

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №3

по дисциплине «Технологии разработки программных приложений»

**Тема практической работы:** «Docker»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группы ИКБО-20-19 | Московка А. А. |
| **Проверил:** | ассистент Иванов М. Е. |

2021 г.

**Содержание**

[Практическая работа №3 «Docker» 3](#_Toc70361841)

[1. Образы 3](#_Toc70361842)

[2. Изоляция 4](#_Toc70361843)

[3. Работа с портами 6](#_Toc70361844)

[4. Именованные контейнеры, остановка и удаление 8](#_Toc70361845)

[5. Постоянное хранение данных 10](#_Toc70361846)

[5.1. Тома 12](#_Toc70361847)

[5.2. Монтирование директорий и файлов 13](#_Toc70361848)

[6. Переменные окружения 15](#_Toc70361849)

[7. Dockerfile 16](#_Toc70361850)

[8. Индивидуальные задания 19](#_Toc70361851)

[Вывод 20](#_Toc70361852)

# Практическая работа №3 «Docker»

**Вариант №5**

1. Образы

Посмотрите на имеющиеся образы: docker images.

Загрузите образ: docker pull ubuntu — будет загружен образ ubuntu:latest — последняя доступная версия. Для загрузки конкретной версии, нужно указать тег, например, 12.04: docker pull ubuntu:12.04.

Посмотрите на имеющиеся образы ещё раз: docker images — должны появиться новые загруженные образы. Посмотрите список контейнеров, выполнив команду: docker ps -a

**Выполнение задания:**

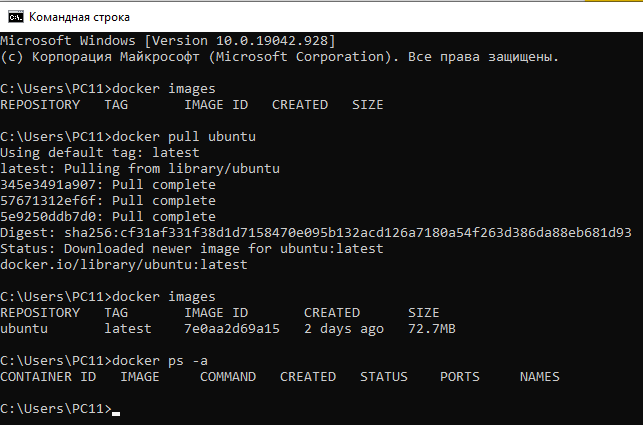


Рис. 1 – Скриншот выполнения первого задания

1. Изоляция

Посмотрим информацию о хостовой системе, выполнив команду hostname. Выполните её ещё один раз. *Вопрос: одинаковый ли результат получился при разных запусках?*

Попробуем выполнить то же самое в контейнерах. Выполните два раза команду docker run ubuntu hostname. *Вопрос: Одинаковый ли результат получился при разных запусках?*

В случае запуска команды в контейнерах, ответ будет немного отличаться, будет разный hostname. Так происходит, потому что из одного образа ubuntu были запущены два изолированных контейнера, поэтому у них и был разный hostname.

Заново выполните docker ps -a — там должны появиться запущенные ранее контейнеры.

Запуск контейнеров производится командой:

docker run --флаги --докера имя\_контейнера команда для запуска -и --флаги --запуска --программы.

Запустите bash в контейнере: docker run ubuntu bash. Ничего не произошло. Это не баг. Интерактивные оболочки выйдут после выполнения любых скриптовых команд, если только они не будут запущены в интерактивном терминале — поэтому для того, чтобы этот пример не завершился, вам нужно добавить флаги -i -t или сгруппированно -it: docker run -it ubuntu bash.

