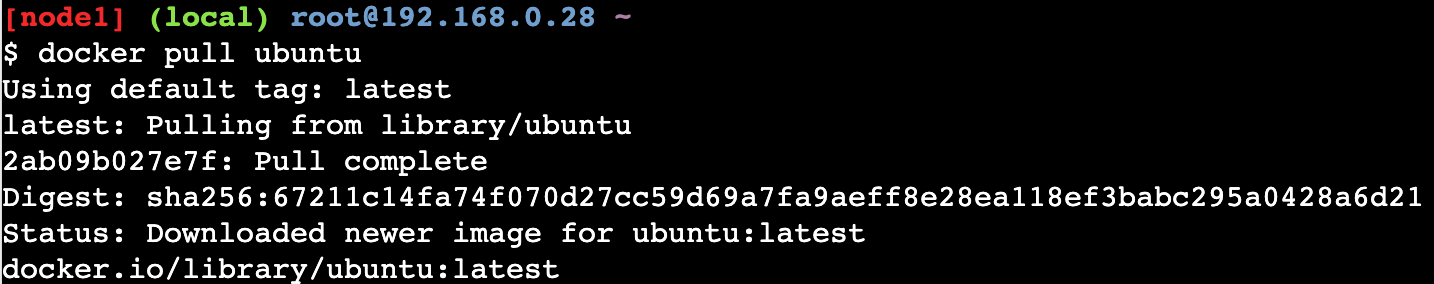
Далее с помощью команды docker pull Ubuntu была загружена Ubuntu последней версии (рис. 2).

Рисунок 2 – загрузка последней версии Ubuntu

После установки последней Ubuntu версии необходимо посмотреть на все образы с помощью команды docker image (рис. 3).

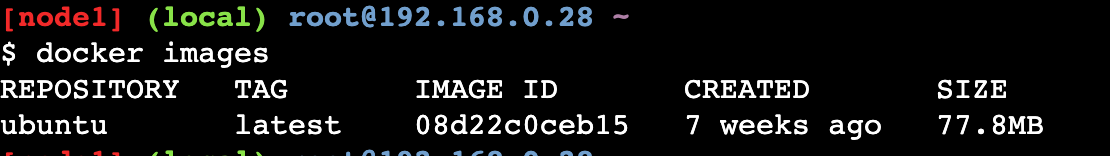


Рисунок 3 – Просмотр образов

Просмотр список текущих контейнеров возможен с помощью команды docker ps -a (рис. 4)



Рисунок 4 – Просмотр текущих контейнеров

## **1.2 Изоляция**

После требовалось вызвать hostname для хостовой системы и для контейнера с Ubuntu и посмотреть на результаты (рис. 5-6).

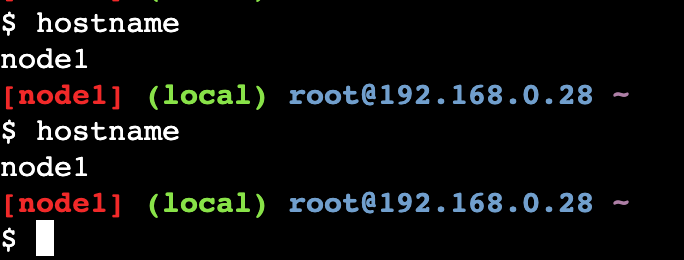


Рисунок 5 – Вызов hostname для хостовой системы

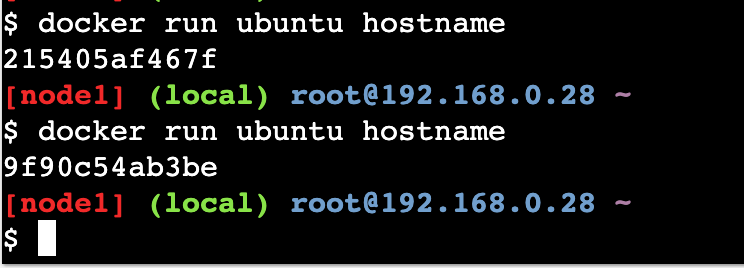


Рисунок 6 – Вызов hostname для контейнера с Ubuntu

В результате выполнения видно, что hostname для хостовой системы одинаковый, а для контейнеров разный. Это происходит потому, что при вызове hostname для контейнера мы открываем два контейнера, что видно на рисунке (рис. 7)



