|  |  |
| --- | --- |
| Федеральное агентство железнодорожного транспорта  Омский государственный университет путей сообщения  Кафедра «Автоматика и системы управления»  ОСНОВЫ DOCKER  Лабораторная работа № 3  по дисциплине «Информационные технологии» | |
|  | Студент гр. 22м                            Н.И. Яковлев  «    »                  2023 г.  Руководитель –  старший преподаватель кафедры «АиСУ»                            Т. В. Васеева  «    »                  2023 г. |
| Омск 2023 | |

Цель работы

Целью данной лабораторной работы является знакомство с программой docker.

# Выполнение работы

## Предсловие

Ниже указанные команды могут выполняться с использованием утилиты «sudo» для выполнения какой-либо команды с учетной записи имеющей право «суперпользователя».

## Установка docker на виртуальную машину

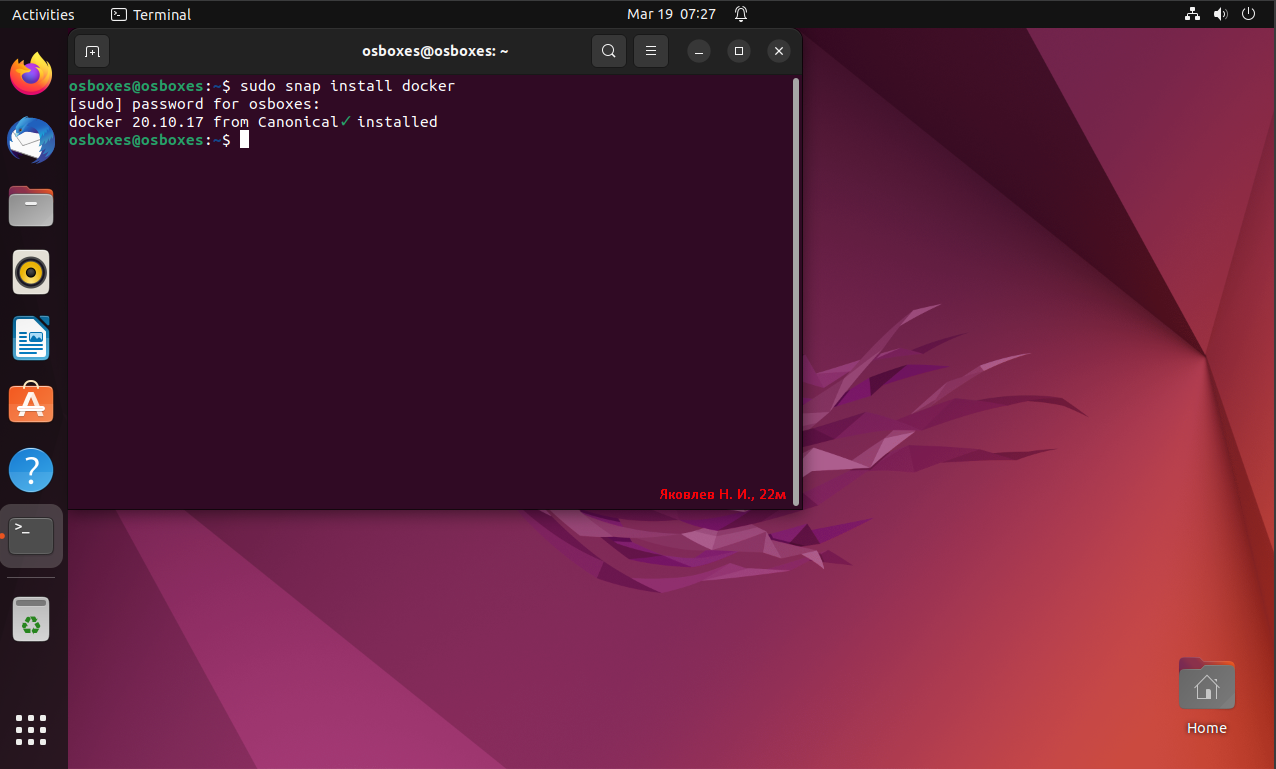
Для установки docker на ВМ, необходимо зайти в «Терминал» и выполнить следующую команду «snap install docker» (рисунок 1).

Рисунок  – Установка docker

Для проверки корректности установки docker, создадим образ «hello-world», для этого необходимо выполнить команду «docker run hello-world» (рисунок 2).

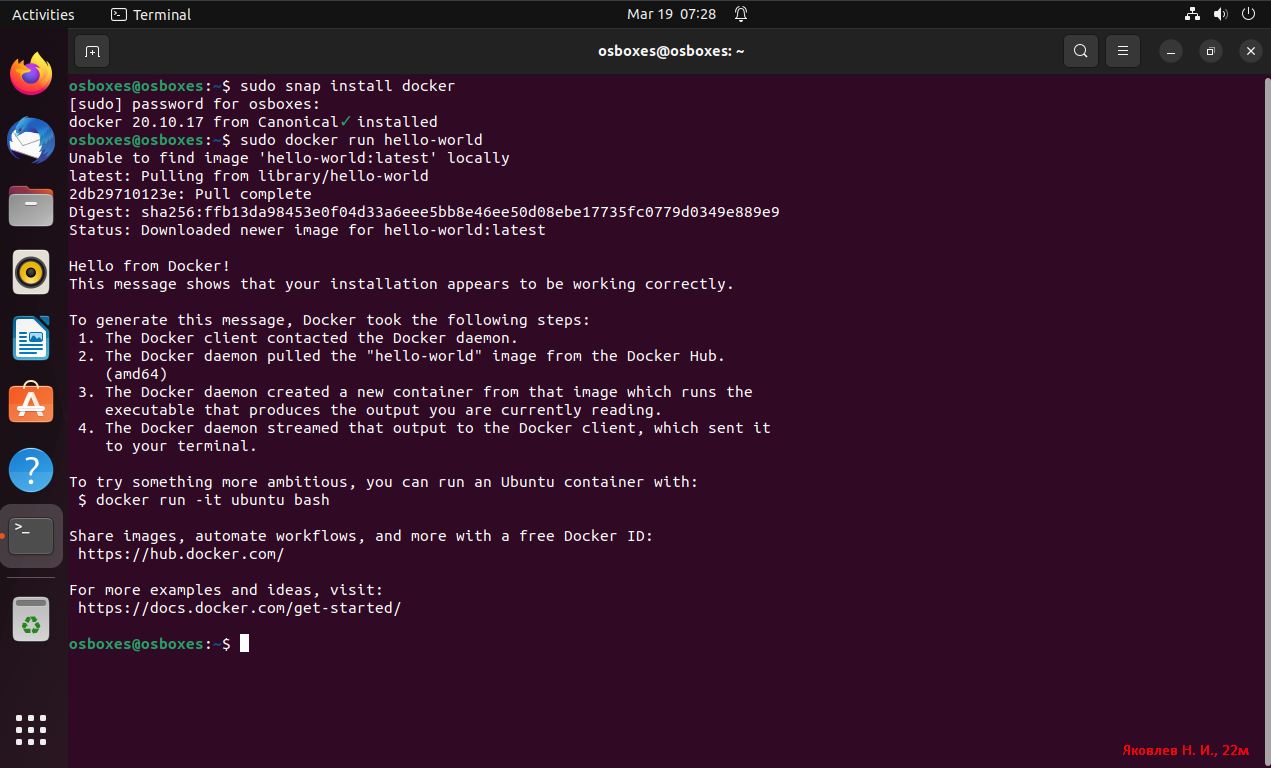


Рисунок  – Образ «hello-world»

## Работа с docker

Для того, чтоб посмотреть имеющиеся образы в docker, необходимо выполнить следующую команду «docker images» (рисунок 3).

