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

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации информационных технологий**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технология разработки программных приложений»

**Практическое задание № 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИНБО-08-22 Самойлов М.М. | (подпись) | |
| Старший преподаватель | *Мельников Д. А*. | (подпись) | |
| Отчет представлен | «23» марта 2024г. | |  | |

Москва 2024 г

**ОГЛАВЛЕНИЕ**

[Задание 3](#_Toc161056966)

[Выполнение заданий 4](#_Toc161056967)

[Вывод 7](#_Toc161056968)

# Выполнение заданий

1. **Образы**

На рисунке 1 отображен запуск команды docker images – получение всех имеющихся на устройстве образов

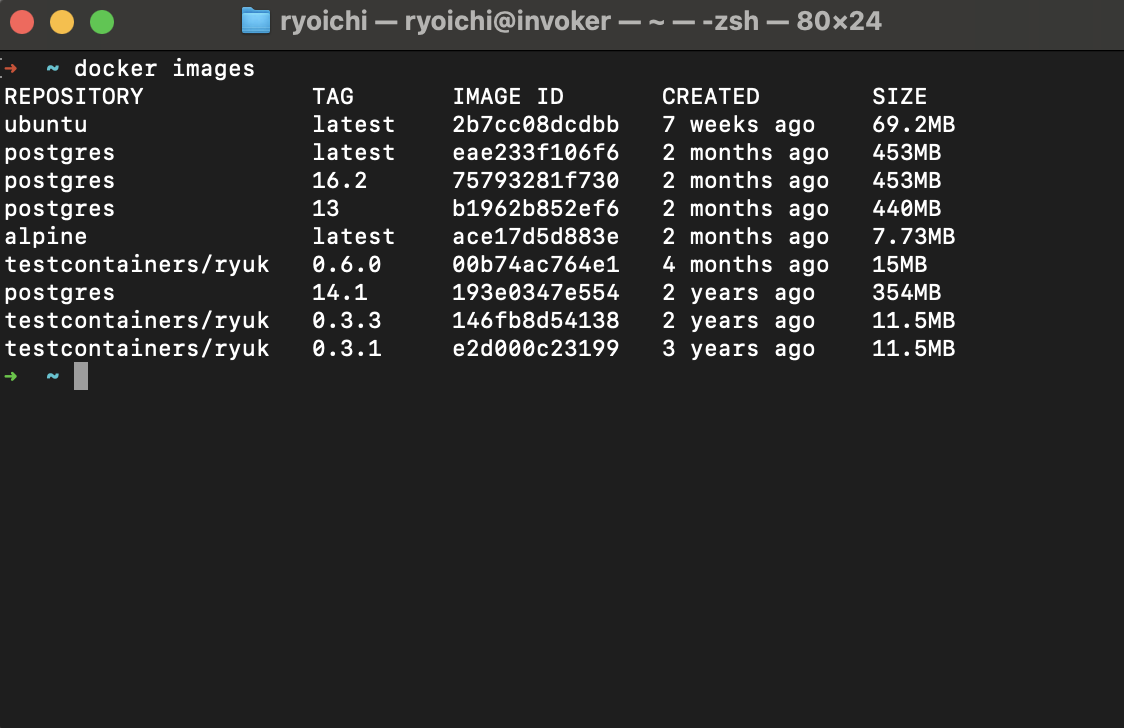


Рисунок 1 – Запуск команды docker images

На рисунке 2 отображен запуск команды docker pull ubuntu – скачивание образа ubuntu с докер хаба