Выполняя запуск контейнера, указывая образ ubuntu, неявно указывался образ ubuntu:latest. Следовательно, следующие команды равнозначны:

• docker run ubuntu hostname

• docker run ubuntu:latest hostname

Если бы мы хотели запустить ubuntu:12.04, то нужно было бы выполнить команду docker run ubuntu:12.04 hostname

**Выполнение задания:**

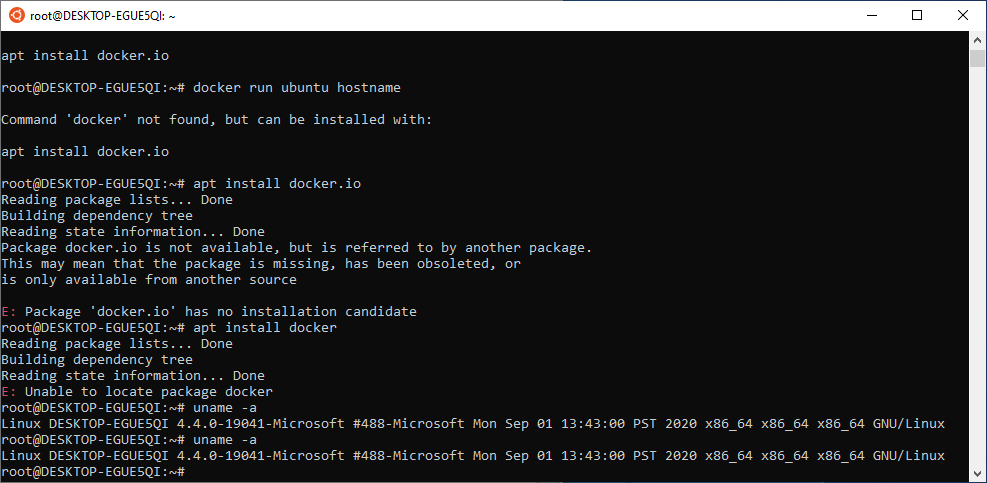


Рис. 2 – Скриншот выполнения второго задания (п. 1)

*Вопрос: одинаковый ли результат получился при разных запусках?*

*Ответ: да, результат одинаковый при двух последовательных запусках.*

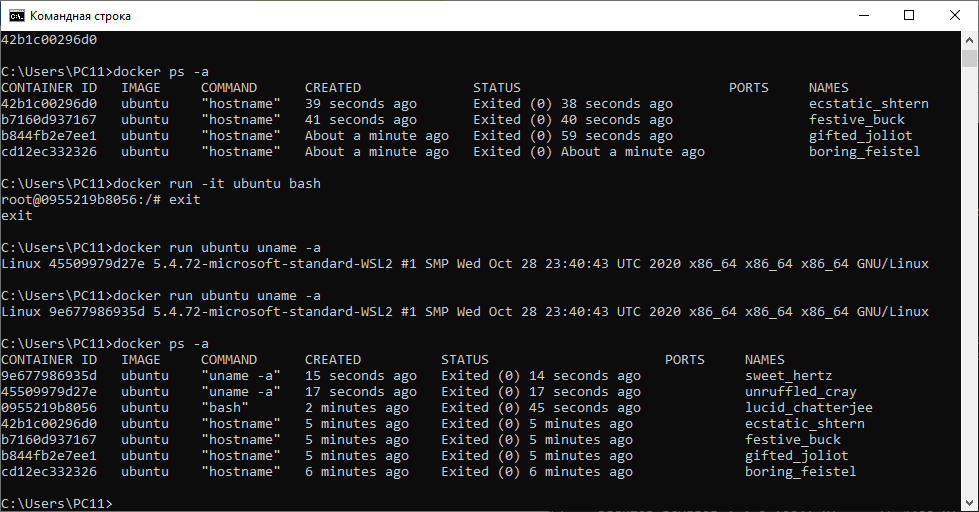


Рис. 3 – Скриншот выполнения второго задания (п. 2)

*Вопрос: Одинаковый ли результат получился при разных запусках?*

*Ответ: нет, на этот раз результаты различаются, потому что запускались различные контейнеры для выполнения команды.*

1. Работа с портами

Для начала, загрузите образ python командой docker pull python.

В качестве примера, запустите встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера. docker run -it python python -m http.server

При запуске пишется, что сервер доступен по адресу http://0.0.0.0:8000/. Однако, если открыть этот адрес, то ничего не будет видно, потому что порты не проброшены. Завершите работу веб-сервера, нажав комбинацию клавиш Ctrl+C.

Для проброса портов используется флаг -p hostPort:containerPort

Добавьте 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) открывается содержимое корневой директории в контейнере.

Для того, чтобы доступный в контейнере на порту 8000 веб-сайт в хостовой системе открывался на порту 8888, необходимо указать флаг -p8888:8000: docker run -it -p8888:8000 python python -m http.server.