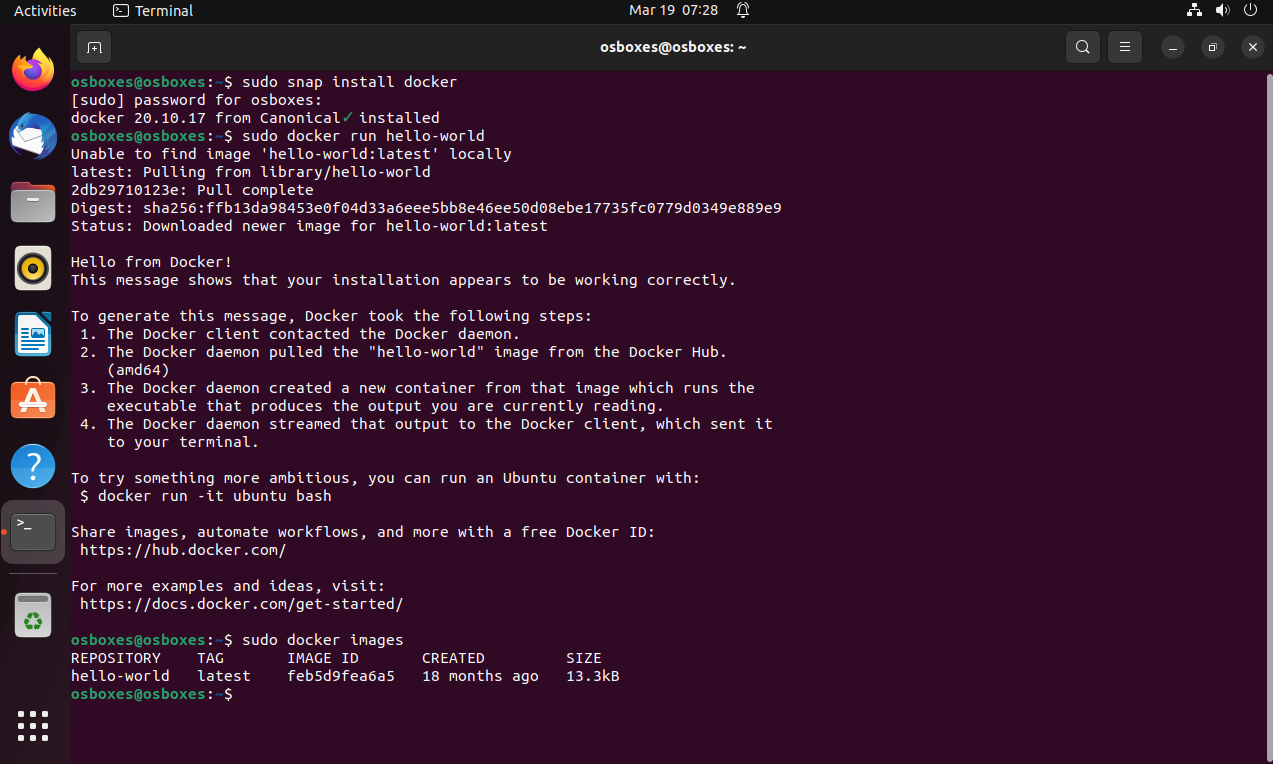


Рисунок  – Команда для просмотра образов

Попробуем удалить образ с помощью команды «docker image rm {id}» (рисунок 4).

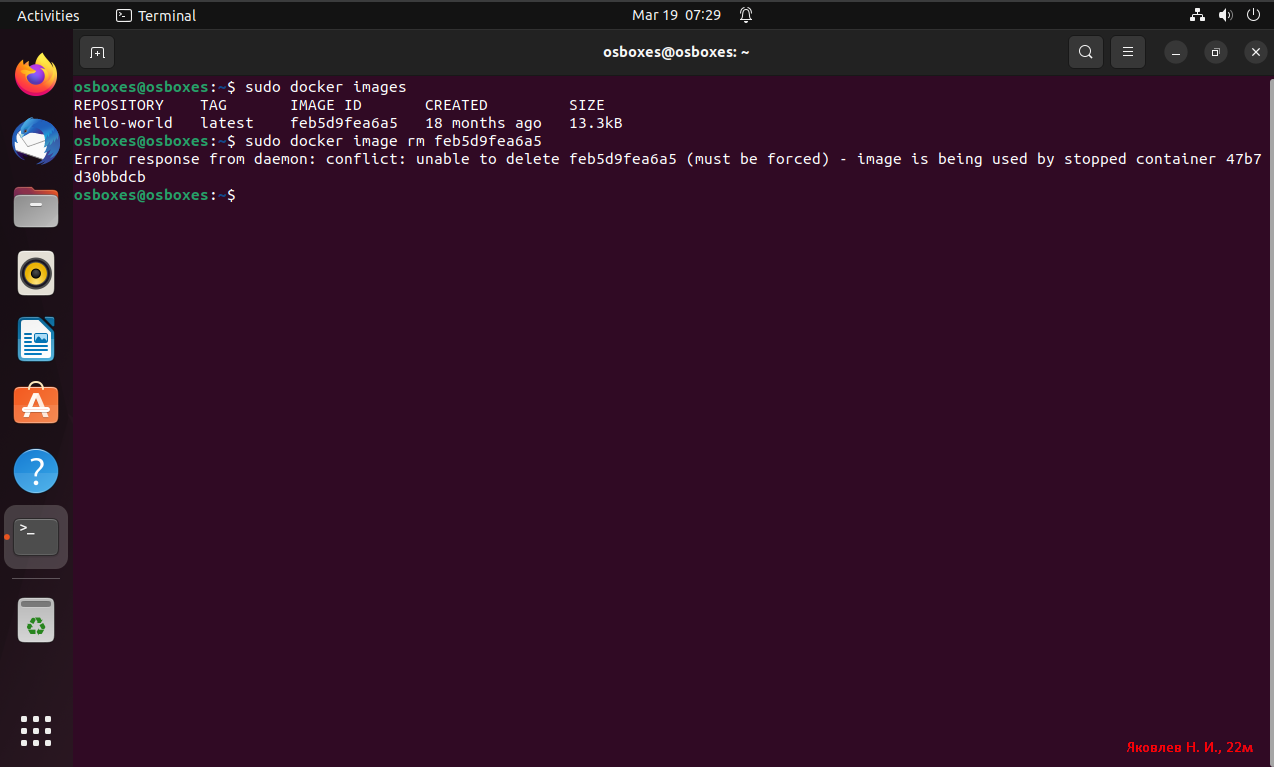


Рисунок  – Команда для удаления образа

В результате получим ошибку, в котором говорится о том, что данный образ используется одним из контейнеров, т.е. иначе говоря, для начала требуется удалить контейнер, а лишь после сам образ. Посмотрим текущие контейнеры с помощью команды «docker ps -a» (рисунок 5).

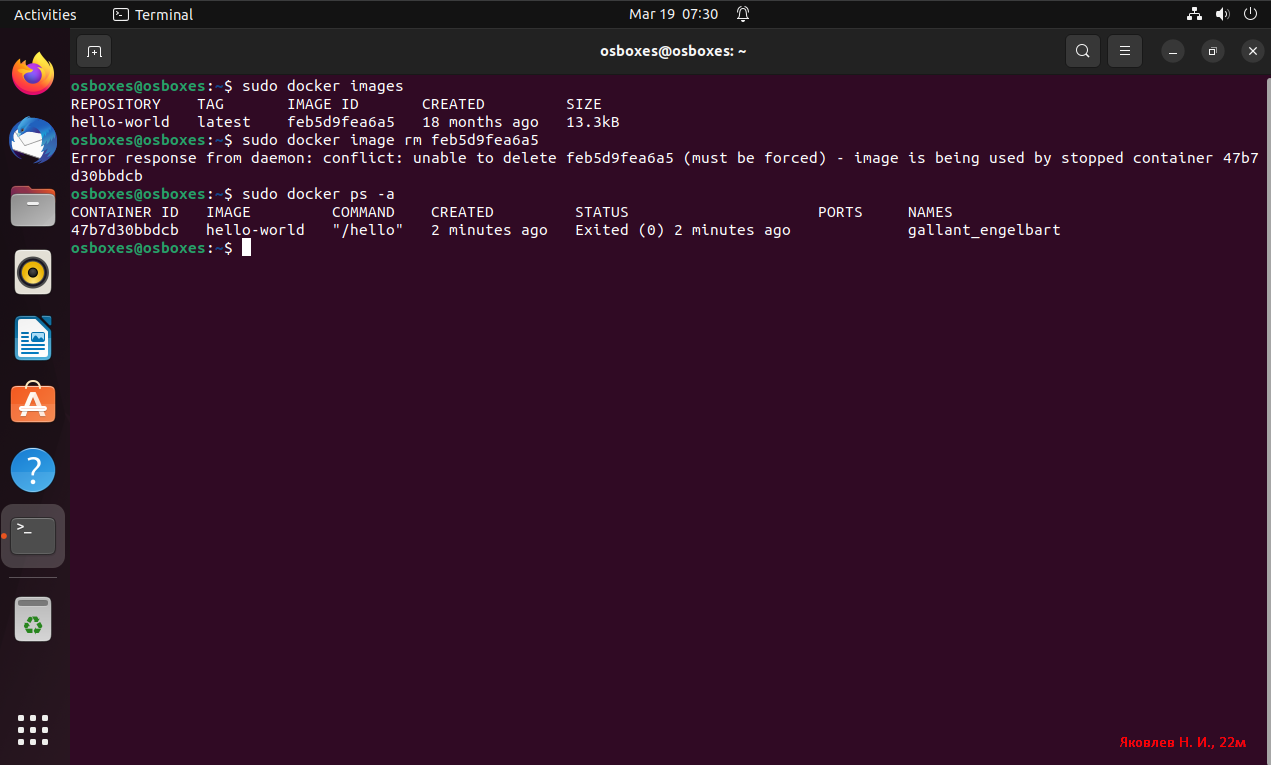


Рисунок  – Команда для просмотра контейнером

Затем удалим контейнер с помощью команды «docker container rm {id}» (рисунок 6).