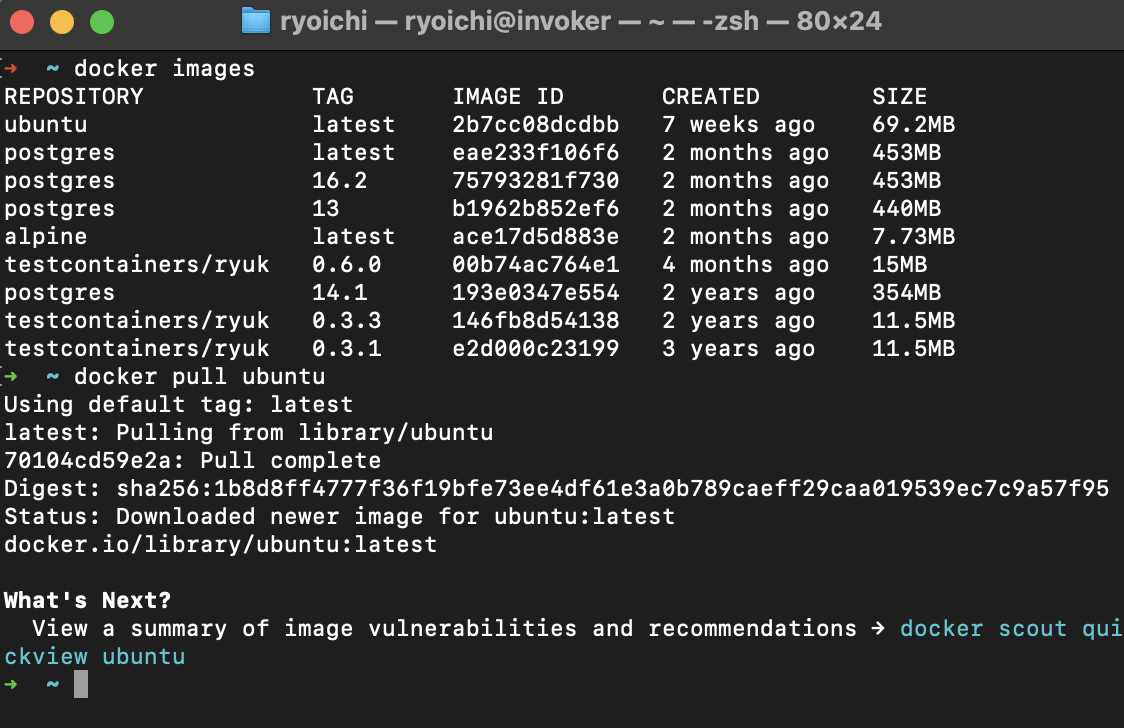


Рисунок 2 – Запуск команды docker pull ubuntu

На рисунке 3 отображен запуск команды docker images после скачивания образа

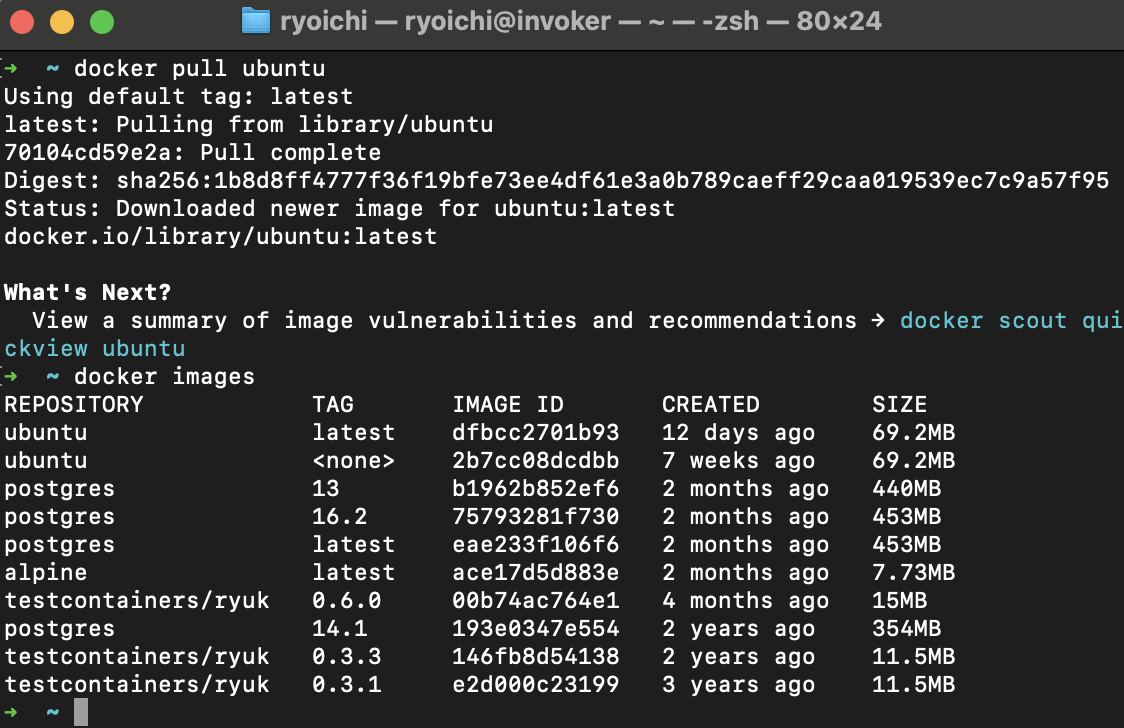


Рисунок 3 – Запуск команды docker images после скачивания образа ubuntu

На рисунке 4 отображен запуск команды docker ps – отображение всех контейнеров

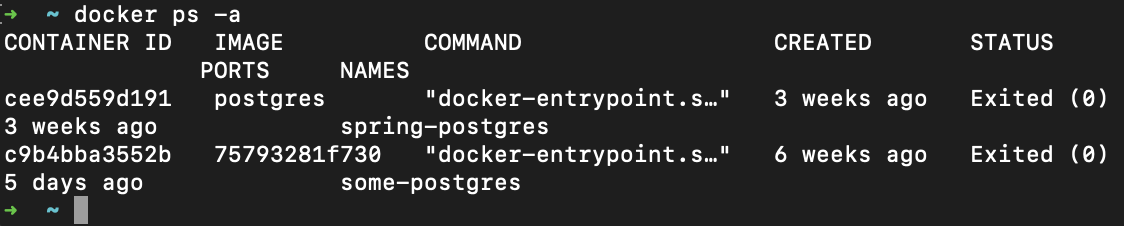


Рисунок 4 – Запуск команды docker ps

1. **Изоляция**

На рисунке 5 отображен запуск команды hostname несколько раз локально. Результат при нескольких вызовах одинаковый.

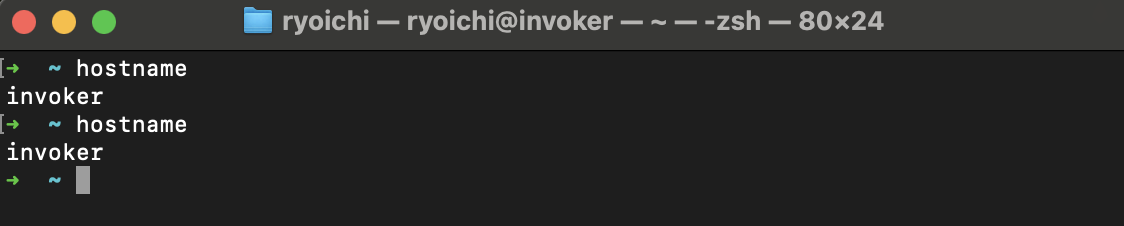


Рисунок 5 – Несколько запусков команды hostname локально

На рисунке 6 отображен запуск команды hostname несколько раз в докер-контейнере Ubuntu. Результат при нескольких вызовах разный.

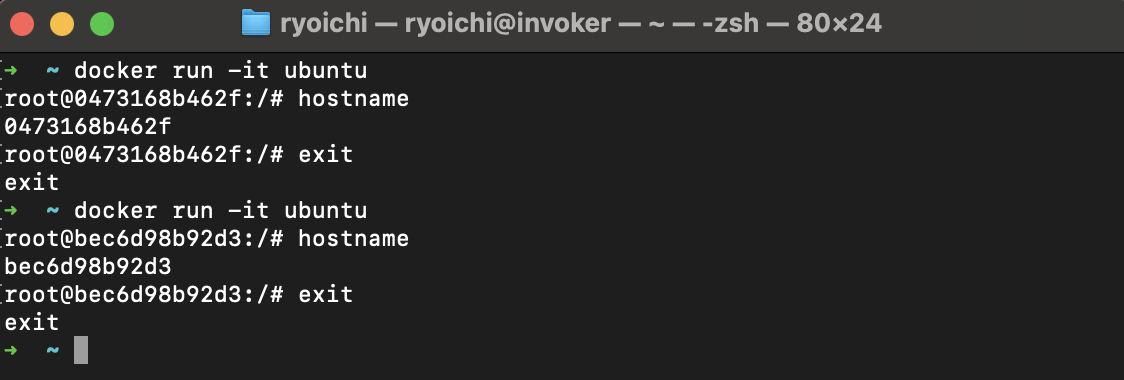


Рисунок 6 – Несколько запусков команды hostname в докер-контейнере Ubuntu

На рисунке 7 отображен запуск команды docker ps



Рисунок 7 – Запуск команды docker ps

На рисунке 8 отображен запуск контейнера без входа в интерактивный режим. В этом случае не происходит переход в bash-терминал.

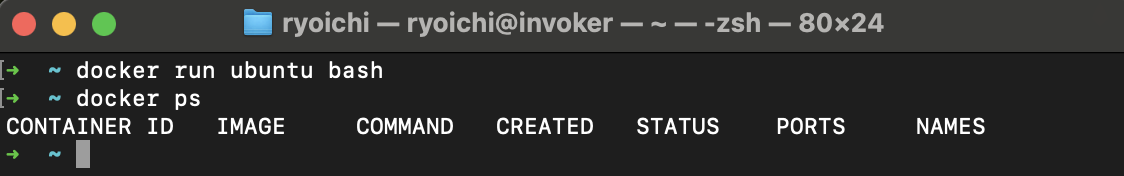


Рисунок 8 – Запуск команды docker run ubuntu bash.

На рисунке 9 отображен запуск контейнера со входом в интерактивный режим. В этом случае происходит переход в bash-терминал.

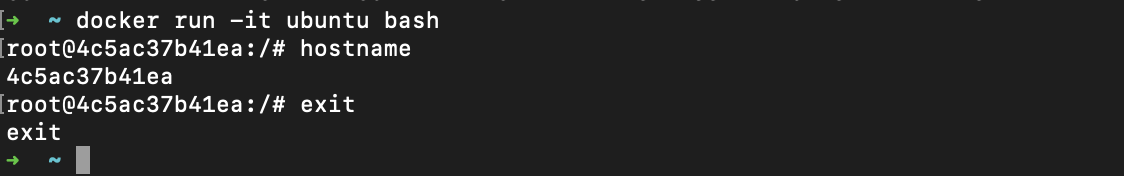


Рисунок 9 – Запуск команды docker run ubuntu bash.