Завершите работу веб-сервера, нажав комбинацию клавиш Ctrl+C.

**Выполнение задания:**

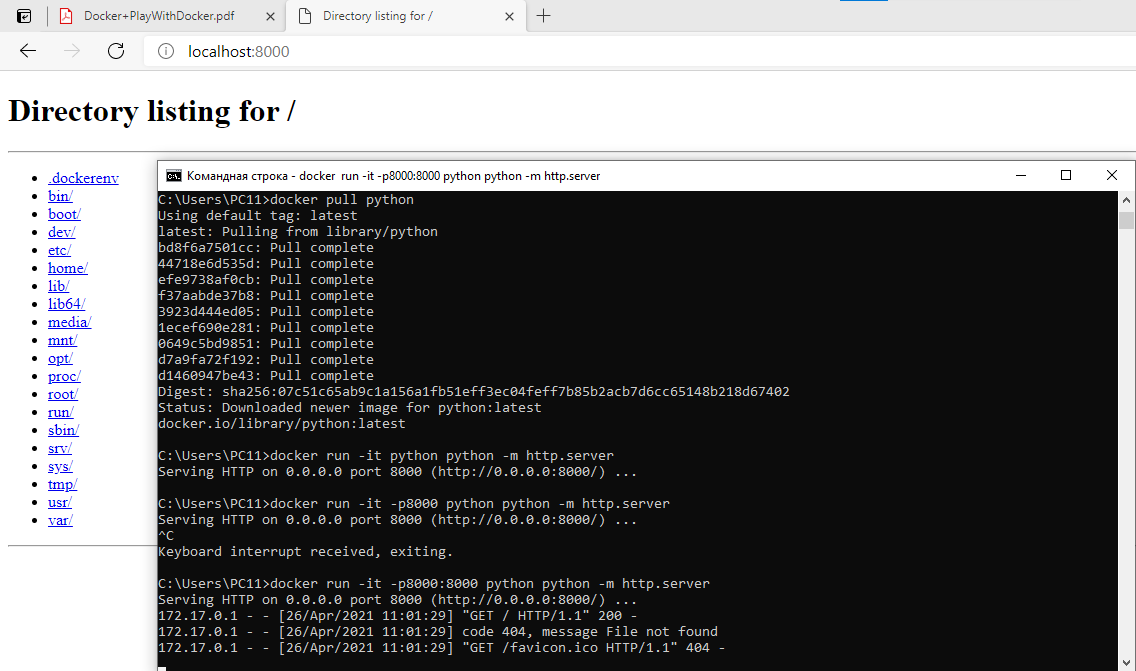


Рис. 4 – Скриншот выполнения третьего задания

1. Именованные контейнеры, остановка и удаление

Запустите контейнер: docker run -it -p8000:8000 python python -m http.server. Нажмите Ctrl+C — выполнение завершится. Для того, чтобы запустить контейнер в фоне, нужно добавить флаг -d/--detach.

Также определим имя контейнеру, добавив флаг --name.

docker run -p8000:8000 --name pyserver -d python python -m http.server

Убедитесь, что контейнер всё ещё запущен: docker ps | grep pyserver — вывод команды не должен быть пустым. Для просмотра логов контейнера, воспользуйтесь командой docker logs pyserver.

Для того, чтобы остановить выполнение контейнера, существует команда docker stop pyserver. Однако, если снова попробовать запустить командой docker run -it -p8000:8000 --name pyserver -d python python -m http.server, то возникнет ошибка: контейнер с таким именем существует. Его нужно удалить docker rm pyserver.

Для остановки и удаления контейнера можно воспользоваться командой docker rm -f pyserver вместо выполнения двух отдельных команд stop и rm. После удаления контейнер с таким именем можно будет создать заново.