Рисунок 7 – Просмотр открытых контейнеров

Запуск bash из контейнера с Ubuntu производится с помощью команды docker run Ubuntu bash. Но при таком запуске ничего не будет выводиться. Для корректного запуска требуется указать флаги i и t, чтобы открылся интерактивный терминал.

Демонстрация запуска bash представлена на рисунке (рис. 8).

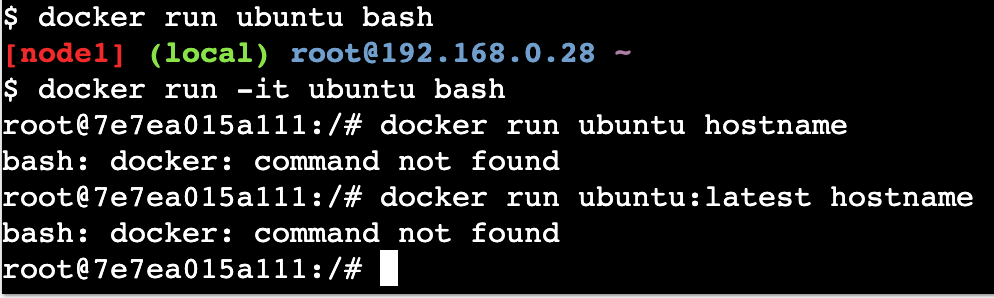


Рисунок 8 – Запуск bash

## **1.3 Работа с портами**

Загрузка образа python с помощью команды docker pull python и запуск встроенного в Python веб-сервера из корня контейнера (рис. 9).