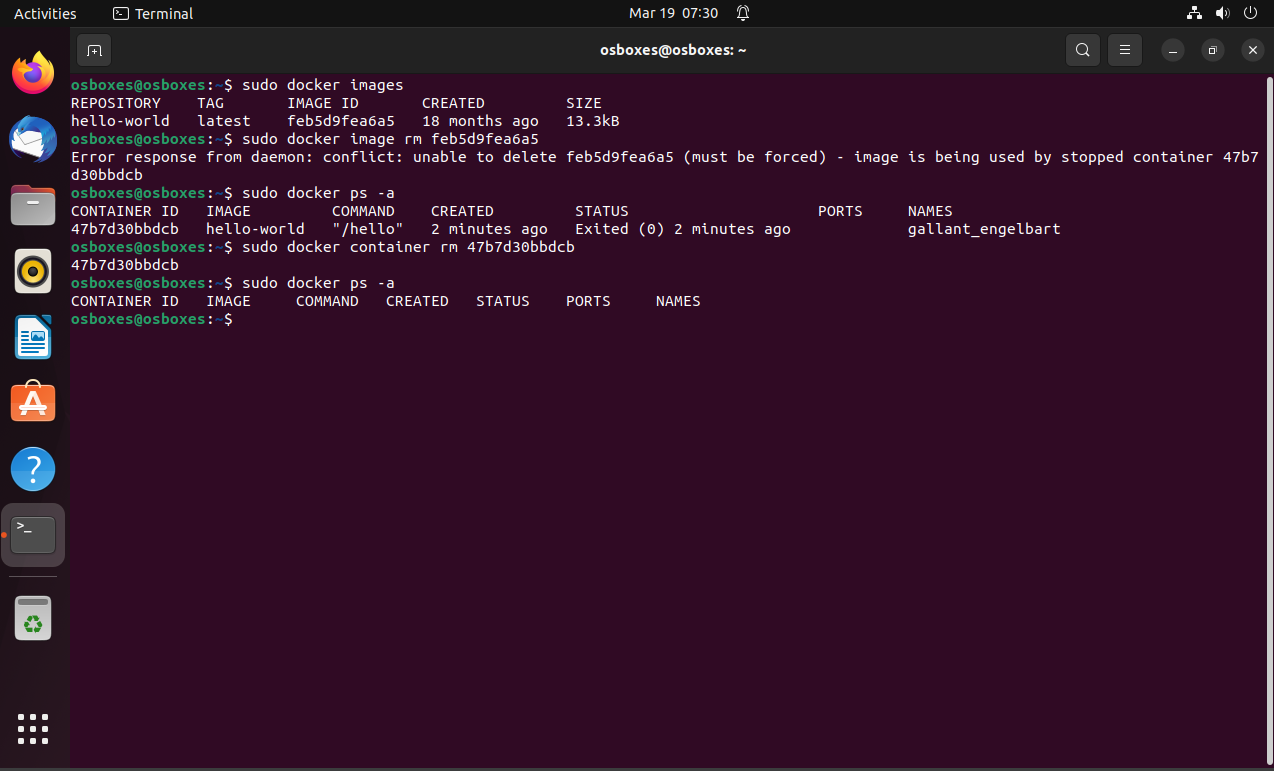


Рисунок  – Команда для удаления контейнера

Наконец удалим ранее созданный образ «hello-world» (рисунок 7).

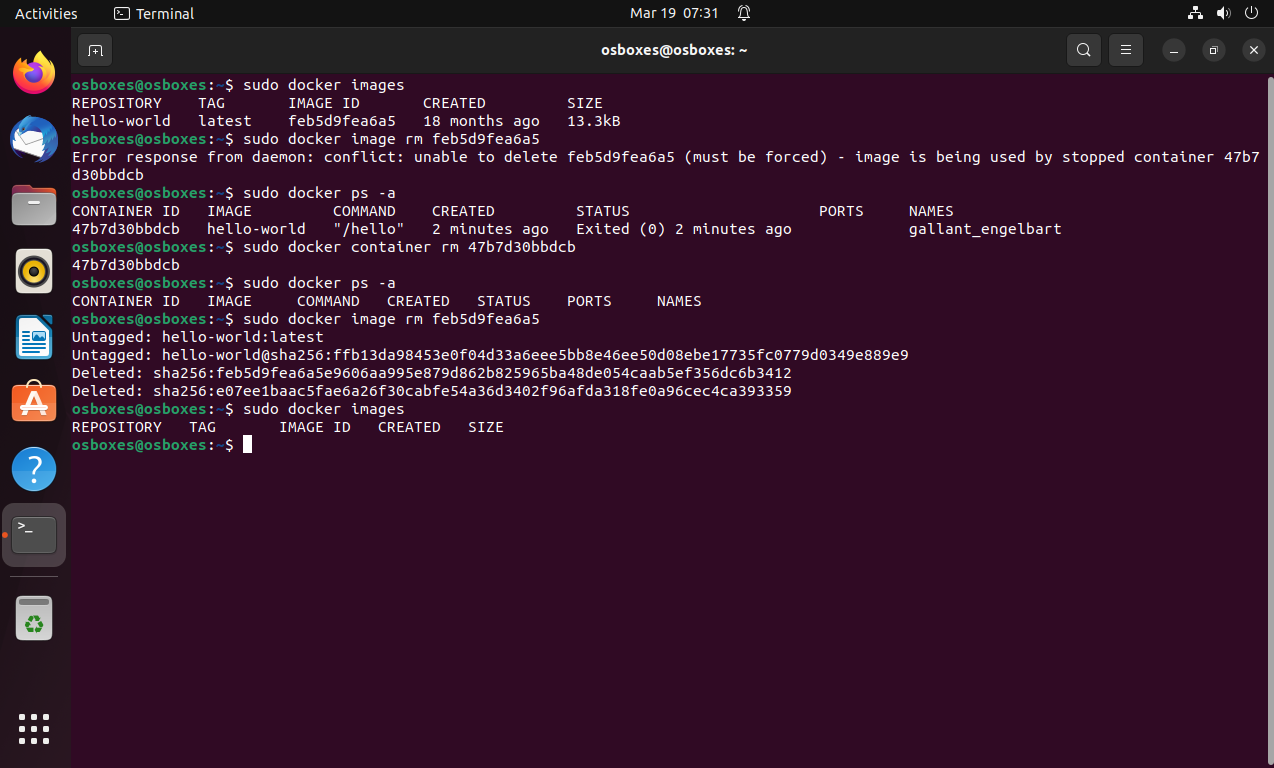


Рисунок  – Удаление образа

Установим образ ОС «alpine» с помощью команды «docker pull alpine» (рисунок 8).

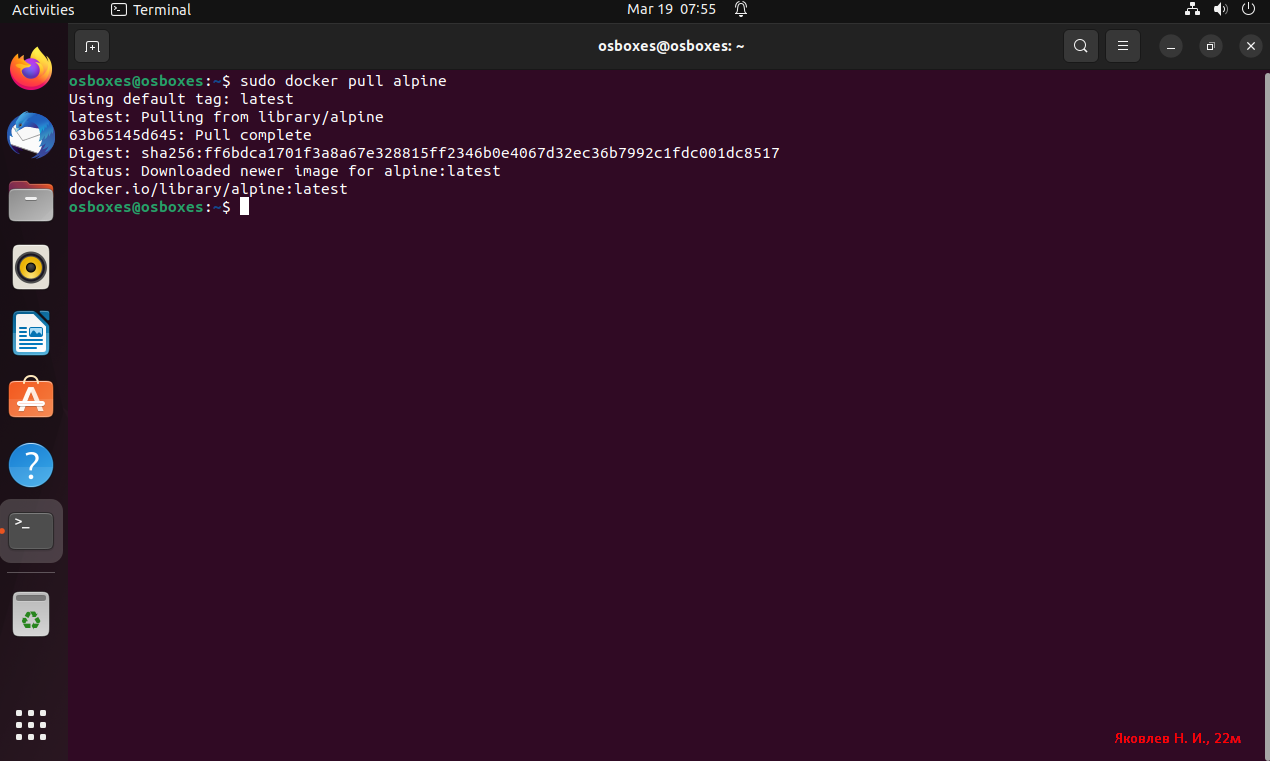


Рисунок  – Команда для установки образа

Запустим созданный образ с помощью команды «docker run -it alpine». После запуска контейнера, мы можем вызывать команды-косноли из нашего терминала внутри контейнера, выполним команды из последнего раздела лабораторной работы (рисунок 9).