Для того, чтобы контейнер удалялся после завершения работы, нужно указать флаг --rm при его запуске — далее в работе мы будем использовать данный флаг:

docker run --rm -p8000:8000 --name pyserver -d python python -m http.server

**Выполнение задания:**

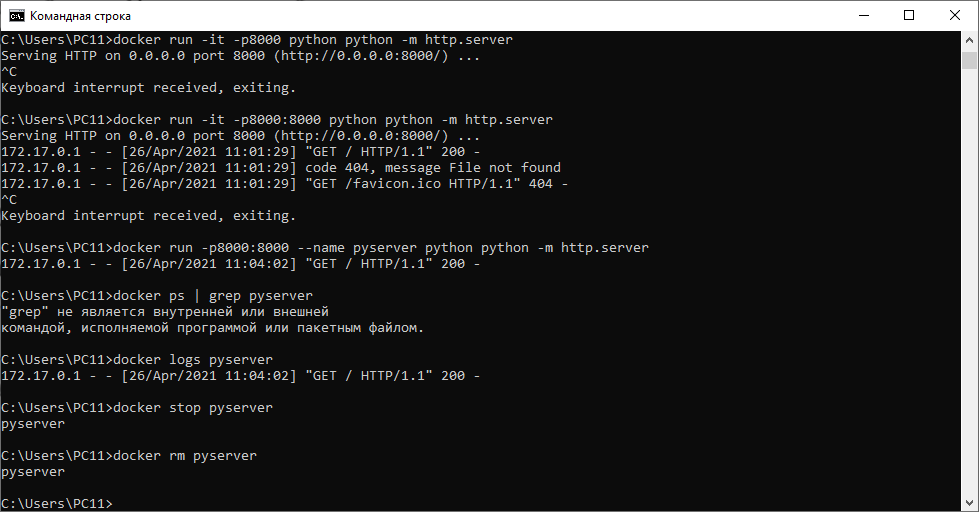
****

Рис. 5 – Скриншот выполнения четвертого задания

1. Постоянное хранение данных

Запустите контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt:

docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt, где -d /mnt указывает модулю http.server какая директория будет корневой для отображения.

*Вопрос: Что значат остальные флаги запуска?* *Где здесь команда, которая выполнится в контейнере?*

Для того, чтобы попасть в уже запущенный контейнер, существует команда docker exec -it pyserver bash — вы попадёте в оболочку bash в контейнере. Попав в контейнер, выполните команду cd mnt && echo "hello world" > hi.txt, а затем выйдите из контейнера, введя команду exit или нажав комбинацию клавиш Ctrl+D.

Если открыть http://0.0.0.0:8000/, там будет доступен файл hi.txt.

Остановим контейнер: docker stop pyserver, а затем снова запустим:

docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt.

Как мы видим, файл hi.txt пропал — это неудивительно, ведь мы запустили другой контейнер, а старый был удалён после завершения работы (флаг --rm). Остановим контейнер: docker stop pyserver.

Для того, чтобы не терялись какие-то данные (например, если запущен контейнер с СУБД, то, чтобы не терялись данные из неё) существует механизм монтирования.