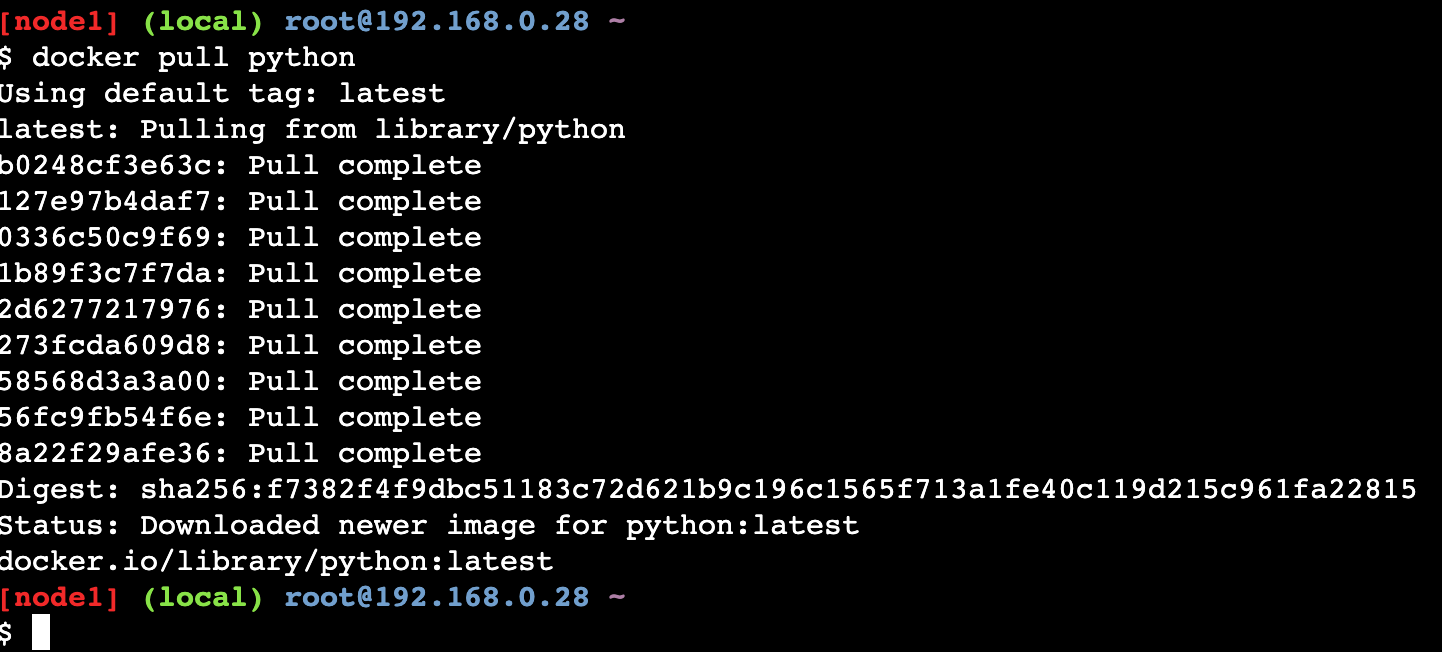


Рисунок 9 – Загрузка образа python

После запуска при попытке входа на данный адрес ничего не будет видно, потому что не проброшен порт. Для проброса портов используется флаг -p hostPort:containerPort Добавим его, чтобы пробросить порт 8000:docker run -it -p8000:8000 python python -m http.server — теперь по адресу http://0.0.0.0:8000/ (если не открывается на Windows, то вместо 0.0.0.0 нужно указать localhost) (рис. 10) открывается содержимое корневой директории в контейнере (рис. 11).

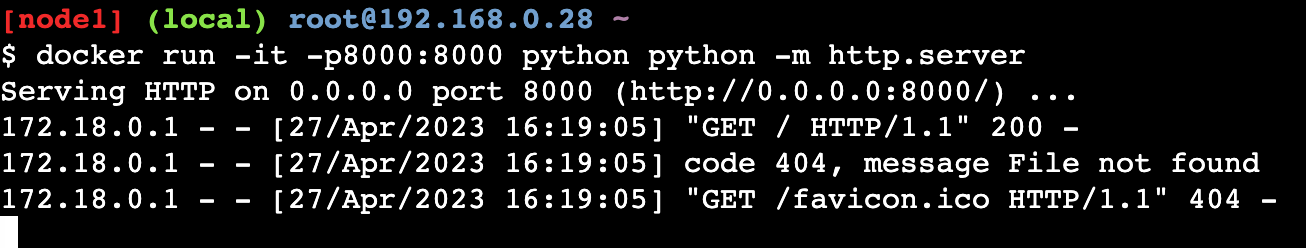


Рисунок 10 – Запуск сервера с проброской порта

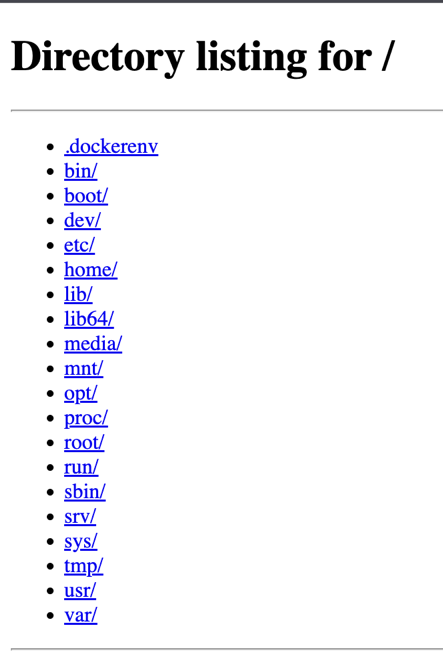


Рисунок 11 – Просмотр содержимого каталога

## **1.4 Именованные контейнеры, остановка и удаление**

Для запуска контейнера в фоновом режиме используется флаг –d. Для задания имени используется флаг –name. Для проверки запуска используется команда docker ps | grep servername. Для просмотра истории действий docker logs servername. Для остановки – docker stop servername. Для удаления из списка – docker rm servername.