1. **Работа с портами**

На рисунке 10 отображен процесс скачивания образа python

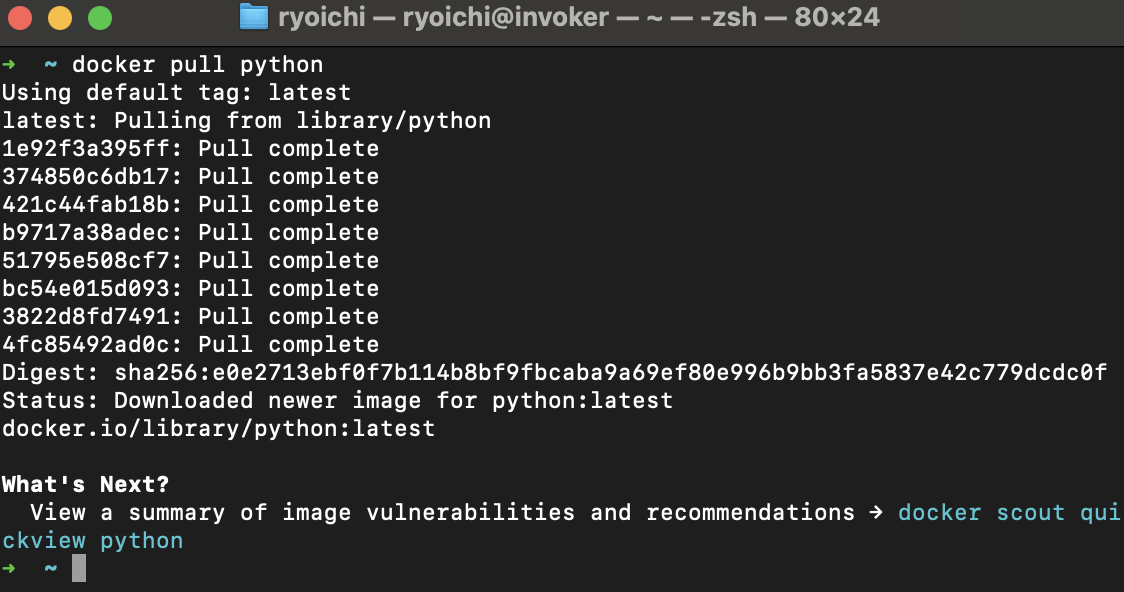


Рисунок 10 – Скачивание образа python

На рисунке 11 отображен процесс запуска python веб-сервера на порту 8080 без пробрасывания портов наружу. Без этого сервис не будет доступен из вне.

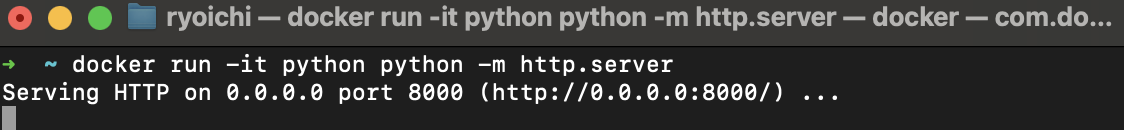


Рисунок 11 – Запуск python веб-сервера на порту 8080 без пробрасывания портов наружу

На рисунке 12 отображен результат запуска python веб-сервера на порту 8080 без пробрасывания портов наружу

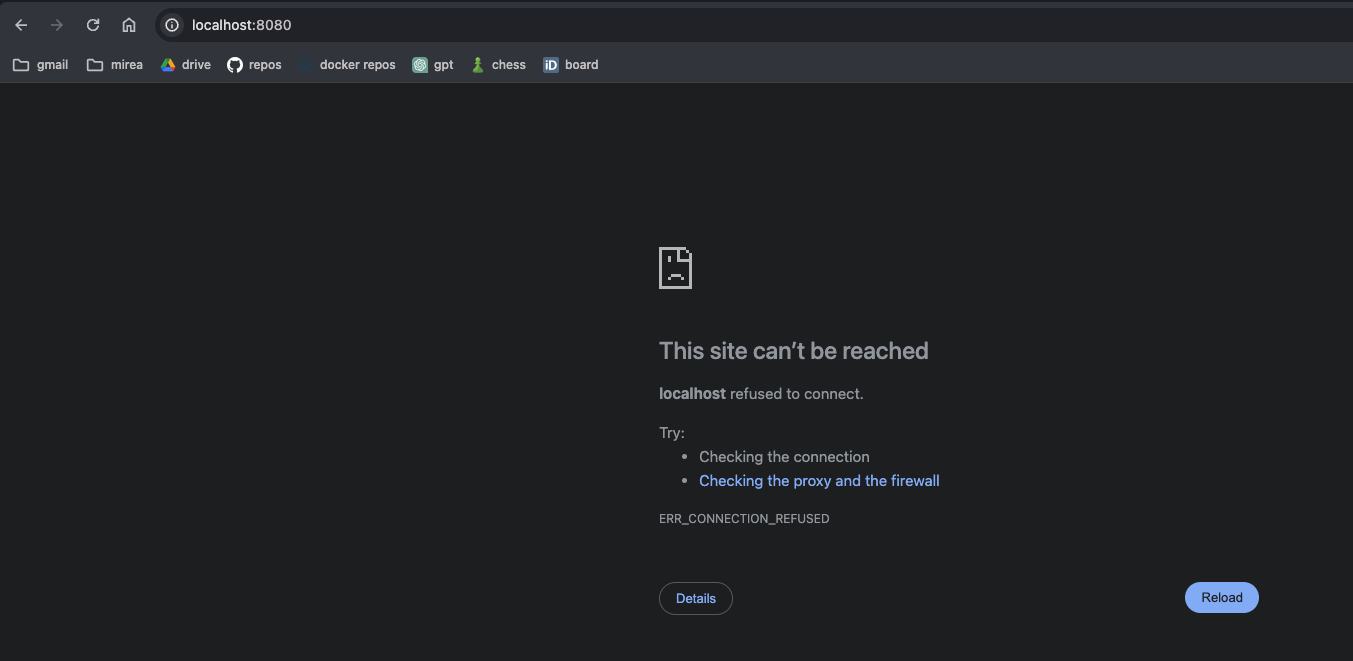


Рисунок 11 – Результат запуска python веб-сервера на порту 8080 без пробрасывания портов наружу

На рисунке 13 отображен процесс запуска python веб-сервера на порту 8080 с пробрасыванием портов наружу.



Рисунок 13 – Запуск python веб-сервера на порту 8000 с пробрасыванием портов наружу

На рисунке 14 отображен результат запуска python веб-сервера на порту 8080 с пробрасыванием портов наружу

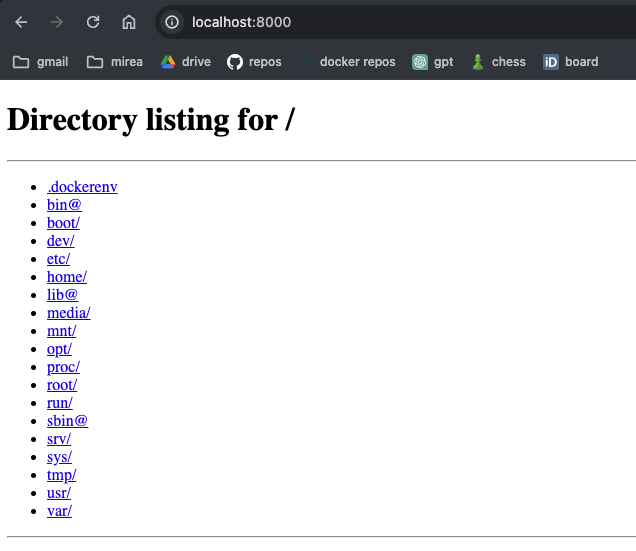


Рисунок 14 – Результат запуска python веб-сервера на порту 8000 с пробрасыванием портов наружу

