Содержание

[Практической работы №4: Docker 2](#_Toc162047308)

[Раздел 1. Образы. 2](#_Toc162047309)

[Раздел 2. Изоляция. 3](#_Toc162047310)

[Раздел 3. Работа с портами. 5](#_Toc162047311)

[Раздел 4. Именованные контейнеры, остановка и удаление. 7](#_Toc162047312)

[Раздел 5. Постоянное хранение данных. 8](#_Toc162047313)

[Раздел 5.1. Тома. 10](#_Toc162047314)

[Раздел. 5.2. Монтирование директорий и файлов 11](#_Toc162047315)

[Раздел 6. Переменные окружения. 12](#_Toc162047316)

[Раздел 7. Dockerfile. 13](#_Toc162047317)

[Раздел 8. Индивидуальные задания. 15](#_Toc162047318)

[Вывод. 17](#_Toc162047319)

Практической работы №4: Docker

**Цель работы:** научиться работать с программным обеспечением *Docker*. Выполнить все шаги из разделов 1-7, а также индивидуальное задание из раздела 8. Сделать вывод о проделанной работе, результаты выразить в отчете.

Раздел 1. Образы.

Посмотрите на имеющиеся образы: docker images.

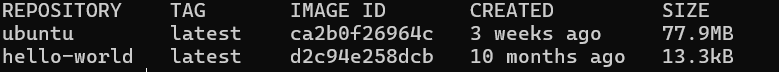


Рис. 1. Имеющиеся образы

Загрузите образ: docker pull ubuntu — будет загружен образ ubuntu:latest — последняя доступная

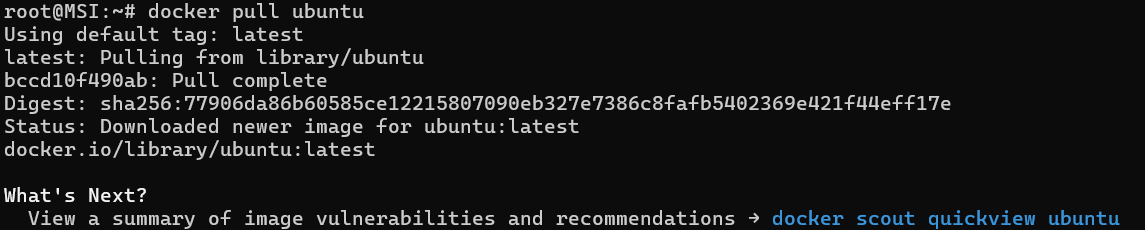


Рис. 2. Загрузка образа ubuntu:latest

Посмотрите на имеющиеся образы ещё раз: docker images — должны появиться новые загруженные образы.

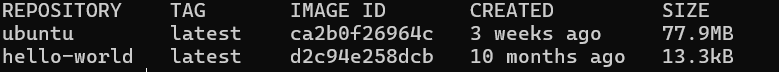


Рис. 3. Имеющиеся образы

Посмотрите список контейнеров, выполнив команду: docker ps -a



Рис. 4. Список контейнеров

Команда docker ps -a выводит только запущенные контейнеры.

Раздел 2. Изоляция.

Посмотрим информацию о хостовой системе, выполнив команду hostname. Выполните её ещё один раз.

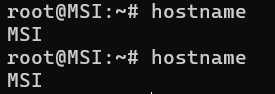


Рис. 5. Выполнение 2-х команд hostname

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

*Ответ:* да, имя хоста не изменяется.

Попробуем выполнить то же самое в контейнерах. Выполните два раза команду docker run ubuntu hostname.

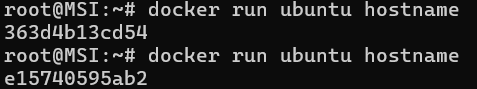


Рис. 6. Выполнение 2-х команд docker run ubuntu hostname

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

*Ответ:* нет, потому что было запушено 2 изолированных контейнера.

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

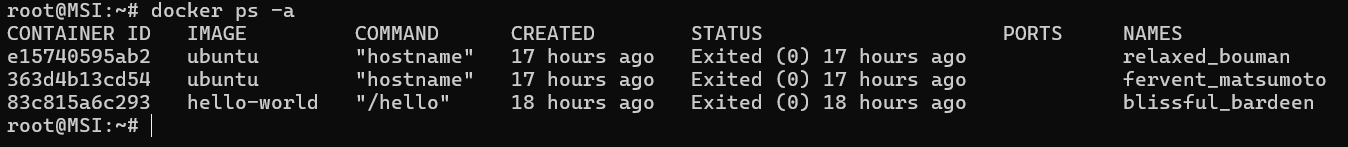


Рис. 6. Выполнение команды docker ps -a

Запустите bash в контейнере: docker run ubuntu bash.