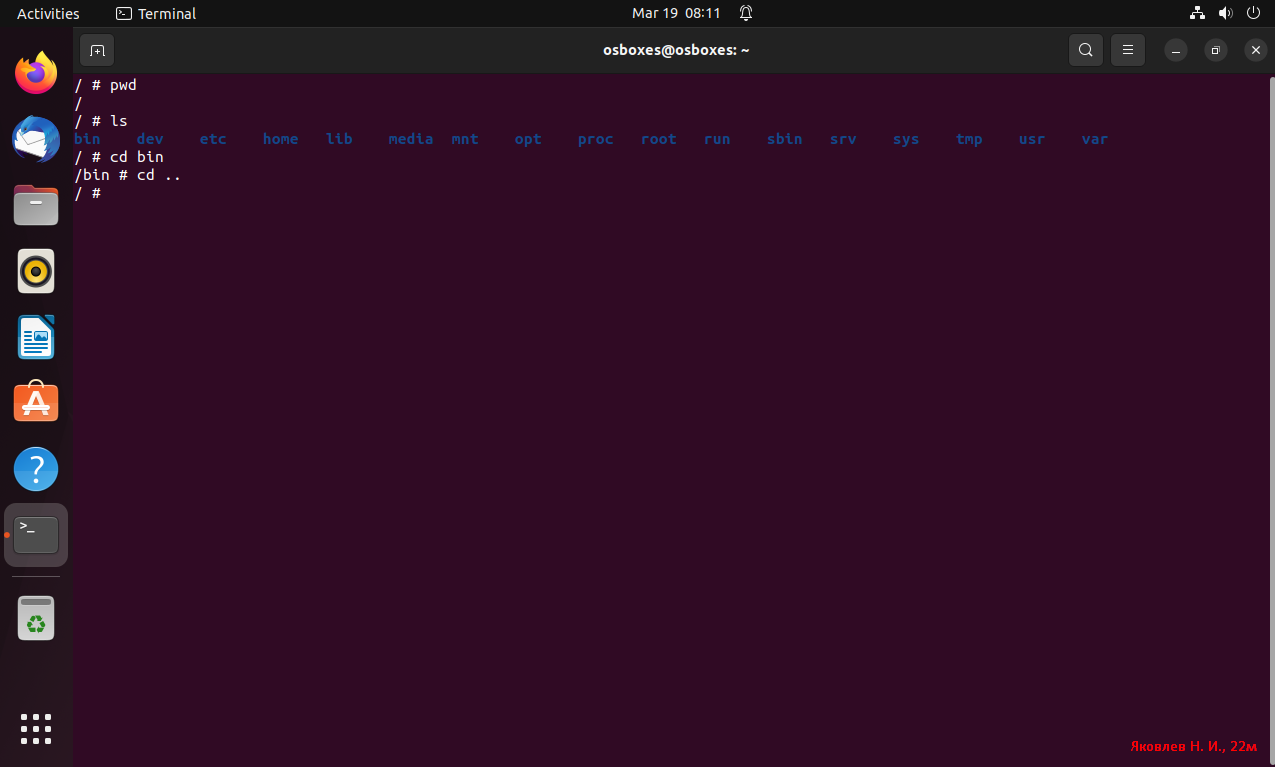


Рисунок  – Работа с терминалом внутри контейнера

Остановим контейнер с помощью команды «docker container stop {id}» (рисунок 10).

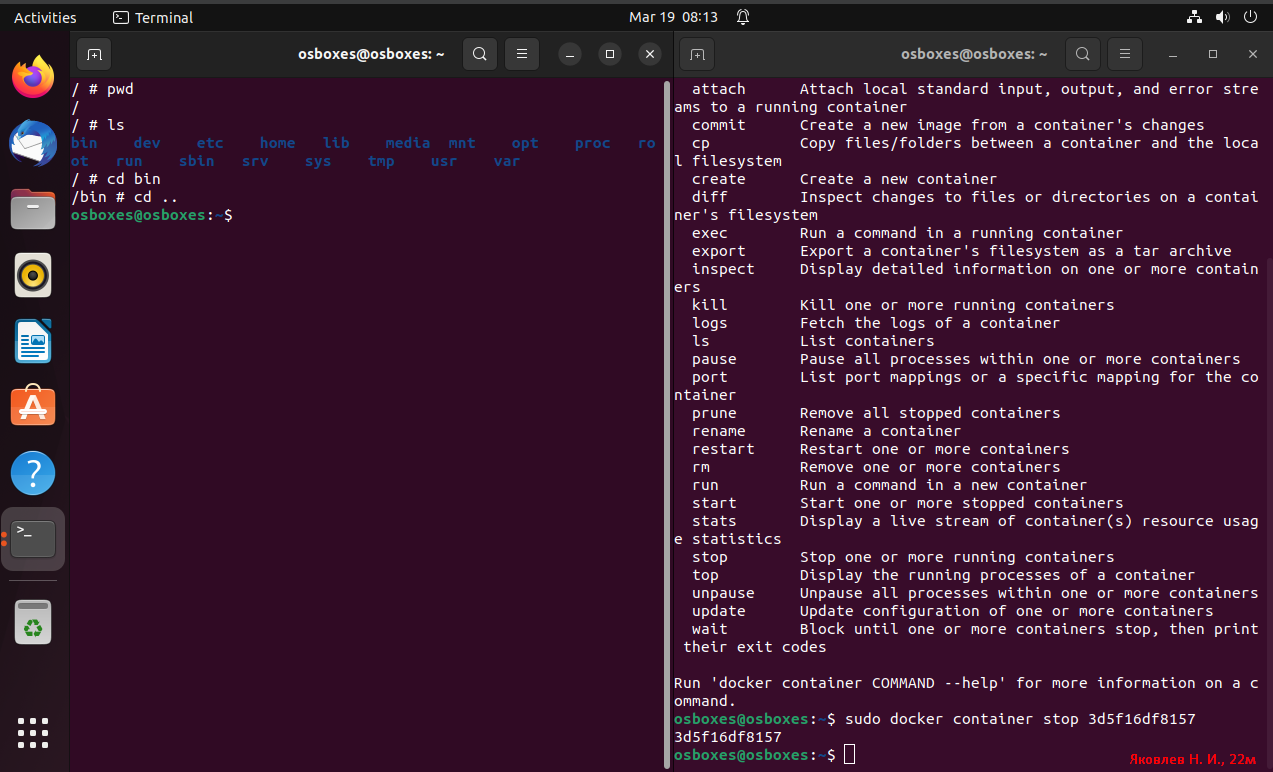


Рисунок  – Команда для остановки контейнера

## Спислк испльзуемых команд в терминале Linux

Таблица №1 – Команды

|  |  |  |
| --- | --- | --- |
| Команда | Значение | Расшифровка ключей |
| snap install docker | Установка docker на ОС | - |
| docker run {name} | Запуск образа | - |
| docker images | Просмотр образов | - |
| docker image rm {id} | Удаление образа | - |
| docker ps -a | Просмотр контейнеров | -a указывает docker вывести также остановленные контейнеры |
| docker container rm {id} | Удаление контейнера | - |
| docker pull {name} | Загрузка контейнера | - |
| docker run -it {name} | Запуск образа | -it указывает docker выделить псевдо-TTY соединение к stdin контейнера |
| docker container stop {id } | Остановка контейнера | - |

# Контрольные вопросы

## Пример создания собственного образа в Linux Ubuntu

Для начала создадим папку, в которой продолжим работу (рисунок 11).

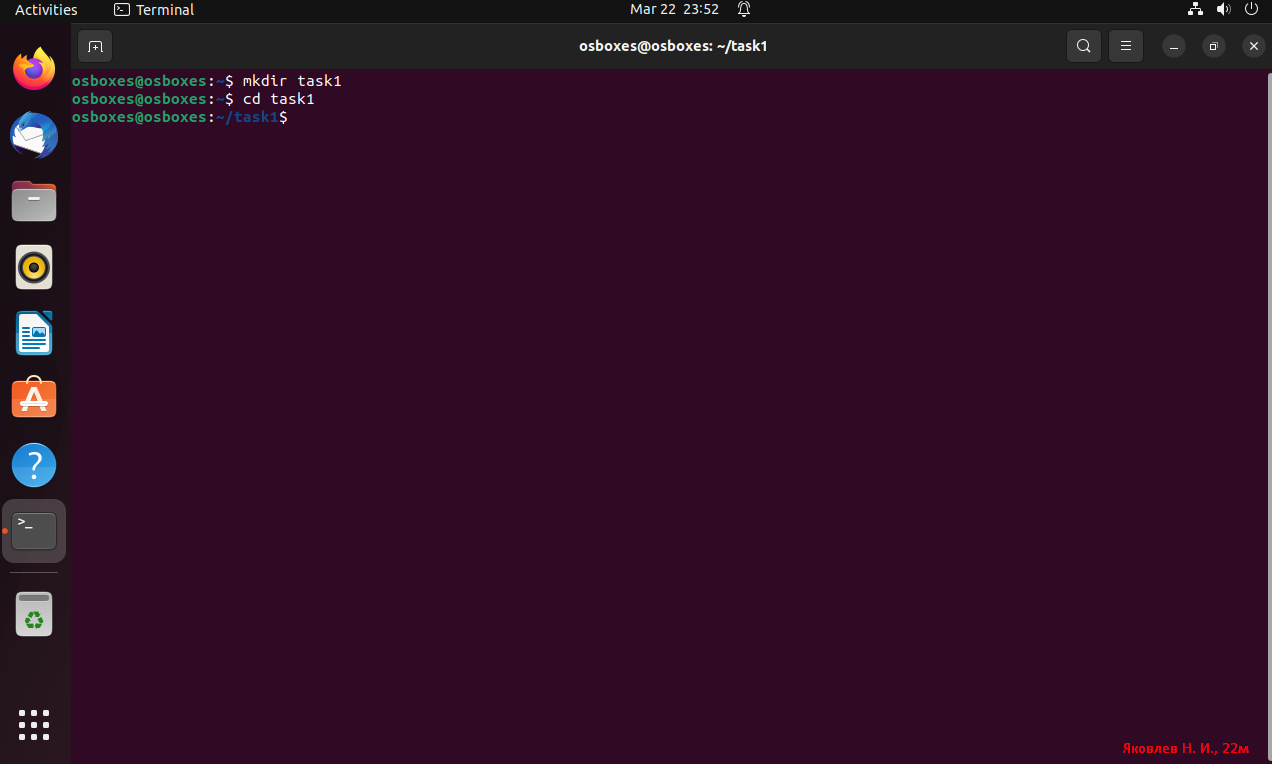


Рисунок  – Создание рабочей папки

Затем в этой папке создадим файл «Dockerfile», он нужен для того, чтоб Docker мог создать образ на основе описания из этого файла (рисунок 12).