На рисунке 15 отображен процесс запуска python веб-сервера на порту 8000 с пробрасыванием портов наружу. При этом порт веб-сервера (8000) маппится на порт 8888 локальной машины

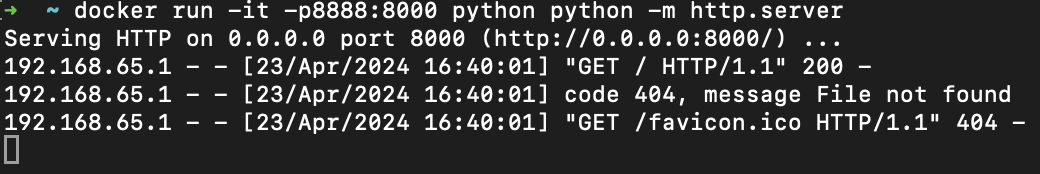


Рисунок 15 – Запуск python веб-сервера на порту 8080 с пробрасыванием портов наружу (+ маппинг порта веб-сервера на 8888)

На рисунке 16 отображен результат запуска python веб-сервера на порту 8080 с пробрасыванием портов наружу

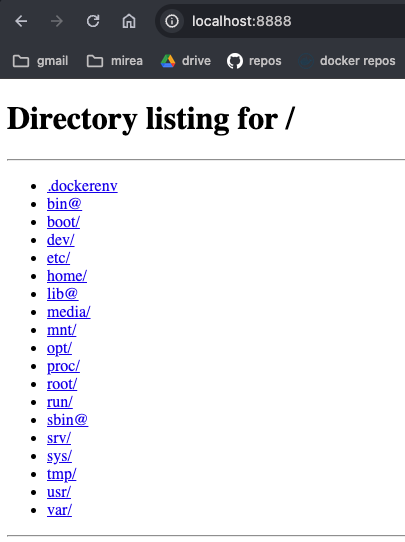


Рисунок 16 – Результат запуска python веб-сервера на порту 8080 с пробрасыванием портов наружу (+ маппинг порта веб-сервера на 8888)

1. **Именованные контейнеры, остановка и удаление**

На рисунке 17 отображен процесс завершения работы контейнера помощью CTRL + C

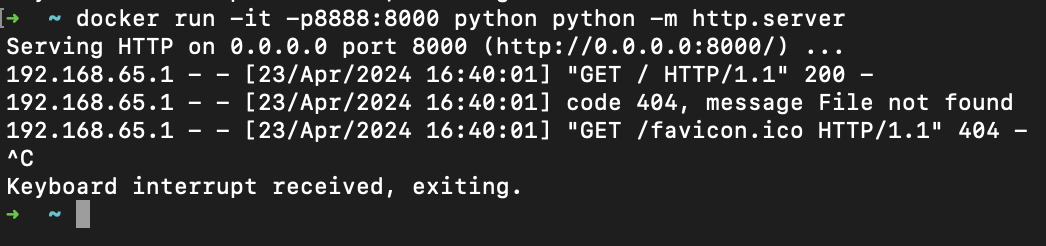


Рисунок 17 – Завершение работы контейнера с помощью CTRL + C

На рисунке 18 отображен процесс запуска контейнера с detach-флагом. В этом случае запущенный контейнер не блокирует терминал.

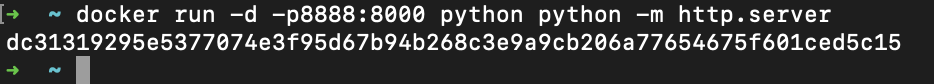


Рисунок 18 – Запуск контейнера с detach-флагом

На рисунке 19 отображен процесс запуска контейнера с задаванием имени контейнера

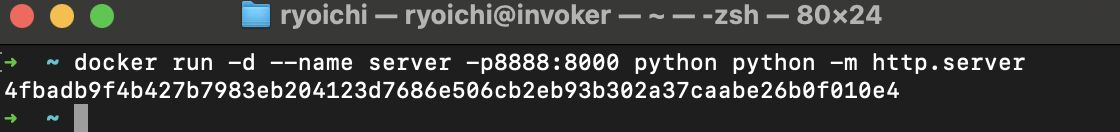


Рисунок 19 – Запуск контейнера с задаванием имени контейнера

На рисунке 20 отображен процесс просмотра всех контейнеров с фильтром pyserver

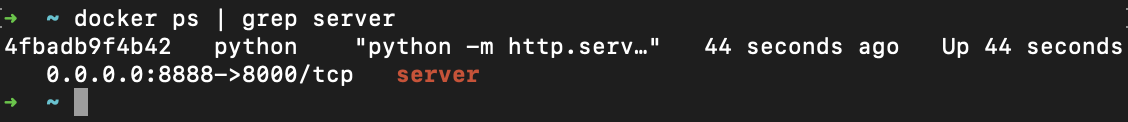


Рисунок 20 – Запуск команды docker ps | grep server

На рисунке 21 отображен процесс просмотра логов контейнера с именем pyserver

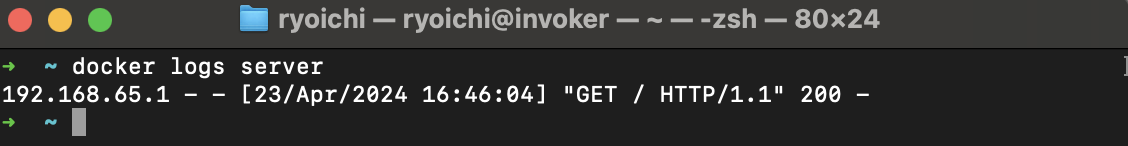


Рисунок 21 – Запуск команды docker logs

На рисунке 22 отображен процесс остановки контейнера server



Рисунок 21 – Запуск команды docker stop

На рисунке 23 отображен процесс запуска контейнера server после его остановки. Выдает ошибку, для решения конфликта необходимо удалить контейнер перед запуском.

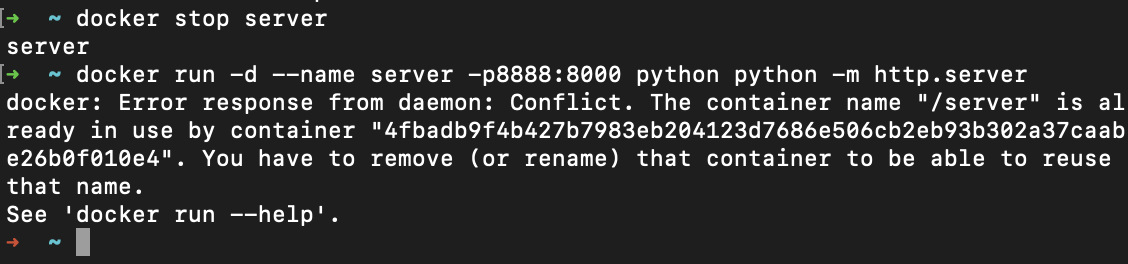


Рисунок 23 – Запуск контейнера server после его остановки

На рисунке 24 отображен процесс удаления и запуска контейнера. Теперь запуск происходит без ошибок