Рис. 7. Вывод команды docker run ubuntu bash



Рис. 8. Вывод команды docker run -it ubuntu bash

Раздел 3. Работа с портами.

Для начала, загрузите образ python командой docker pull python.

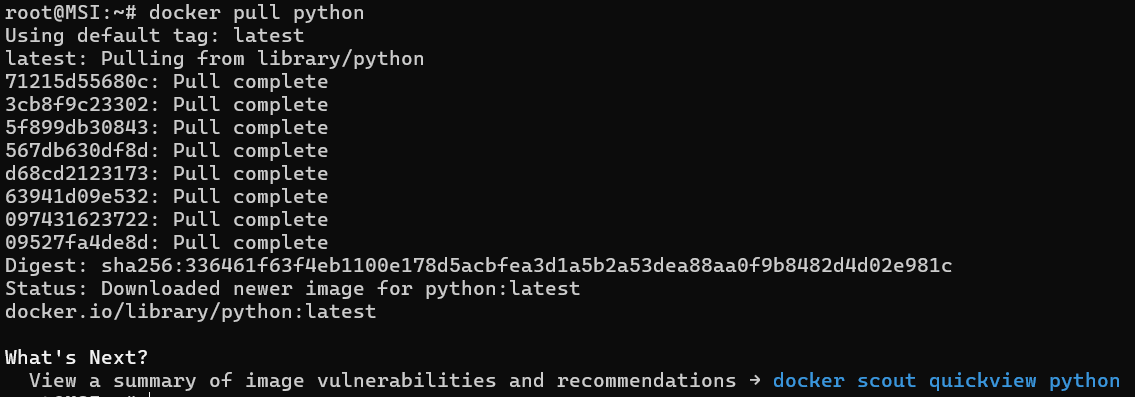


Рис. 9. Загрузка образа python

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.

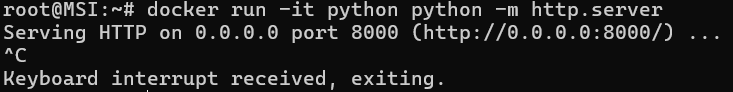


Рис. 10. Запуск сервера

Сервер не будет работаться, т.к. небыли проброшены порты. Сделаем это.

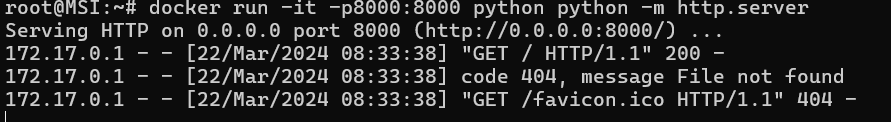


Рис. 11. Проброс портов и запуск сервера.

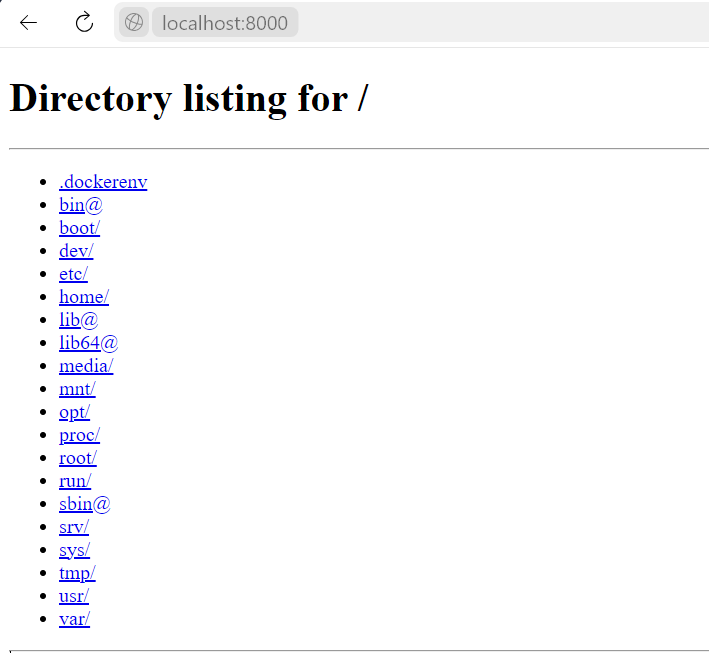


Рис. 12. Сайт с содержимым корневой директории в контейнере

Раздел 4. Именованные контейнеры, остановка и удаление.