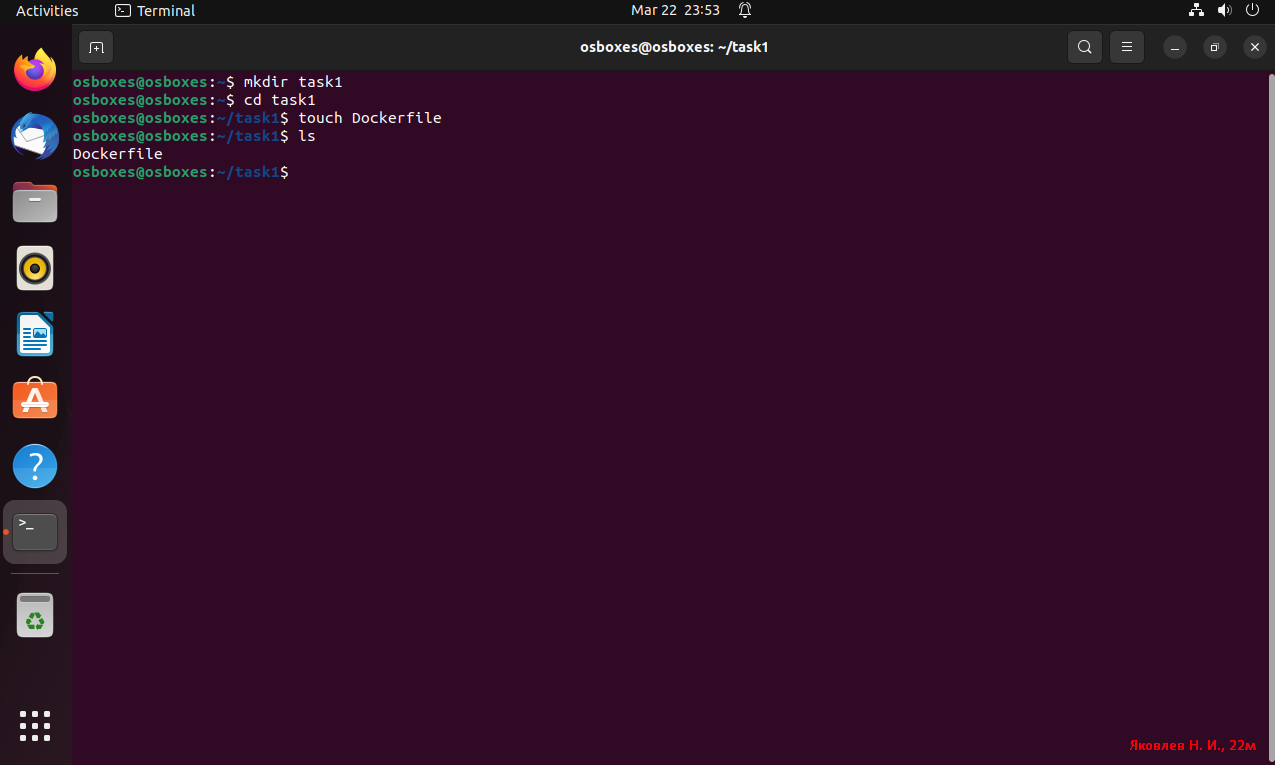


Рисунок  – Dockerfile

Отредактируем файл следуюзим образом (рисунок 13).

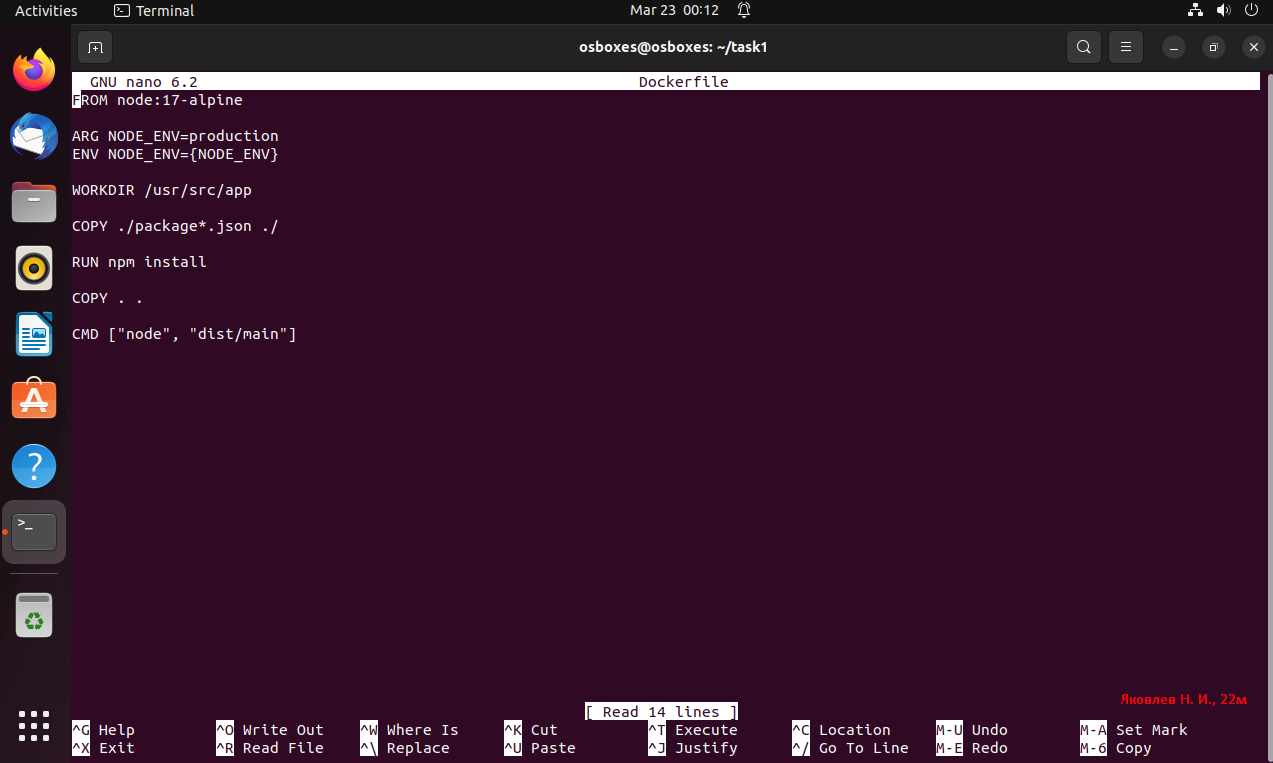


Рисунок  – Содержимое Dockerfile

Разберем построчно «Dockerfile» (таблица №2)

Таблица №2 – Dockerfile

|  |  |
| --- | --- |
| Строчка | Описание |
| FROM node:17-alpine | Указывает родительский образ, в нашем случае «node:17-alpine» это образ ОС alpine с установленным NodeJS 17-ой версии |
| ARG NODE\_ENV=production | Создаем локальный аргумент «NODE\_ENV» со значением «production» |
| ENV NODE\_ENV=${NODE\_ENV} | Устанавливаем переменную окружения «NODE\_ENV» со значением локального аргумента |
| WORKDIR /usr/src/app | Устанавливаем папку в которой будет происходить работы внутри контейнера |
| COPY ./package\*.json ./ | Копируем файлы по шаблону «package\*.json» с локального компьютера в образ |
| RUN npm install | Запускаем команду установки зависимостей npm |
| COPY . . | Копируем локальные файлы в образ |
| CMD ["node", "dist/main"] | Устанавливаем команду, которая будет запускаться всякий раз при запуске контенйера с этим образом |

Также добавим несколько файлов в рабочую папку: «package.json» и «main.js» (рисуноки 14 - 16).