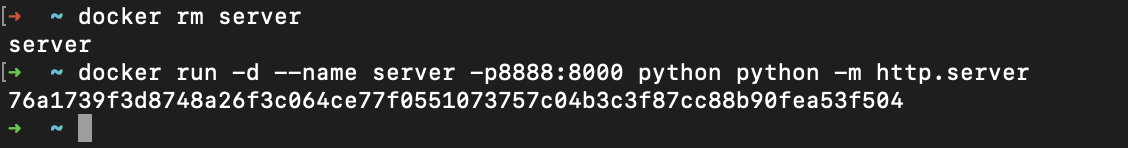


Рисунок 24 – Процесс удаления и запуска контейнера

На рисунке 25 отображен процесс тестирования флага –rm – задает автоматическое удаление контейнера после завершения работы

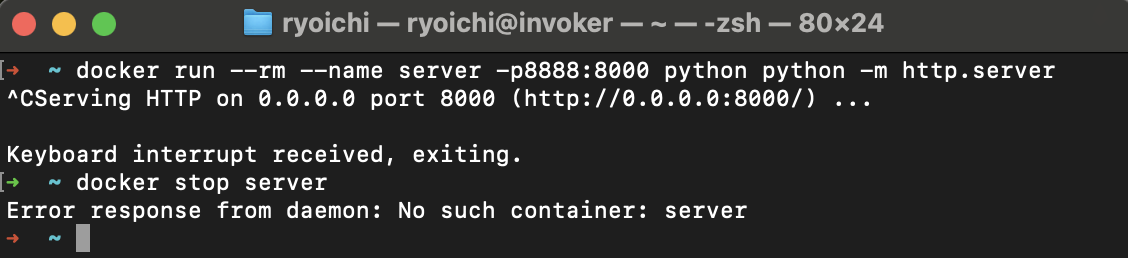


Рисунок 25 – Процесс тестирования флага –rm

1. **Постоянное хранение данных**

На рисунке 26 отображен запуск контейнера, который будет отдавать содержимое директории /mnt



Рисунок 26 – Запуск контейнера, который будет отдавать содержимое директории /mnt

На рисунке 28 отображен процесс записи hello world в hi.txt с помощью команды cd mnt && echo "hello world" > hi.txt

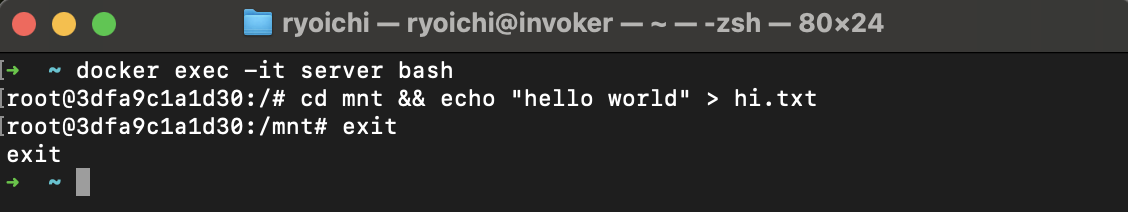


Рисунок 28 – Процесс записи hello world в hi.txt

На рисунке 29 отображен процесс захода на веб-сервер, отображена содержимое директории mnt, файл hi.txt

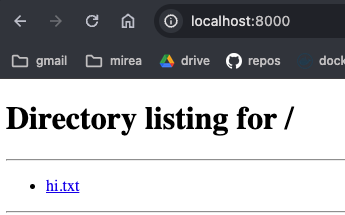
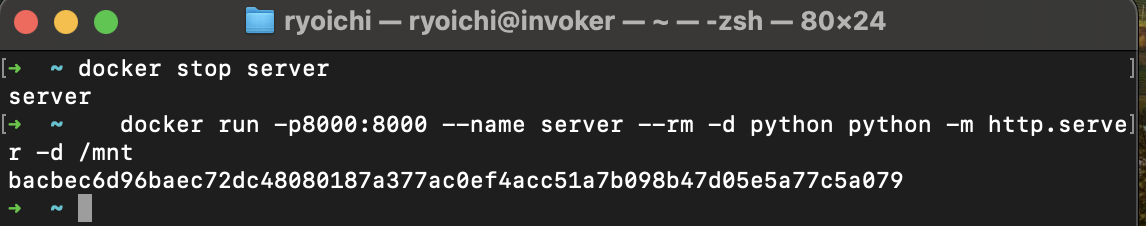


Рисунок 29 – Hi.txt файл на веб-сервере

На рисунке 30 отображен остановки контейнера и запуска заново

Рисунок 30 – Остановка контейнера и запуск заново

На рисунке 31 отображен результат просмотра директории веб-сервера после перезапуска контейнера

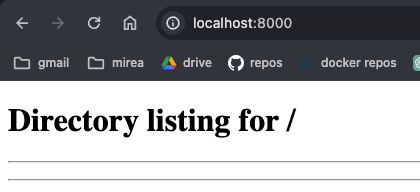


Рисунок 31 – Результат просмотра директории веб-сервера

* 1. **Тома**

На рисунке 32 отображен запуск контейнера с монтированным томом с помощью параметра -v и маппинга пути в ubuntu на путь на локальной машине

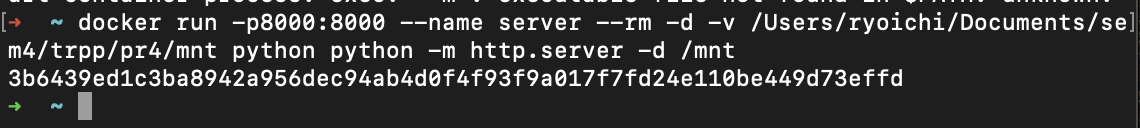


Рисунок 32 – Запуск контейнера с монтированным томом

На рисунке 33 отображено создание файла hi.txt в контейнере

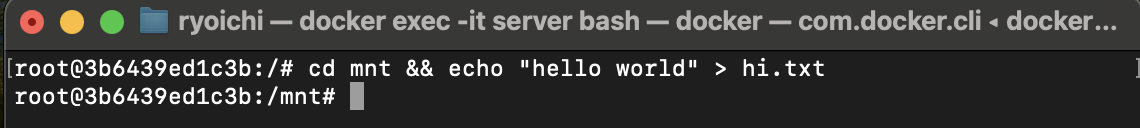


Рисунок 33 – Создание файла hi.txt

На рисунке 34 отображен процесс остановки и еще одного запуска контейнера

****

Рисунок 34 – Перезапуск контейнера

На рисунке 35 отображена проверка на наличие файла hi.txt после перезапуска. Файл присутствует

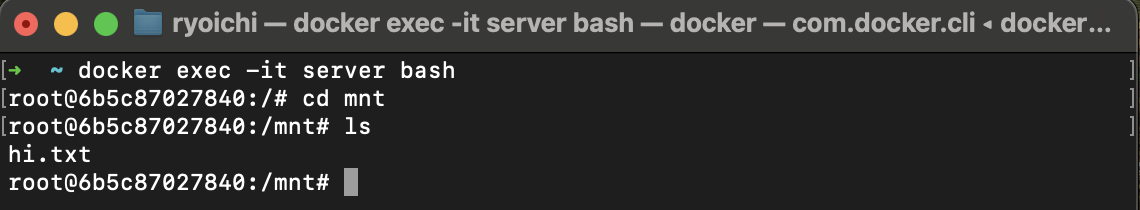


Рисунок 35 – Проверка на наличие файла hi.txt

На рисунке 36 отображен просмотр места, где хранятся файлы с помощью команды docker inspect server,