Запустите контейнер в фоне при помощи команды docker run -p8000:8000 --name pyserver -d python python -m http.server. Убеждаемся, что контейнер все еще запущен при помощи команды: docker ps | grep pyserver. Для просмотра логов контейнера, воспользуйтесь командой docker logs pyserver.

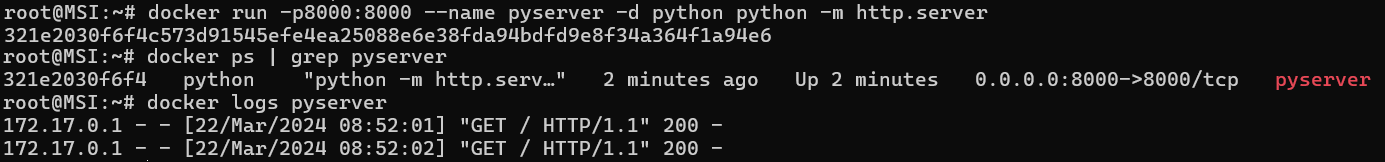


Рис. 13. Запуск и проверка работы контейнера

Останавливаем контейнер при помощи команды: docker stop pyserver. Удаляем контейнер при помощи команды: docker rm pyserver.

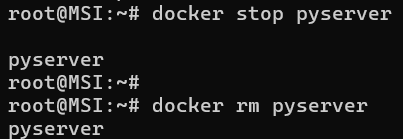


Рис. 14. Остановка и удаление контейнера

Раздел 5. 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?

*Ответ*: -p8000:8000: Этот флаг определяет проброс портов. --name pyserver: Этот флаг задает имя для контейнера. --rm: Этот флаг указывает Docker на удаление контейнера после его остановки. -d: Этот флаг запускает контейнер в фоновом режиме.

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.

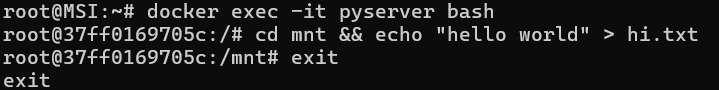


Рис. 15. Добавлении файла hi.txt с содержимого “hello world”

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

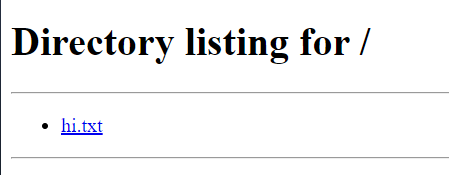


Рис. 16. Отображение файла hi.txt на веб-сайте

Остановим контейнер: docker stop pyserver, а затем снова запустим: docker run -p8000:8000 --name pyserver --rm -d python python -m http.server -d /mnt.

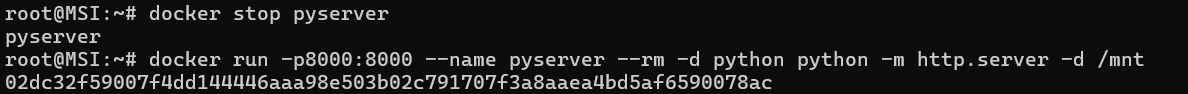


Рис. 17. Остановка и повторный запуск контейнера

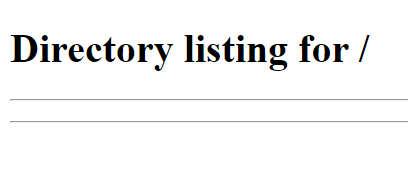


Рис. 18. Отображение веб-страницы после повторного запуска сервера

Как мы видим, файла hi.txt нет. Мы не видим этот файл из-за того, что все изменения происходили в границах первого контейнера.

Раздел 5.1. 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:

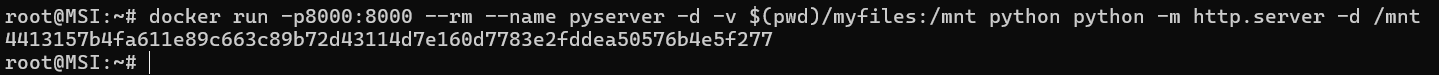


Рис. 19. Создание нового контейнера

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.