**Выполнение задания:**

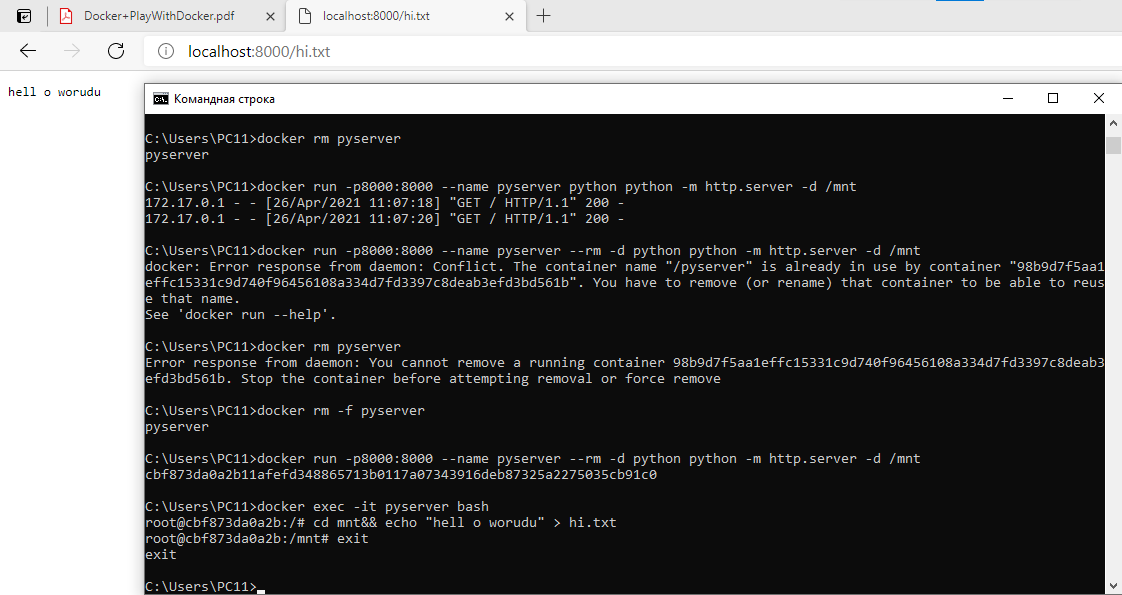


Рис. 6 – Скриншот выполнения пятого задания

*Вопрос: Что значат остальные флаги запуска?* *Где здесь команда, которая выполнится в контейнере?*

*Ответ:*

*-it - используется для запуска контейнера в терминале*

*-p:8000:8000 - задаем порты для входа в контейнер локально*

*--name - задаем имя нашему контейнеру*

*pyserver - название нашего контейнера*

*-rm - как только работа с контейнером будет закончена, он удалится*

*-d - флаг для перехода*

*python - запускаем python внутри контейнера*

*-m - флаг запуска локального сервера*

*http.server - запуск локального сервера*

*/mnt - указываем директорию, которую покажем на локальном сервере*

*В контейнере выполнятся все команды после названия образа контейнера (в данном случае это «python»).*

* 1. Тома

Первый способ — это создать отдельный том с помощью ключа -v myvolume:/mnt, где myvolume — название тома, /mnt — директория в контейнере, где будут доступны данные.

Попробуйте снова создать контейнер, но уже с примонтированным томом:



Затем, если создать файл (выполнить docker exec -it pyserver bash и внутри контейнера выполнить cd mnt && echo "hello world" > hi.txt), то даже после удаления контейнера данные в этом томе будут сохранены.

Чтобы узнать где хранятся данные, выполните команду

docker inspect -f "{{json .Mounts }}" pyserver, в поле Source будет храниться путь до тома на хостовой машине.

Для управления томами существует команда docker volume, ознакомиться с которой предлагается самостоятельно.