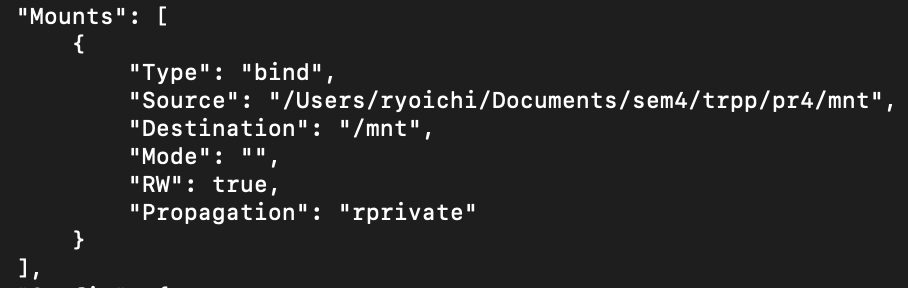


Рисунок 36 – Просмотр места, где хранятся файлы

* 1. **Монтирование директорий и файлов**

На рисунке 37 отображен запуск контейнера с пробрасыванием директории

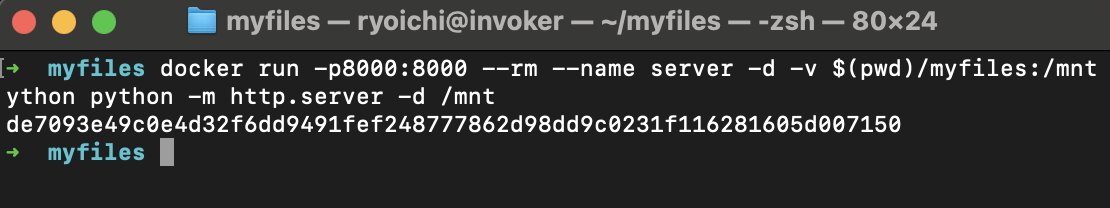


Рисунок 37 – Запуск контейнера с пробрасыванием директории

На рисунке 38 отображен процесс просмотра файлов проброшенной директории

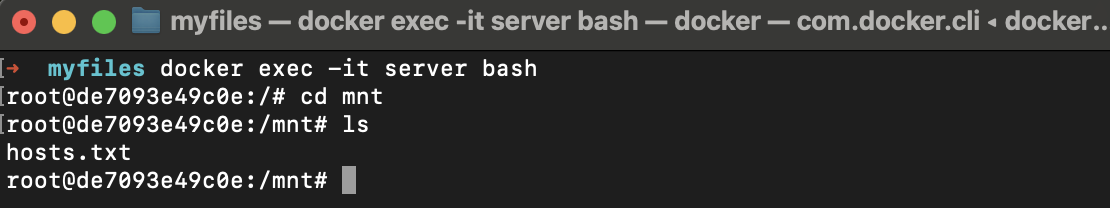


Рисунок 38 – Процесс просмотра файлов проброшенной директории

На рисунке 39 отображен процесс просмотра директории myfiles на хостовой машине после создания файла hi.txt в контейнере

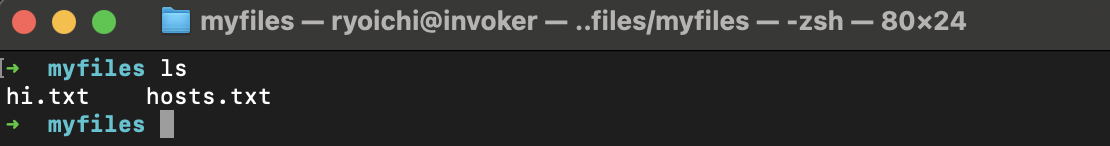


Рисунок 39 – Процесс просмотра директории myfiles на хостовой машине после создания файла hi.txt в контейнере

На рисунке 40 отображен процесс монтирования одного файла со сменой имени

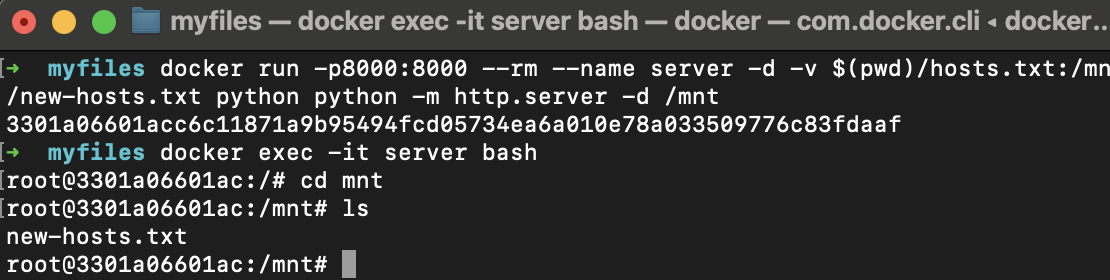


Рисунок 40 – Процесс монтирования одного файла со сменой имени

1. **Переменные окружения**

На рисунке 41 отображен процесс добавления переменной окружения LOGIN=”Michael” и просмотра всех переменных контейнера с помощью терминала Bash