Демонстрация выполнения команд представлена на рисунке (рис. 11 – 11.1).

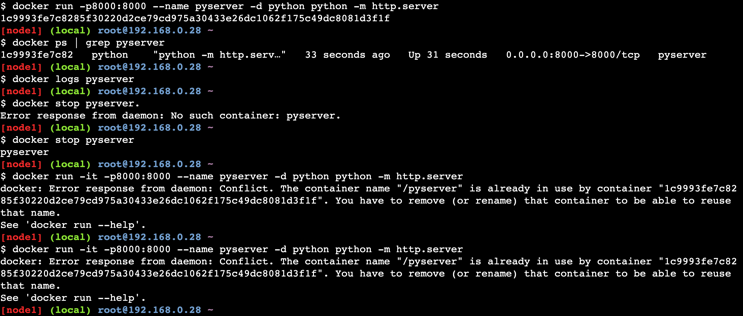


Рисунок 11 – Запуск контейнера в фоновом режиме с установленным именем

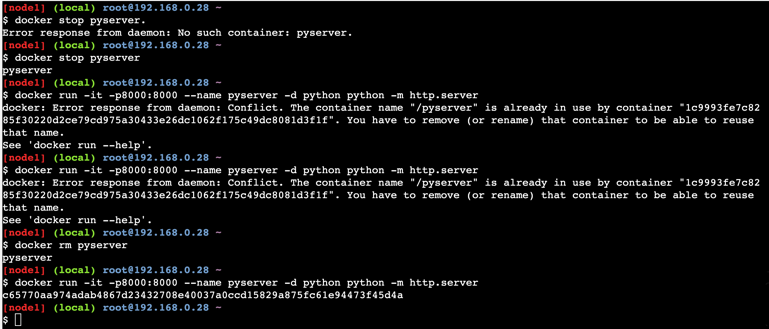


Рисунок 11.1 – Запуск контейнера в фоновом режиме с установленным именем

## **1.5 Постоянное хранение данных**

Далее на рисунках (рис. 12-13.1) демонстрируется запуск контейнера, в котором веб-сервер будет отдавать содержимое директории /mnt, создание файла hi.txt, просмотр содержимого и остановка сервера.