**Выполнение задания:**

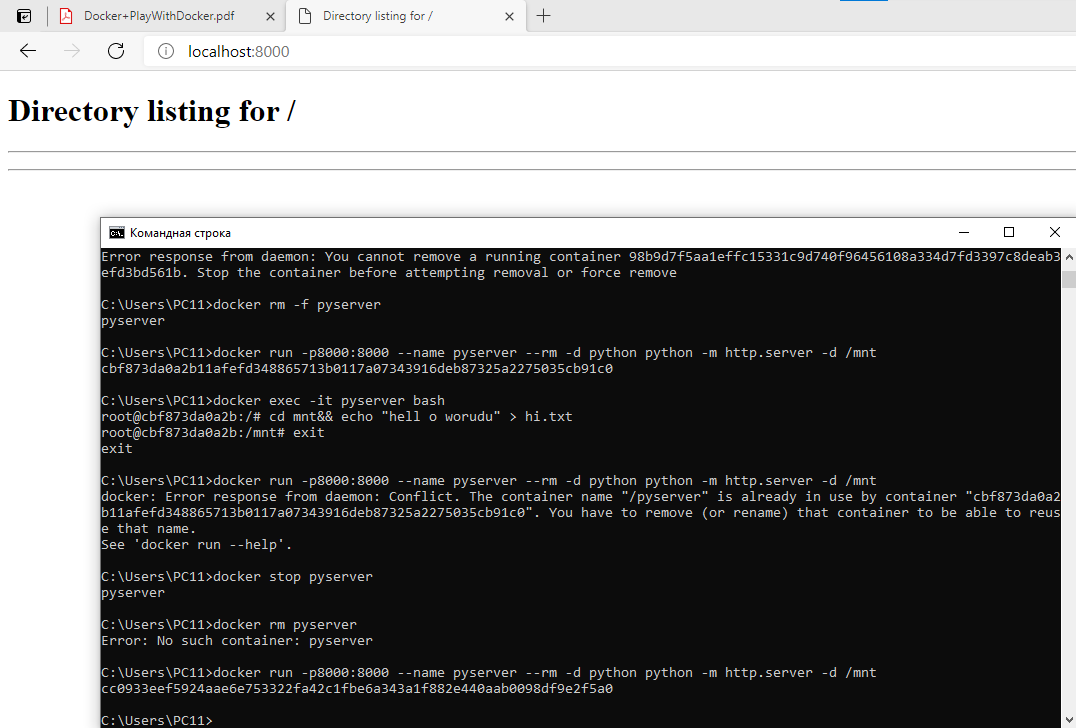
****

Рис.7 – Скриншот выполнения задания 5 пункт 1

* 1. Монтирование директорий и файлов

Сперва, остановите контейнер, созданный на предыдущем шаге: docker stop pyserver.

Иногда требуется пробросить в контейнер конфигурационный файл или отдельную директорию. Для этого используется монтирование директорий и файлов.

Создадим директорию и файлы, которые будем монтировать. Часть из них нам понадобится дальше: создайте директорию: mkdir myfiles, в ней создайте файл host.txt: touch myfiles/host.txt

Запустите контейнер:

docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \

python -m http.server -d /mnt

Команда pwd — выведет текущую директорию, например: /home/user/dome-directory, в итоге получился абсолютный путь до файла: /home/user/dome-directory/myfiles.

Обратный слеш (\) перед переводом строки экранирует символ перевода строки и позволяет написать одну команду в несколько строк.

Затем, зайдите в контейнер: docker exec -it pyserver bash, перейдите в директорию /mnt командой cd /mnt. Если вывести список файлов командой ls, то там будет файл host.txt, примонтированный вместе с директорией myfiles

Создайте файл echo "hello world" > hi.txt, а затем выйдите из контейнера: exit. Теперь на хостовой машине в директории myfiles/ появится файл hi.txt. Проверить можно командой ls myfiles.

Остановите контейнер: docker stop pyserver.

Для того, чтобы примонтировать один файл, нужно указать ключ -v, например:

-v $(pwd)/myfiles/host.txt:/mnt/new-name-of-host.txt – файлу в контейнере присвоится другое имя: new-name-of-host.txt.

Если на Windows возникают ошибки при монтировании, убедитесь, что вы используете bash, а не cmd.exe.

**Выполнение задания:**

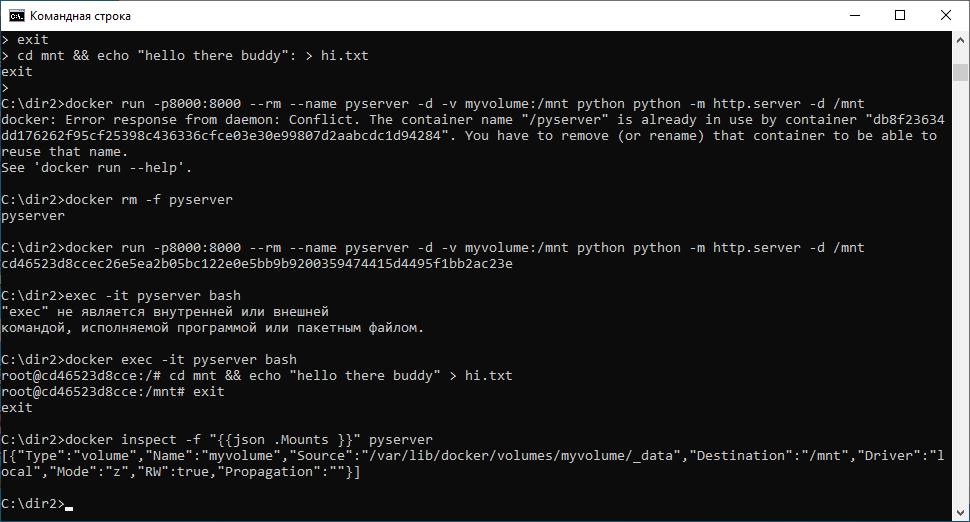


Рис. 8 – Скриншот выполнения задания 5 пункт 2

1. Переменные окружения

Для передачи переменных окружения внутрь контейнера используется ключ -e. Например, чтобы передать в контейнер переменную окружения MIREA со значением «ONE LOVE», нужно добавить ключ -e MIREA="ONE LOVE".