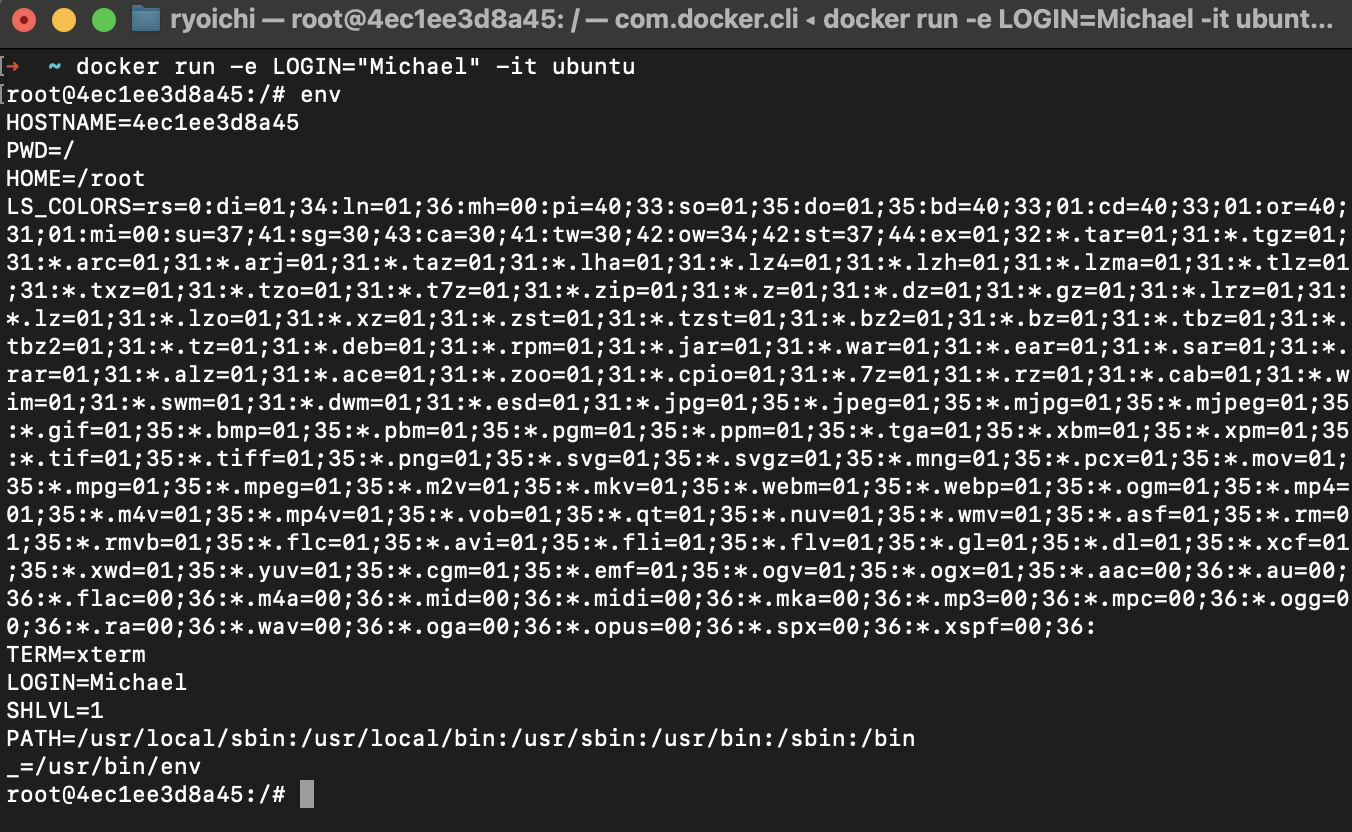


Рисунок 41 – Тестирование указывания переменных окружения при запуске контейнера

1. **Dockerfile**

На рисунках 42-43 отображены докер-файл и процесс билда образа из докер-файла

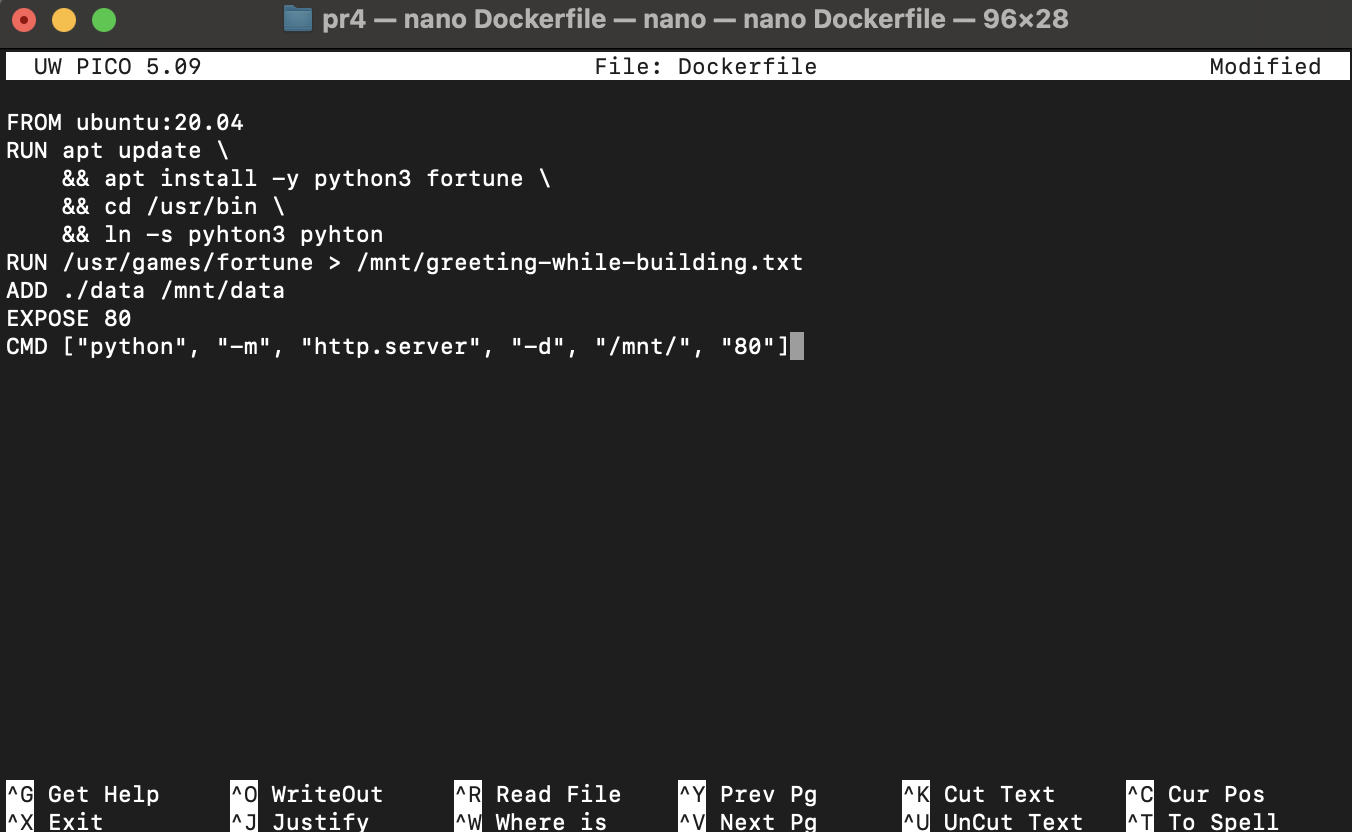


Рисунок 42 – Докер-файл

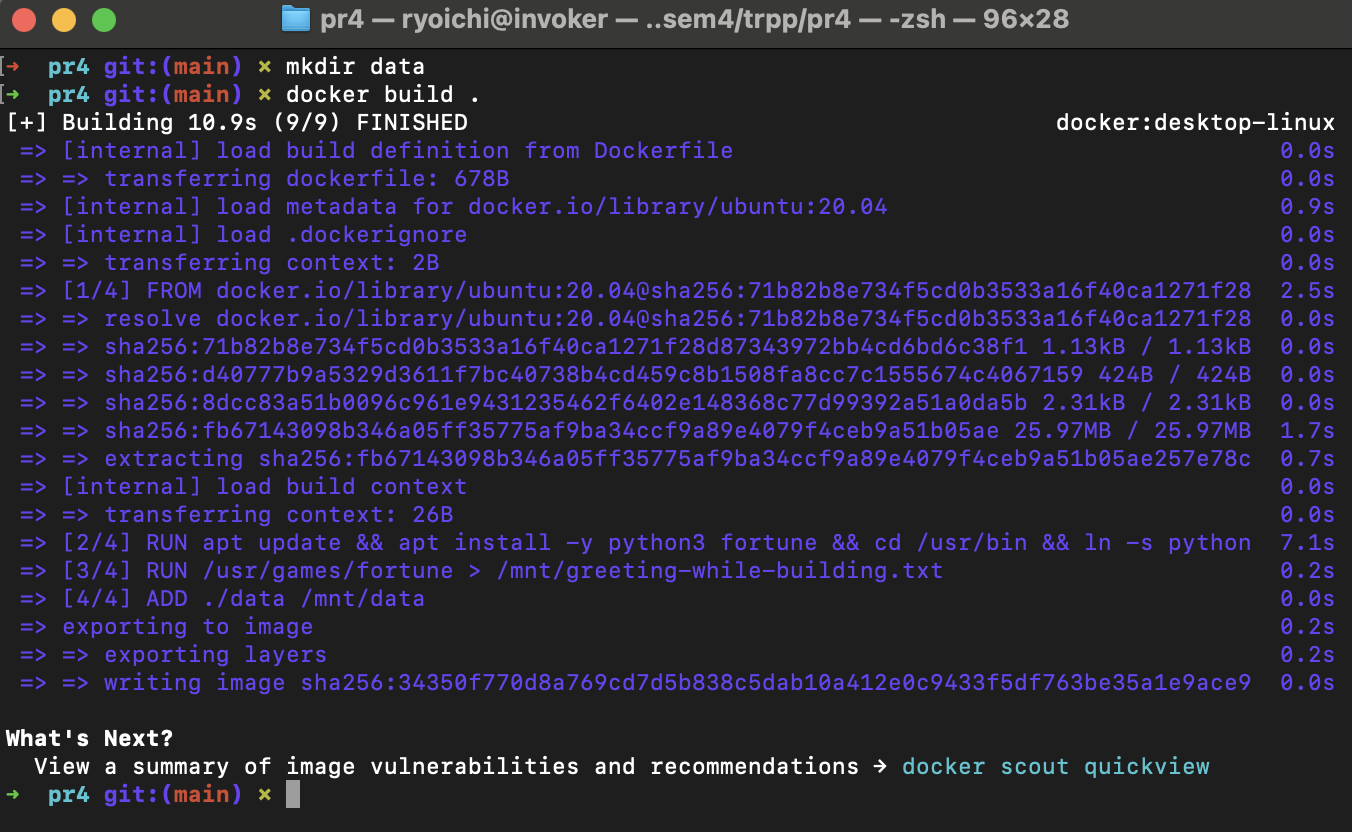


Рисунок 43 – Билд образа из докер-файла

На рисунке 44 отображены процесс запуска контейнера

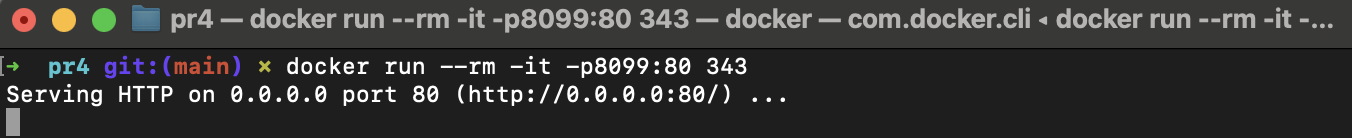


Рисунок 44 – Запуск контейнера

На рисунке 45 отображены процесс тестирования запущенного контейнера

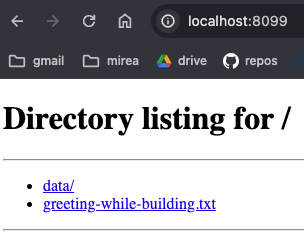


Рисунок 45 – Тестирование запущенного контейнера

1. **Индивидуальные задания**

На рисунке 46-47 отображен докер-файл и процесс билда образа из докер-файла