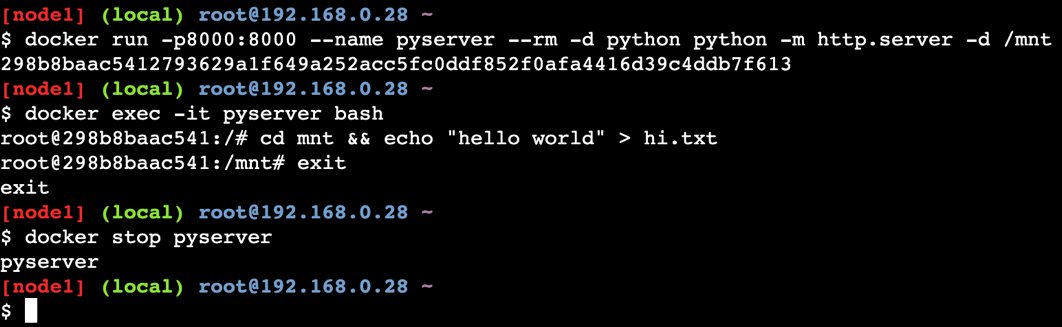


Рисунок 12 – Запуск контейнера, создание файла и остановка контейнера

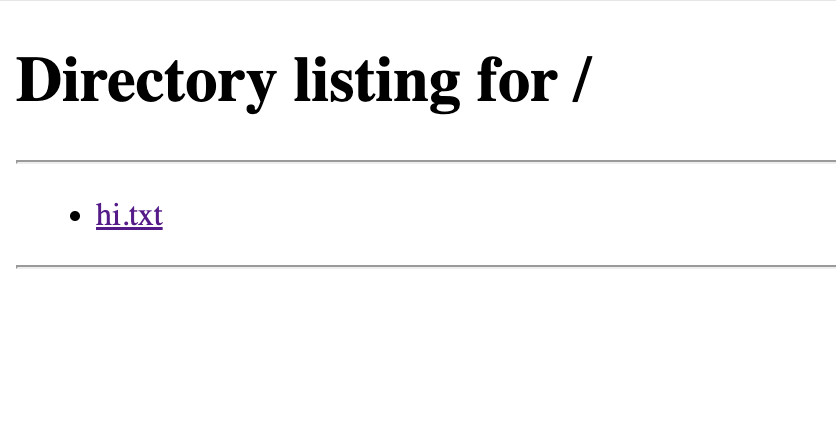


Рисунок 13 – Просмотр содержимого контейнера



Рисунок 13.1 – Просмотр содержимого контейнера

### **1.5.1 Тома**

После повторного запуска контейнера и просмотра содержимого мы обнаружим, что файл hi.txt пропал.

Для сохранения данных есть несколько способов. Первый – создание отдельного тома с помощью ключа –v. В таком случае даже после удаления контейнера созданные файлы будут сохраняться (рис. 14).



Рисунок 14 – Монтирование тома

### **1.5.2 Монтирование директорий и файлов**

Второй способ – монтирование директорий и файлов. Для этого создается директория и файлы и монтируются при запуске контейнера (рис. 15).