Проверьте, выведя все переменные окружения, определённые в контейнере с помощью утилиты env: docker run -it --rm -e MIREA="ONE LOVE" ubuntu env. Среди списка переменных будет и MIREA

**Выполнение задания:**

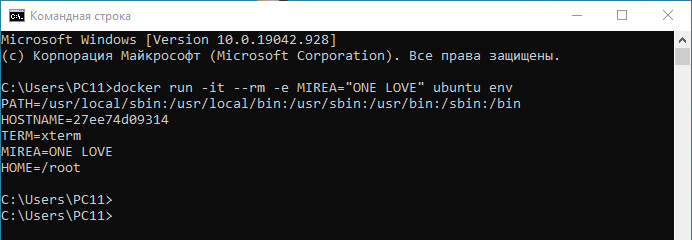
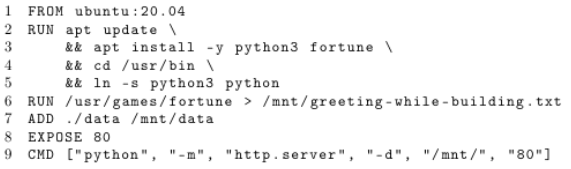


Рис. 9 – Скриншот выполнения шестого задания

1. Dockerfile

Соберите образ, в который будут установлены дополнительные пакеты, примонтируйте директорию и установите команду запуска. Для этого создаётся файл Dockerfile (без расширения).



Будьте внимательны при копировании, могут скопироваться неправильные минусы и лишние пробелы.

В строке (1) указывается базовый образ, на основе которого будет строиться новый образ.

В строках (2-5) указана команда, которая выполнится в процессе сборки. На самом деле, там выполняются несколько команд, соединённых && для того, чтобы создавать меньше слоёв в образе.

В строках (6) тоже указана команда, которая сгенерирует случайную цитату и перенаправит вывод в файл /mnt/greeting-while-building.txt. Файл будет сгенерирован во время сборки образа.

В строке (7) копируется всё содержимое директории ./data хостовой машины в директорию /mnt, которая будет доступна в контейнере.

В строке (8) указывается, какой порт у контейнера будет открыт.

В строке (9) указывается команда, которая будет выполнена при запуске, где 80 — порт, который будет слушать веб-сервер.

Соберите образ с тегом mycoolimage с помощью команды docker build -t mycoolimage . Точка в конце указывает на текущую директорию, где лежит Dockerfile.

Запуск производится командой docker run --rm -it -p8099:80 mycoolimage, где порт 8099 — порт на хостовой машине.

Хорошая статья с описанием Dockerfile на Хабре: Изучаем Docker, часть 3: файлы Dockerfile.