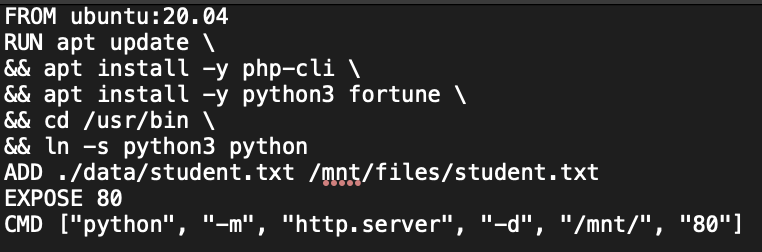


Рисунок 46 – Докер-файл

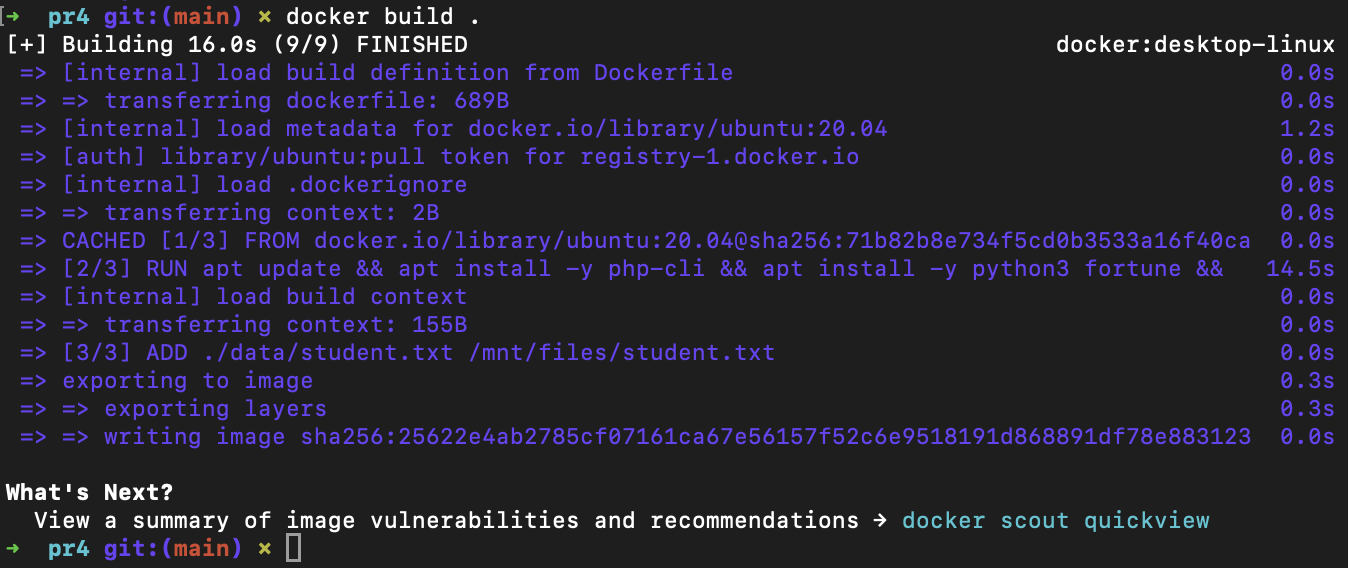


Рисунок 47 – Докер-файл

На рисунке 48 отображены процесс запуска контейнера с образом из докер-файла

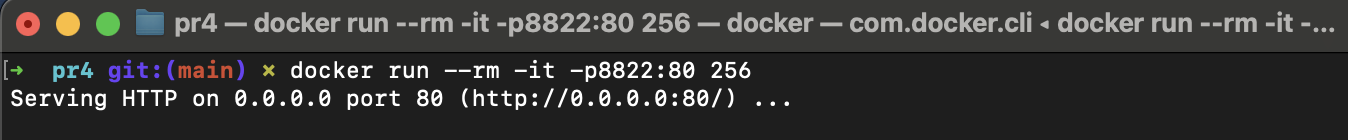


Рисунок 48 – Запуск контейнера с образом из докер-файла

На рисунке 49 отображены процесс тестирования запущенного контейнера с образом из докер-файла

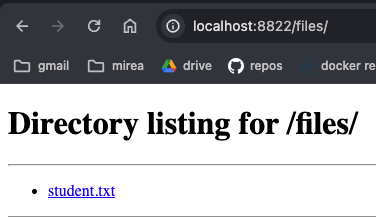


Рисунок 49 – Тестирование запущенного контейнера с образом из докер-файла

**Вывод**

В ходе работы мы познакомились с Docker, изучили его возможности, в том числе запуск контейнеров из образов, создание собственных образов, монтирование данных с хост-машины в контейнер, переменные окружения, работу с портами.