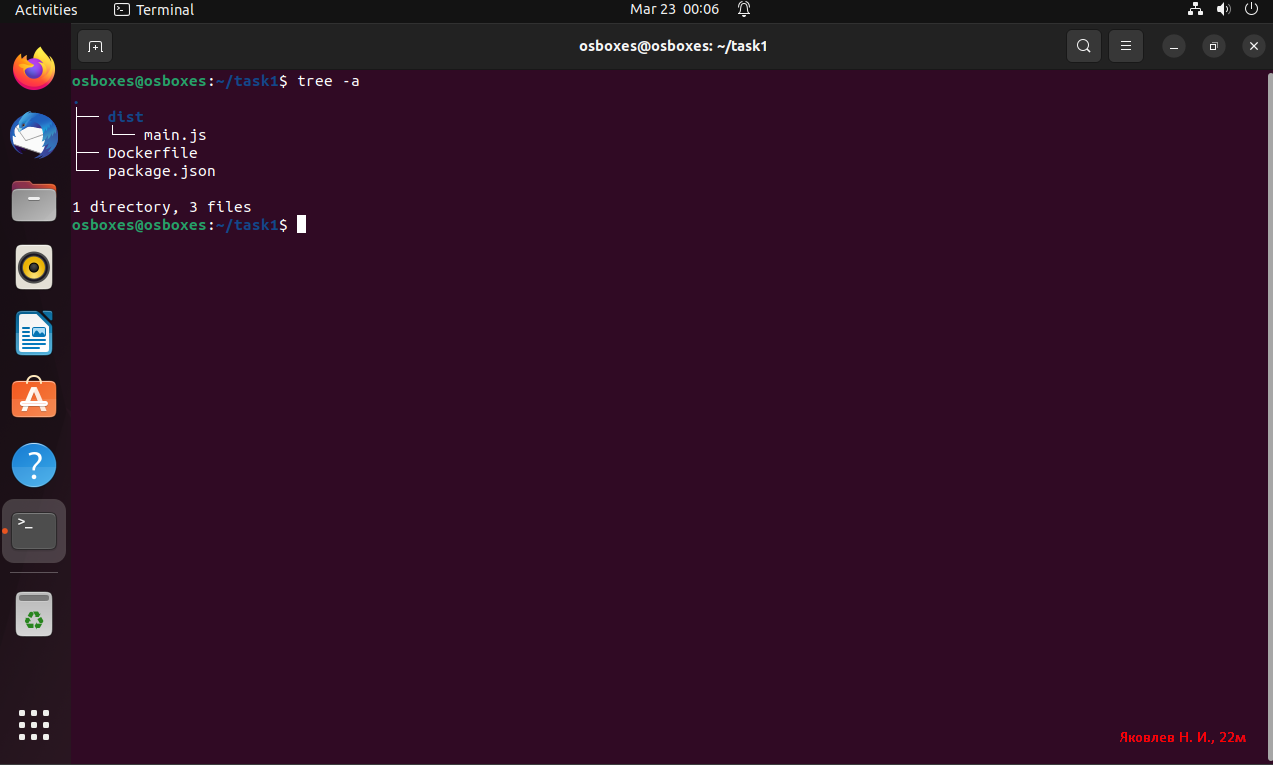


Рисунок  – Структура

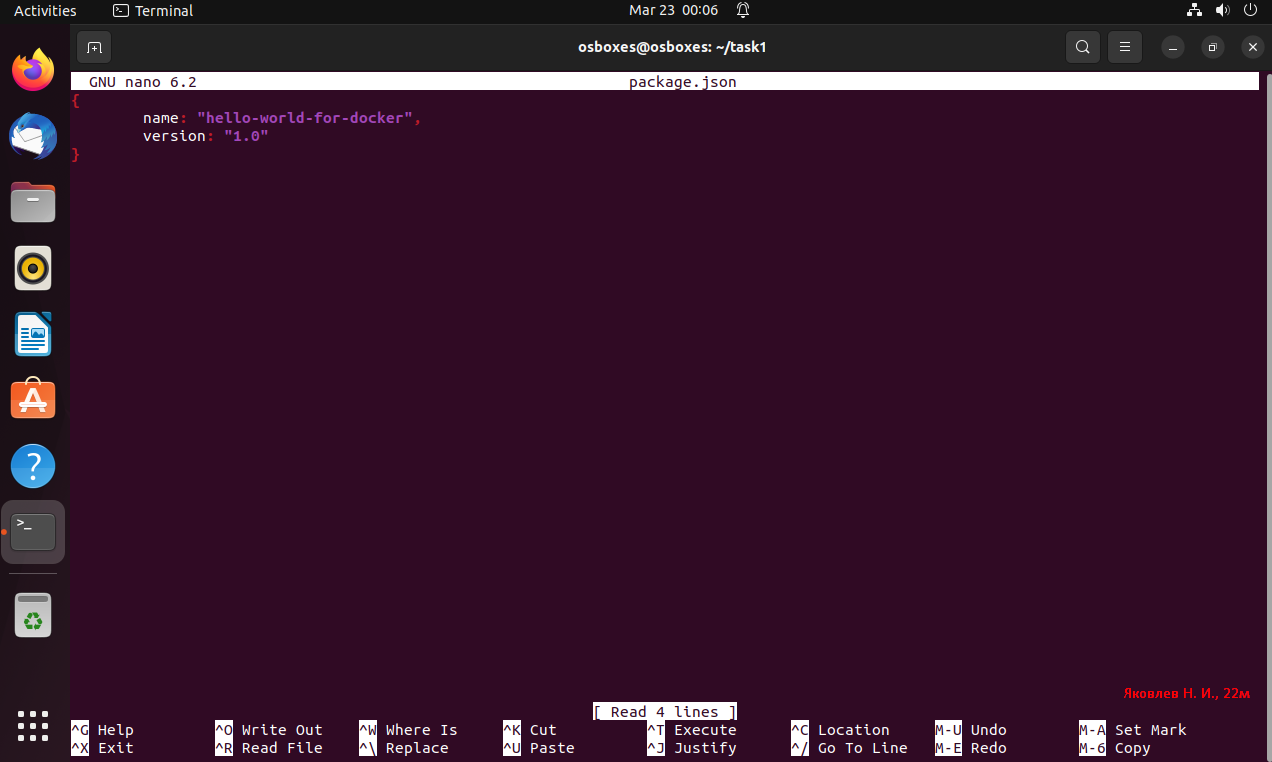


Рисунок  – Содержимое package.json

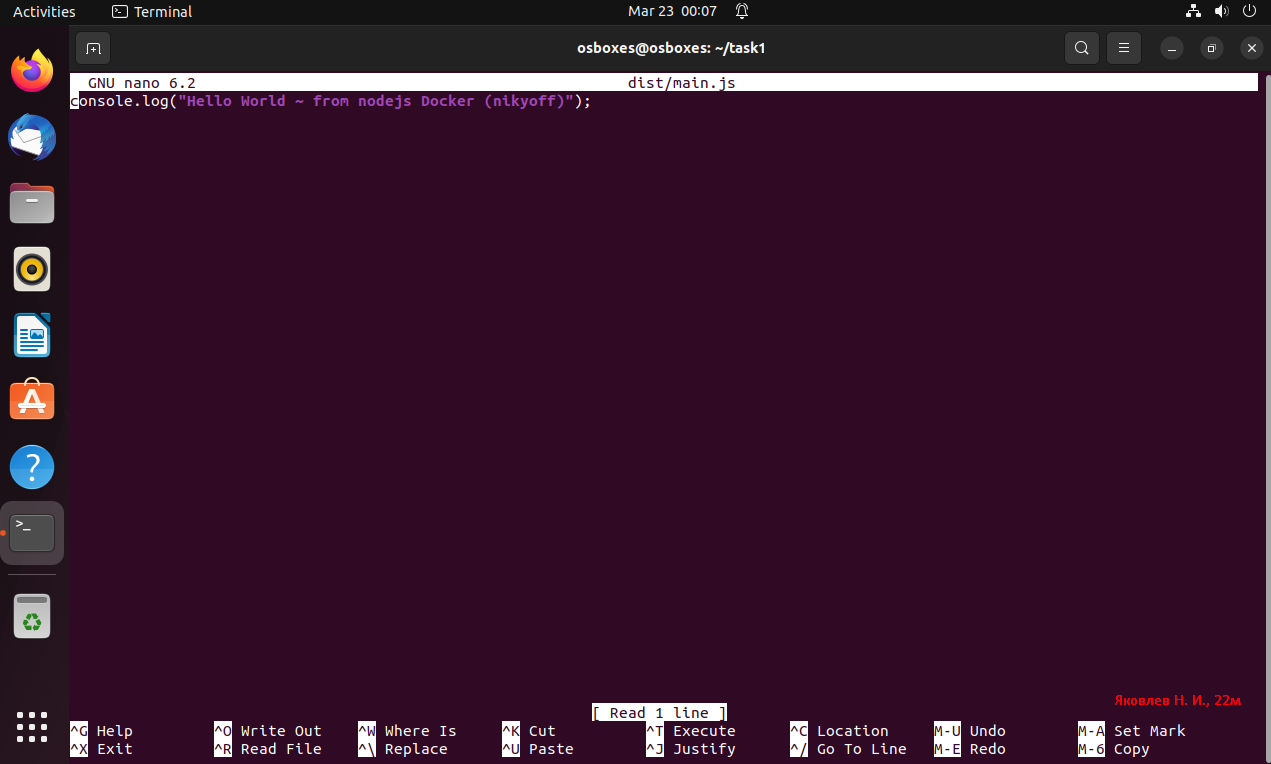


Рисунок  – Содержимое main.js

После необходимо собрать образ с помощью команды «docker build -t nikyoff/app:1.0», в данном случае флаг «-t» указывает создать образ с определенным именем и опциональным тегом формата «name:tag». Мы создаем образ с указанием имени автора/репозитория «nikyoff», названием образа «app» и тегом версии «1.0» (рисунок 14).