Рис. 20. Отображение проделанных команд

Раздел. 5.2. Монтирование директорий и файлов

Cоздайте директорию: mkdir myfiles, в ней создайте файл host.txt: touch myfiles/host.txt.

Запустте контейнер: docker run -p8000:8000 --rm --name pyserver -d -v $(pwd)/myfiles:/mnt python \ python -m http.server -d /mnt.

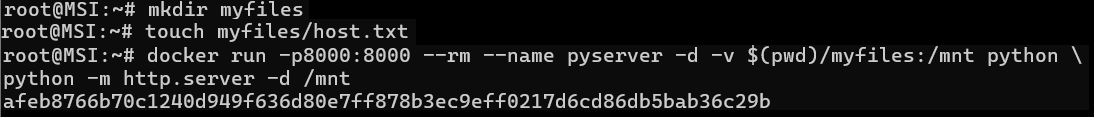


Рис. 21. Отображение проделанных команд

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

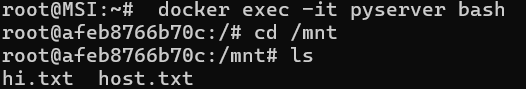


Рис. 22. Отображение проделанных команд

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.

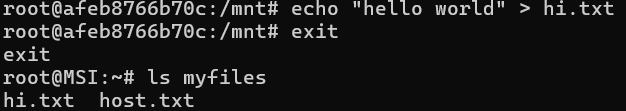


Рис. 23. Отображение проделанных команд

Раздел 6. Переменные окружения.

Для передачи переменных окружения внутрь контейнера используется ключ -e. Например, чтобы передать в контейнер переменную окружения MIREA во 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

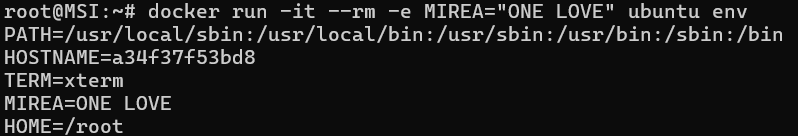


Рис. 24. Отображение проделанных команд

Раздел 7. Dockerfile.