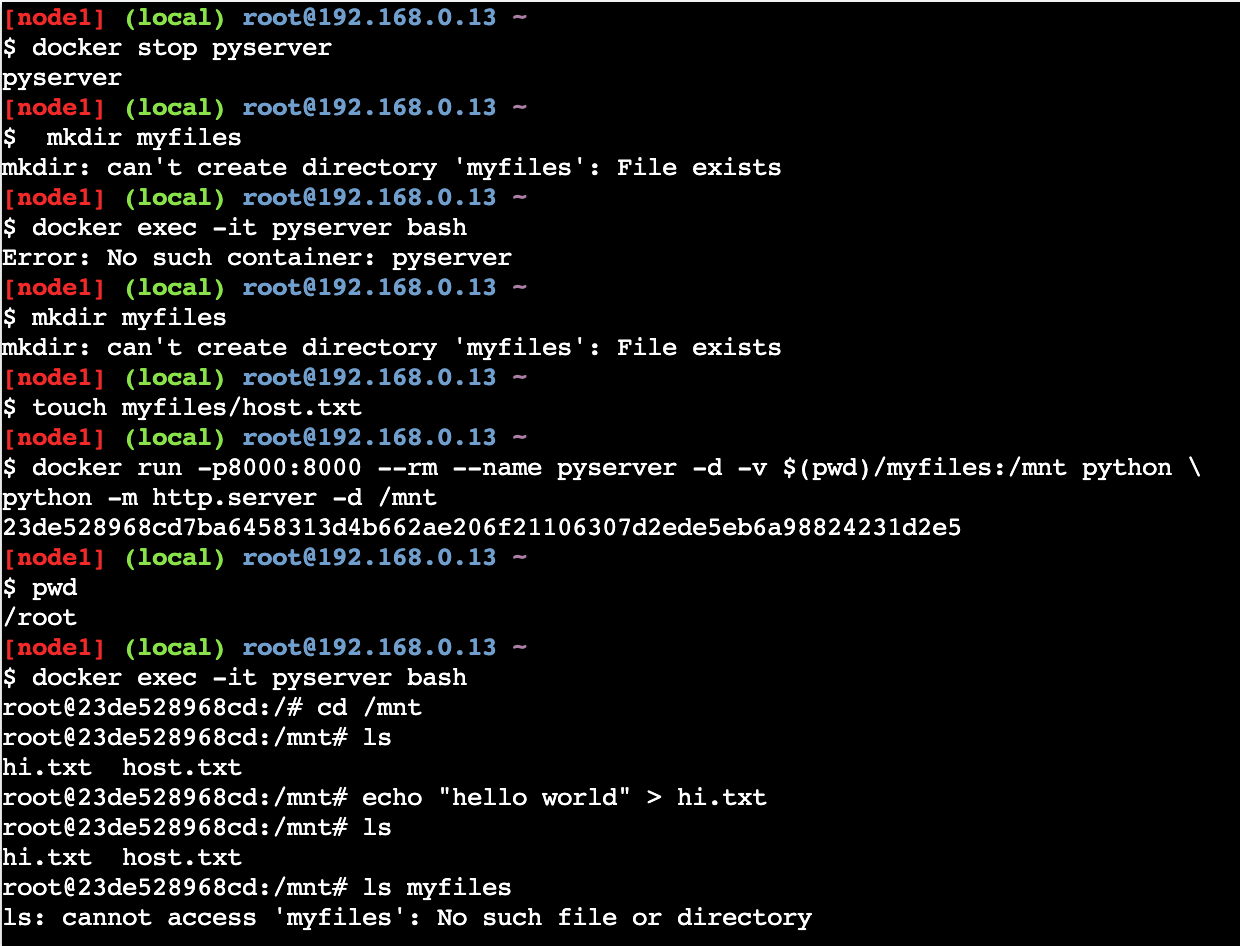


Рисунок 15 – Монтирование директорий и файлов

## **1.6 Переменные окружения**

Передача переменных окружения внутрь контейнера делается с помощью ключа -e (рис. 16).



Рисунок 16 – Передача переменных окружения

## **1.7 Dockerfile**

Создание докерфайла (рис. 17).

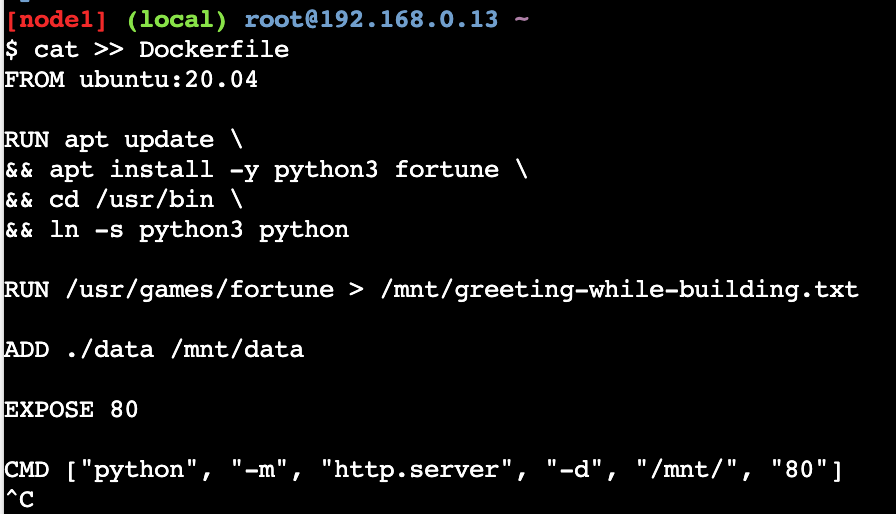


Рисунок 17 – Создание докерфайла

Сборка образа, запуск и просмотр содержимого контейнера представлены на рисунках (рис. 18-19).