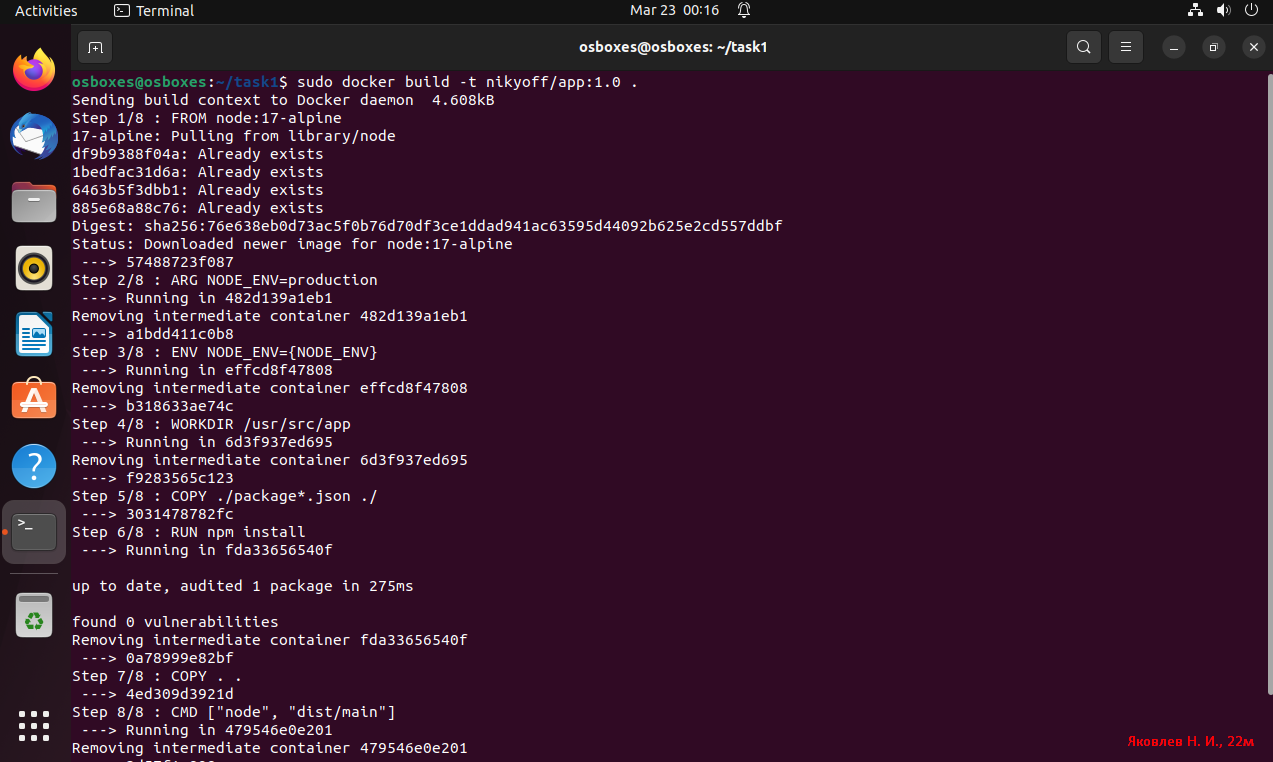


Рисунок  – Сборка образа

Пропишем команду для просмотра текущих образов Docker и увидим созданный нами образ (рисунок 15).

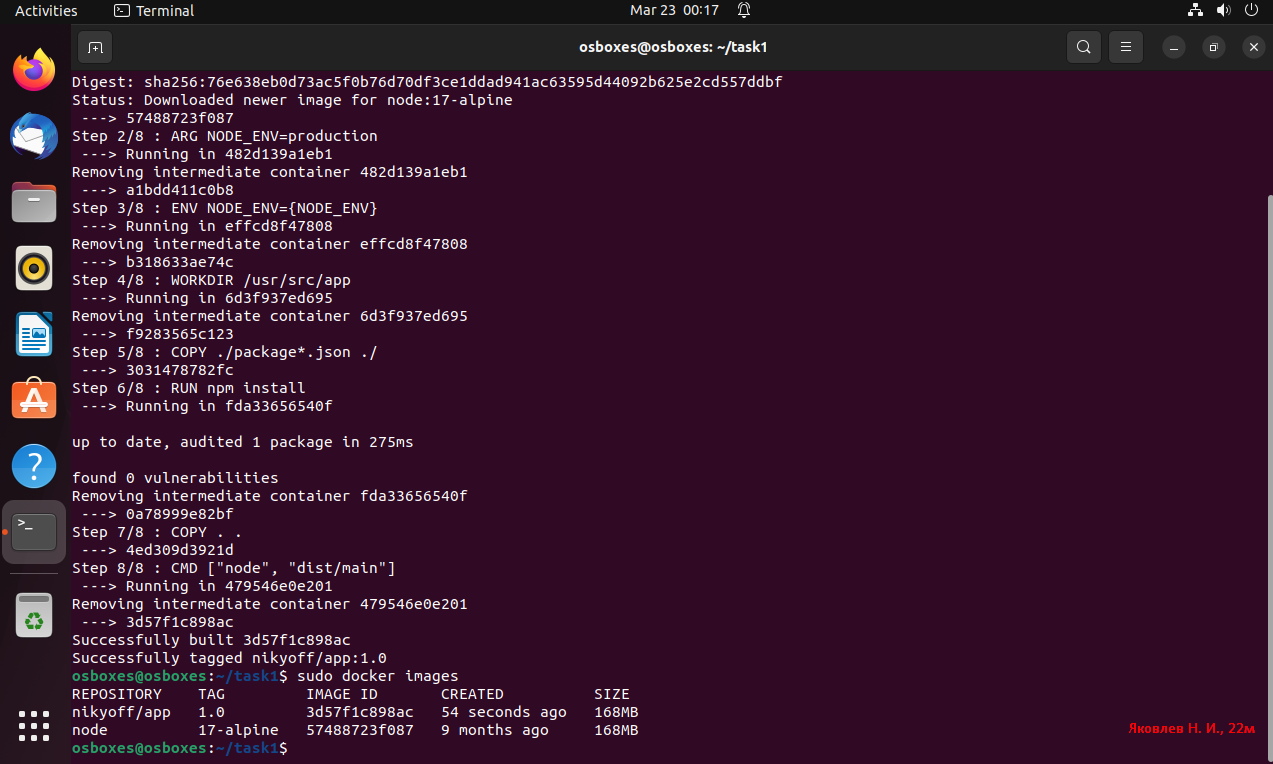


Рисунок  – Текущие образы Docker

Теперь остается только запустить наш образ с помощью команды «docker run nikyoff/app» (рисунок 16).

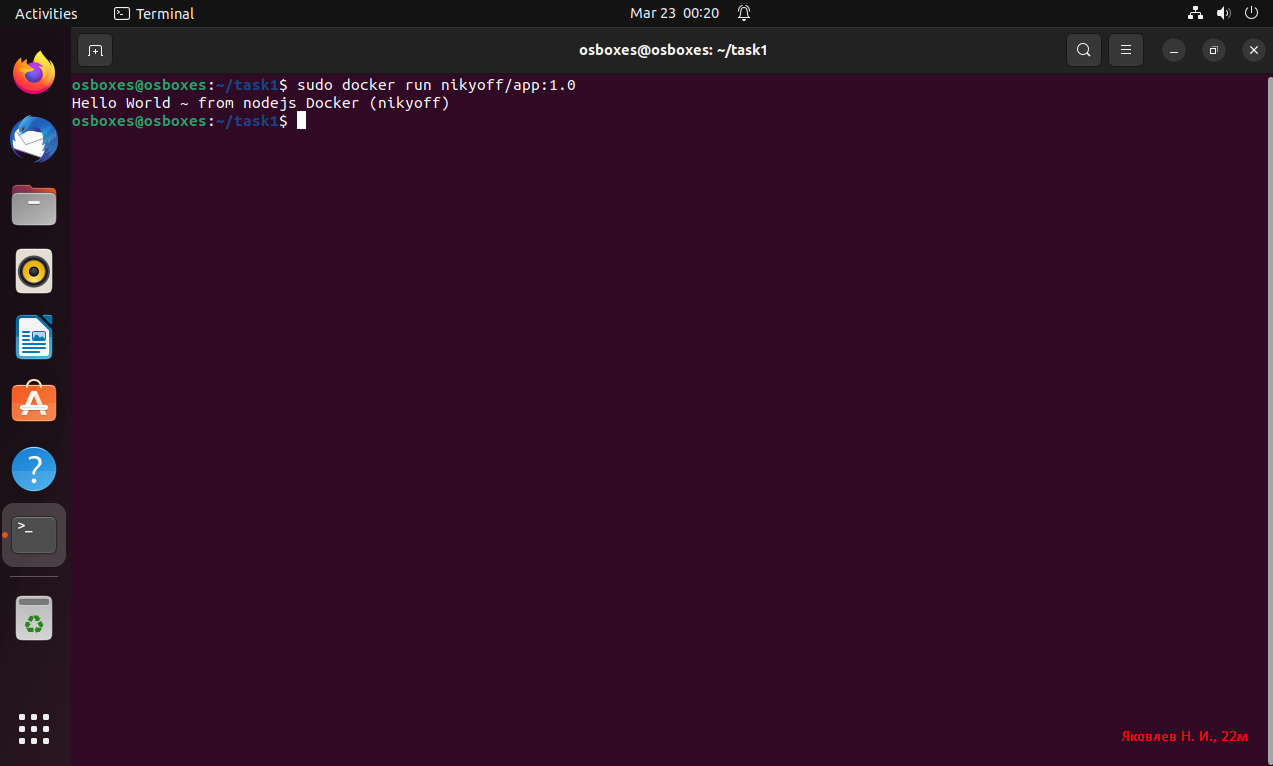


Рисунок  – Запуск нашего образа

## Архитектура Docker? Docker Hub?

На официальном сайте Docker прилагается следующая диаграмма (рисунок 20)