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

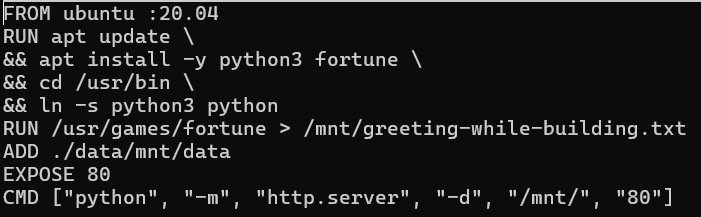


Рис. 25. Отображение Dockerfile

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

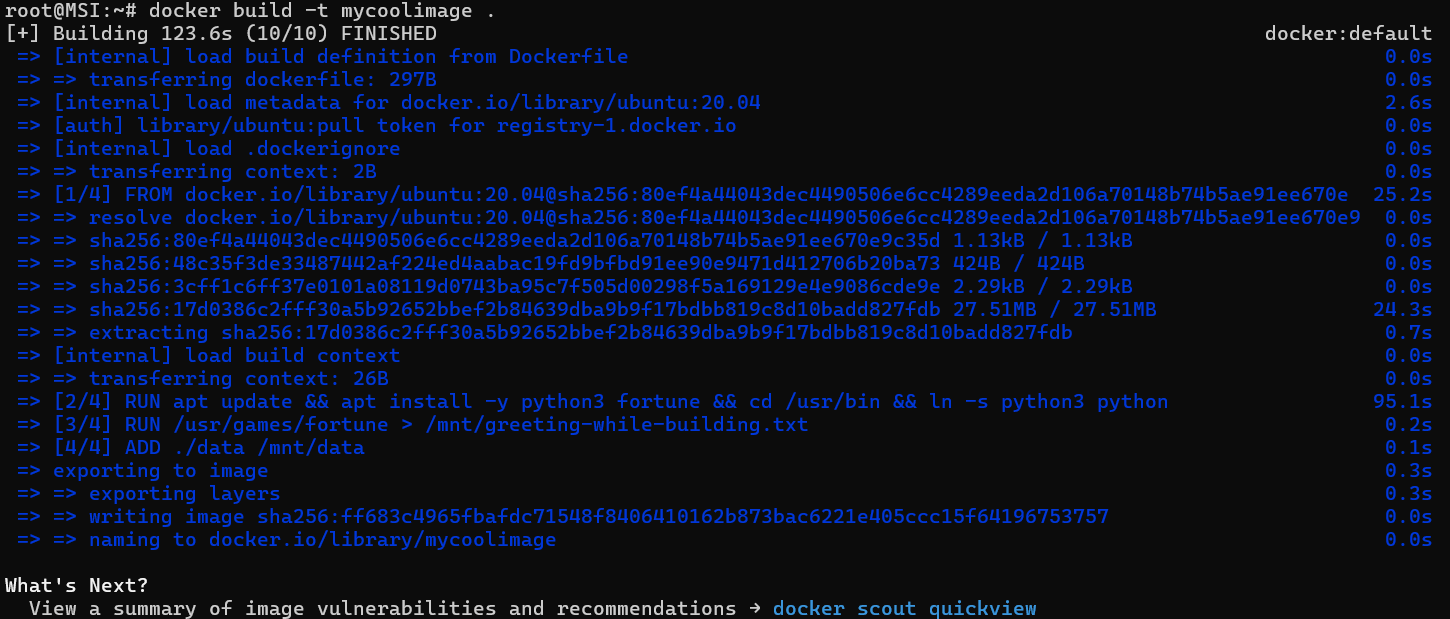


Рис. 26. Отображение проделанной команды

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

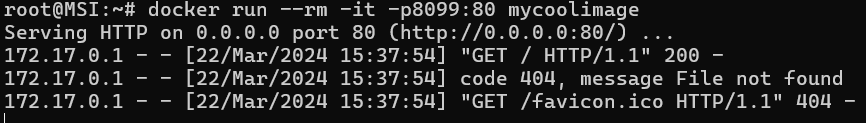


Рис. 27. Запуск контейнера

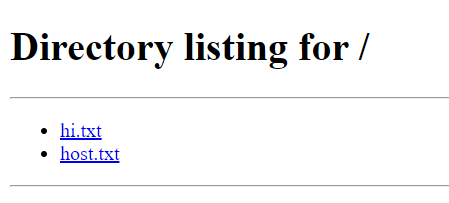


Рис. 28. Отображение веб-сайта

Раздел 8. Индивидуальные задания.

Вариант № 14. Номер по списку – 29.

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



Рис. 29. Файл student.txt

Для установки пакетов использовать команду apt install -y название-пакета.

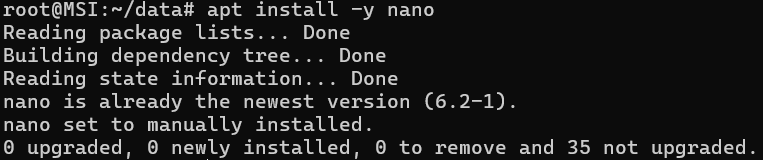


Рис. 30. Установка пакета nano согласно варианту

Необходимо использовать базовый образ ubuntu:20.10

Примонтировать файл data/student.txt как /mnt/files/student.txt 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.

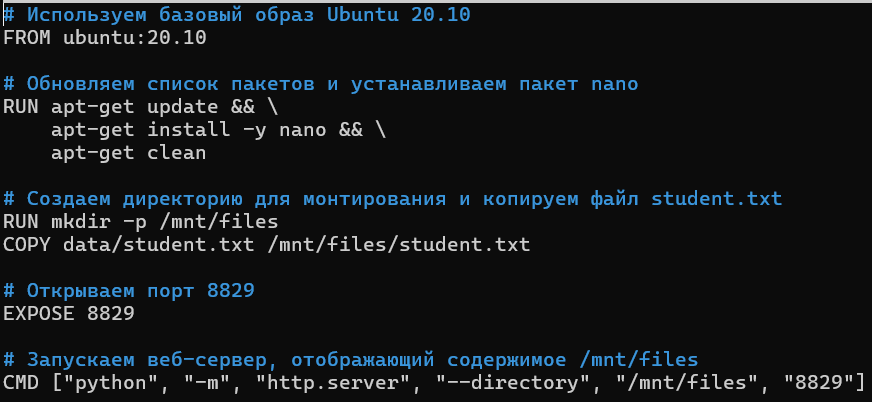


Рис. 31. Содержимое файла Dockerfile



Рис. 32. Команда запуска контейнера.

Вывод.

При выполнение работы были достигнуты цели научиться работать с программным обеспечением *Docker*. Выполнить все шаги из разделов 1-7, а также индивидуальное задание из раздела 8. Сделать вывод о проделанной работе, результаты выразить в отчете.