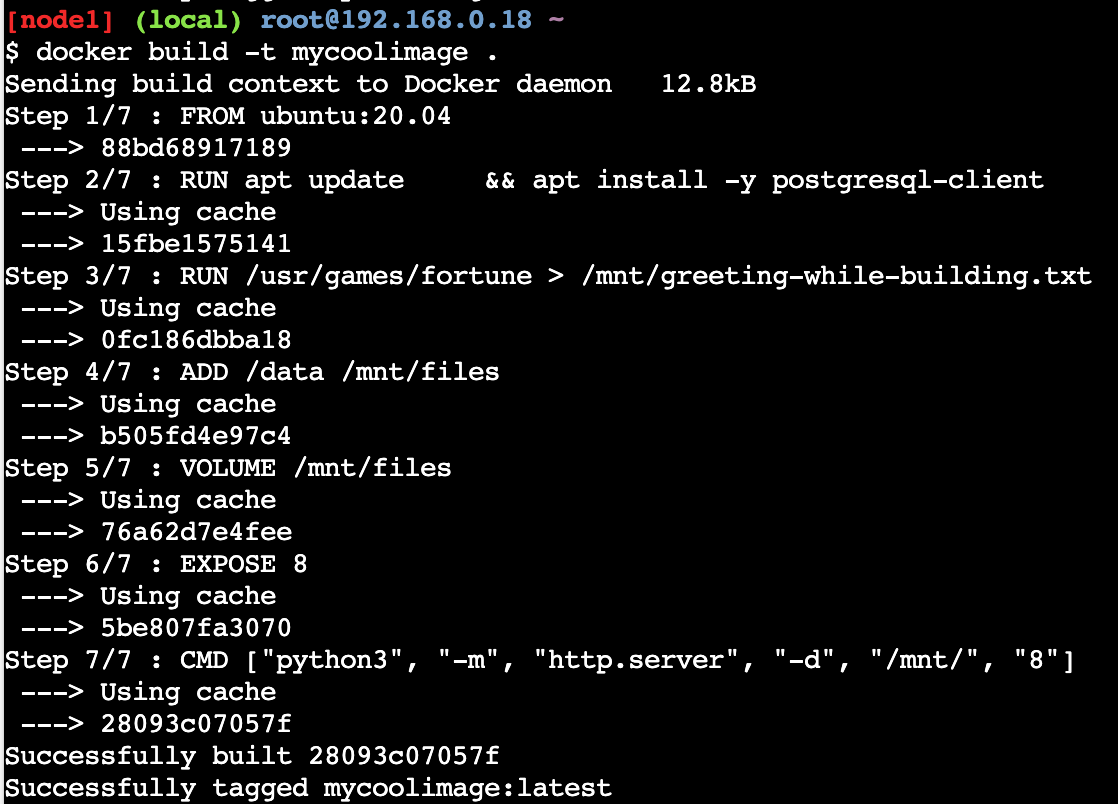


Рисунок 18 – Сборка образа и запуск контейнера

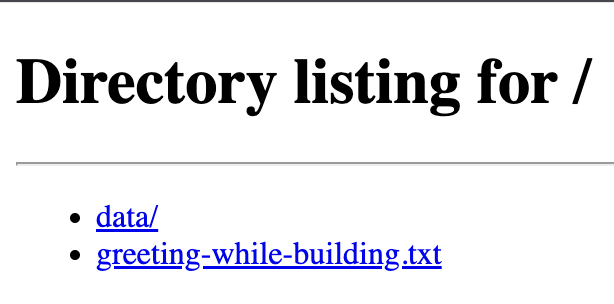


Рисунок 19 – Просмотр содержимого контейнера

## **Индивидуальные задания**

Персональный вариант: 2.

Создадим директорию data, содержащий файл student.txt (Рисунок 20-21).