**Выполнение задания:**

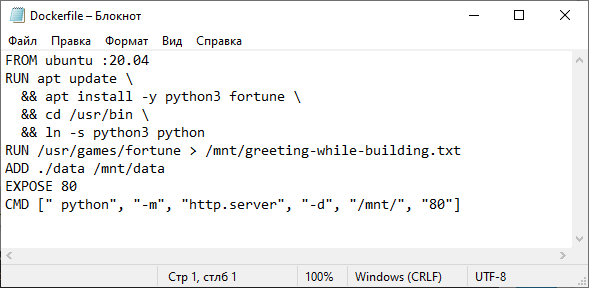


Рис. 10 – Скриншот содержимого файла Dockerfile

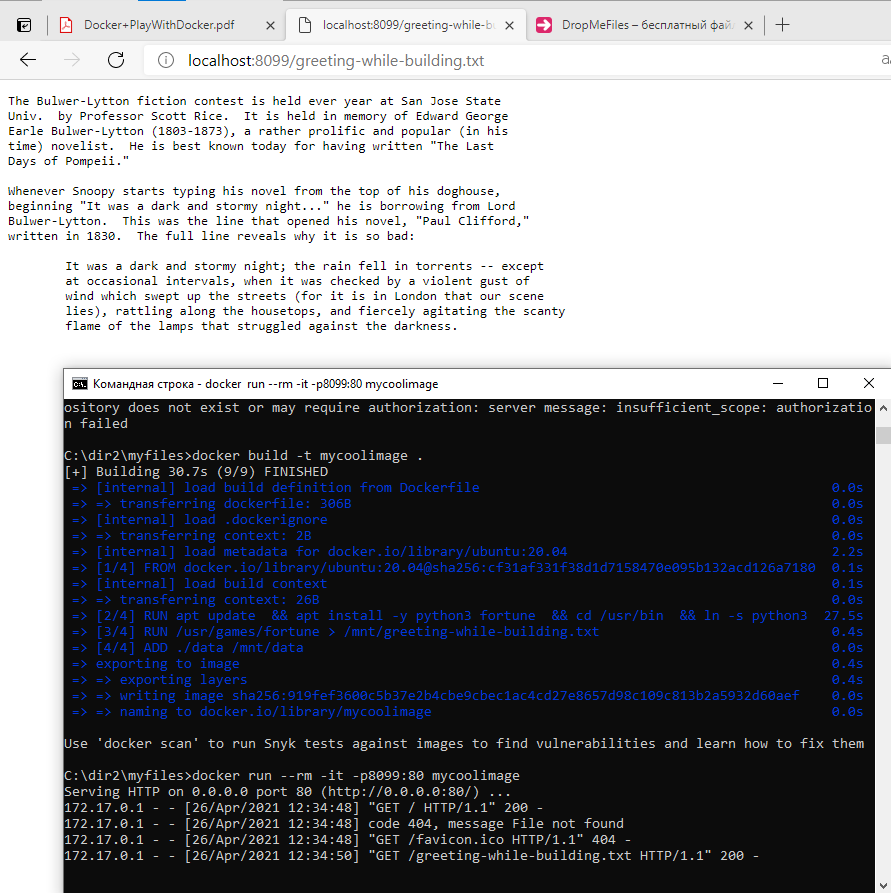


Рис. 11 – Скриншот успешного запуска образа mycoolimage

1. Индивидуальные задания

Написать Dockerfile, собрать образ, запустить контейнер (и записать команду для его запуска).

Для монтирования создайте директорию data и в ней файл student.txt, содержащий ФИО, название группы и номер варианта.

Для установки пакетов использовать команду apt install -y название-пакета. В качестве примера можно использовать Dockerfile из раздела 7.

Чётные варианты:

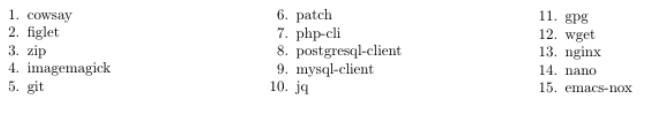
* необходимо использовать базовый образ ubuntu:20.10
* примонтировать файл data/student.txt как /mnt/files/student.txt в контейнере.

Нечётные варианты:

* необходимо использовать базовый образ ubuntu:20.04
* примонтировать директорию data в директорию /mnt/files/ в контейнере.

Запустить веб-сервер, отображающий содержимое /mnt/files, в хостовой системе должен открываться на порту (8800 + номер варианта). Например, для 22-го варианта это порт 8822

Установить пакет, согласно варианту:



**Выполнение задания:**

У меня 5 вариант, поэтому я устанавливаю образ «git» и запускаю сервер на порту 8805.

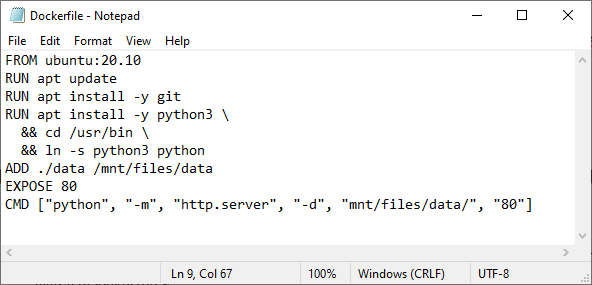


Рис. 12 – Скриншот содержимого файла Dockerfile индивидуального варианта

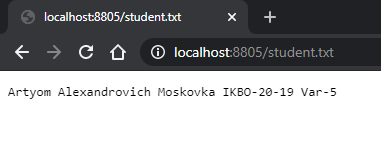


Рис. 13 – Скриншот содержимого файла student на локальной сервере

Вывод

В результате выполнения третьей практической работы были изучены основные команды для создания контейнеров, получены навыки запуска локального сервера посредством Docker и взаимодействия с файлами.