Рисунок 20 – Новый директорий и файлы



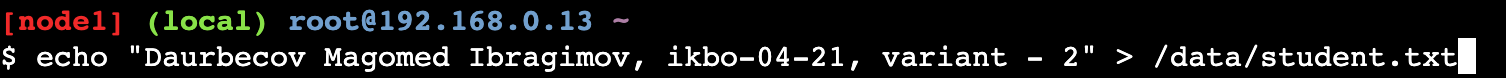
Рисунок 20 – Создание директорию data

Рисунок 21 – Создание и заполнение файла student.txt

Выполнение индивидуального задания (рис. 22-25).

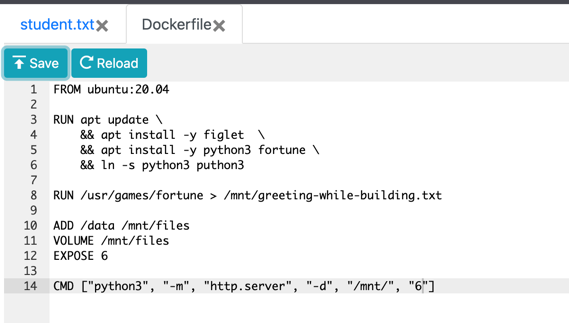


Рисунок 22 – Создание dockerfile по заданию

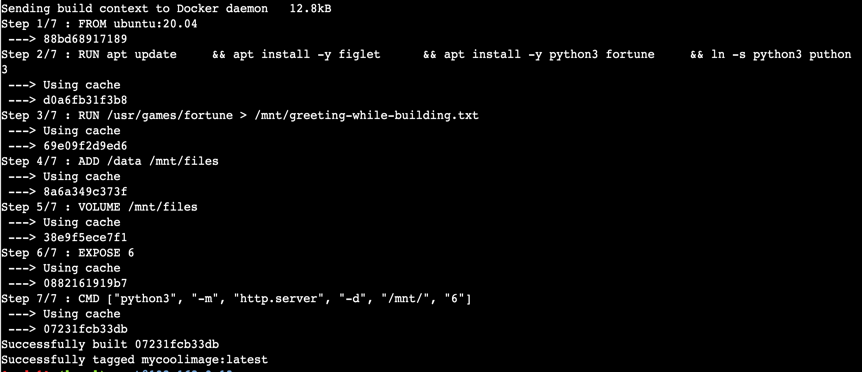
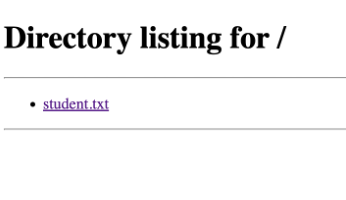


Рисунок 23 – Сборка образа



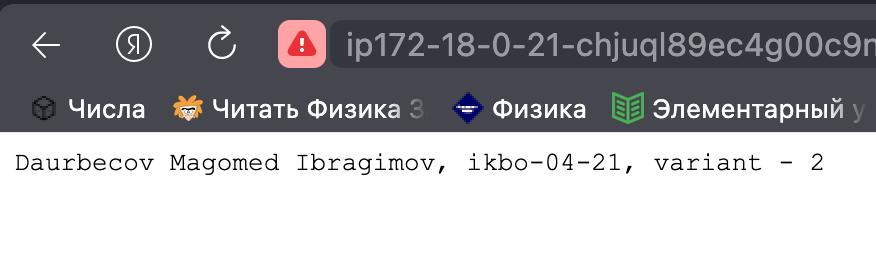
Рисунок 24 – Просмотр содержимого контейнера

Рисунок 25 – Просмотр содержимого файла

Вывод

В ходе работы были получены навыки работы с Docker:  
• Установка Docker;  
• Скачивание и использование образов;  
• Работа с контейнерами;  
• Работа с переменными окружениями;  
• Создание Dockerfile.

Список информационных источников

1. Методические указания по выполнению практической работы №3: Docke, МИРЭА – Российский технологический университет, 2023.
2. Docker – [Электронный ресурс] URL: [https://www.docker.com/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fwww.docker.com%2F)