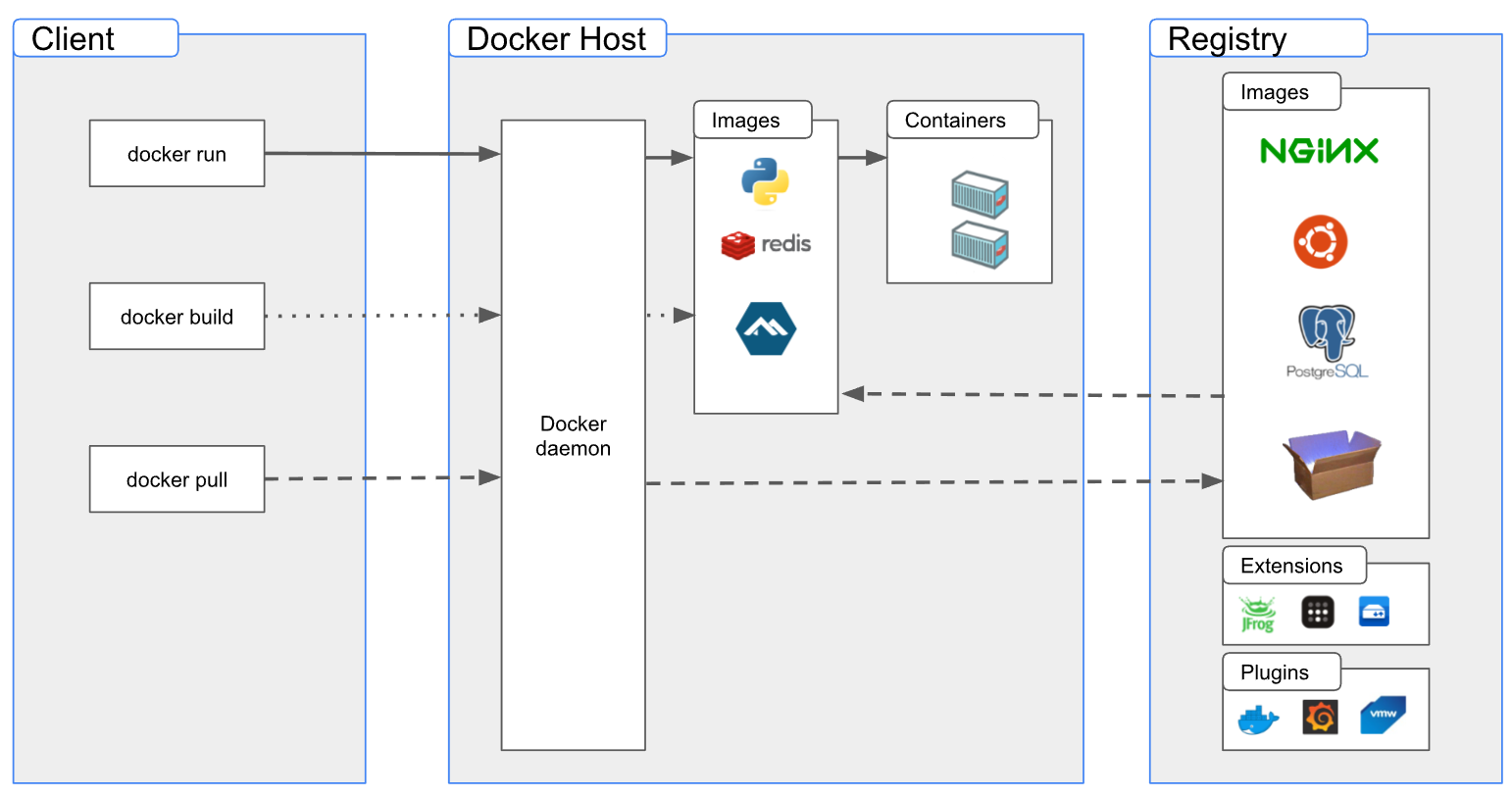


Рисунок  – Архитектура Docker

Как видно из диаграммы, архитектура разбивается на три составляющие: клиент, Docker хост и реестр. По своей сути нам представлена архитектура клиент-сервера, в которой клиент выполняет какие-либо запросы на сервер, в данном случае Docker Host при этом клиент может получить как удаленный так и локальный доступ к Docker Host. Внутри хоста мы видим Docker demon, который используется для того, чтоб клиент мог взаимодействовать со средой Docker. Реестр необходим для того, чтоб хранить какие-либо удаленные образы, расширения, плагины и.т.д., при этом реестры бывают как публичные (Docker Hub), так и приватные. Заглядывая глубже в сам Docker, мы видим, что он состоит из образов (Images) и контейнеров (Containers). Образ в Docker представляет из себя некоторый шаблон, который используется для создания контейнеров. Контейнер представляет из себя изолированную платформу, именно по этой причине они являются безопасными, ведь доступ к какой-либо части контейнера извне нужно сначала настроить/открыть. Над образами и контейнерами могут выполняться различные команды-операций: создание, удаление, контейнеры можно запускать, останавивать и.т.д.

## Выполните лабу в Docker Desktop через приложение для Windows

Создадим образ «hello-world», для этого необходимо выполнить команду «docker run hello-world» (рисунок 21).

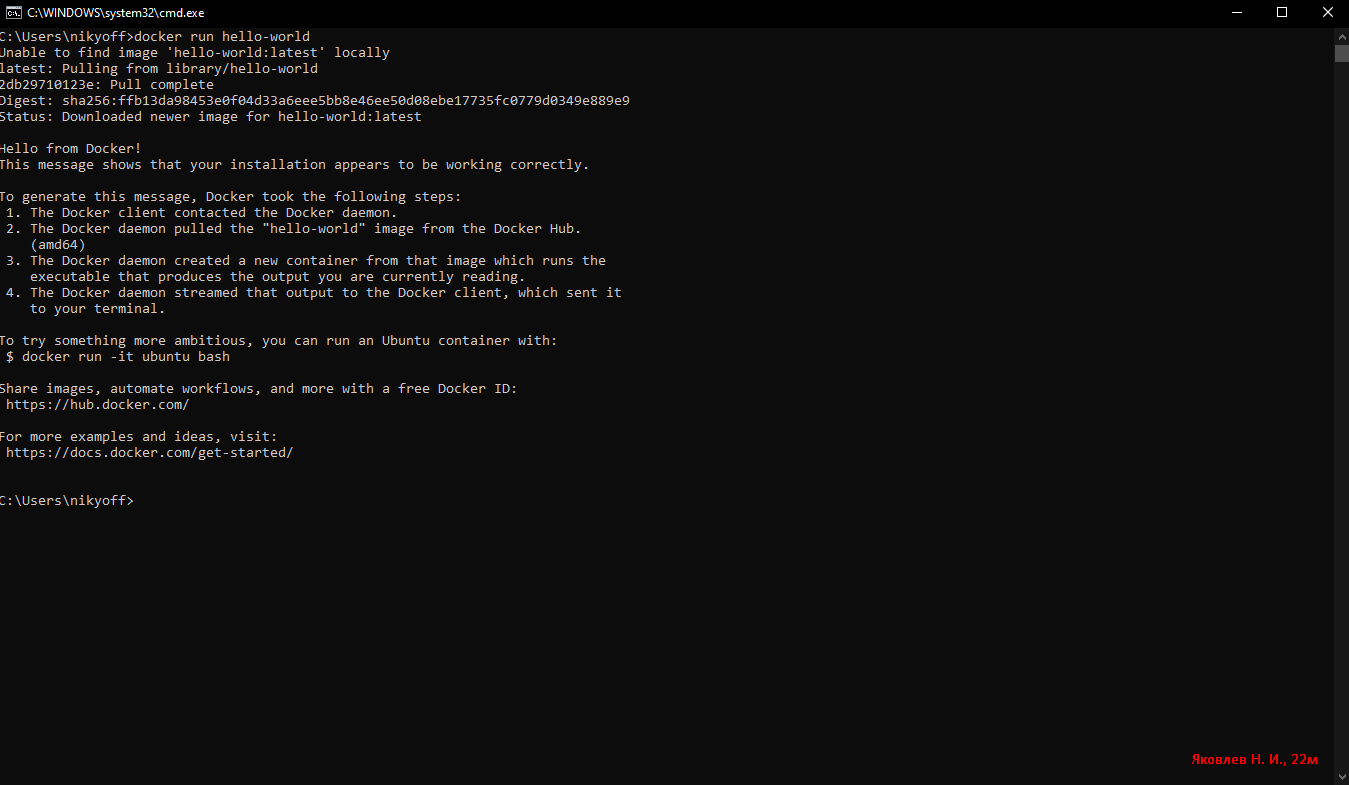


Рисунок  – Образ «hello-world»

Посмотрим текущие обазы, для этого необходимо выполнить следующую команду «docker images» (рисунок 22).

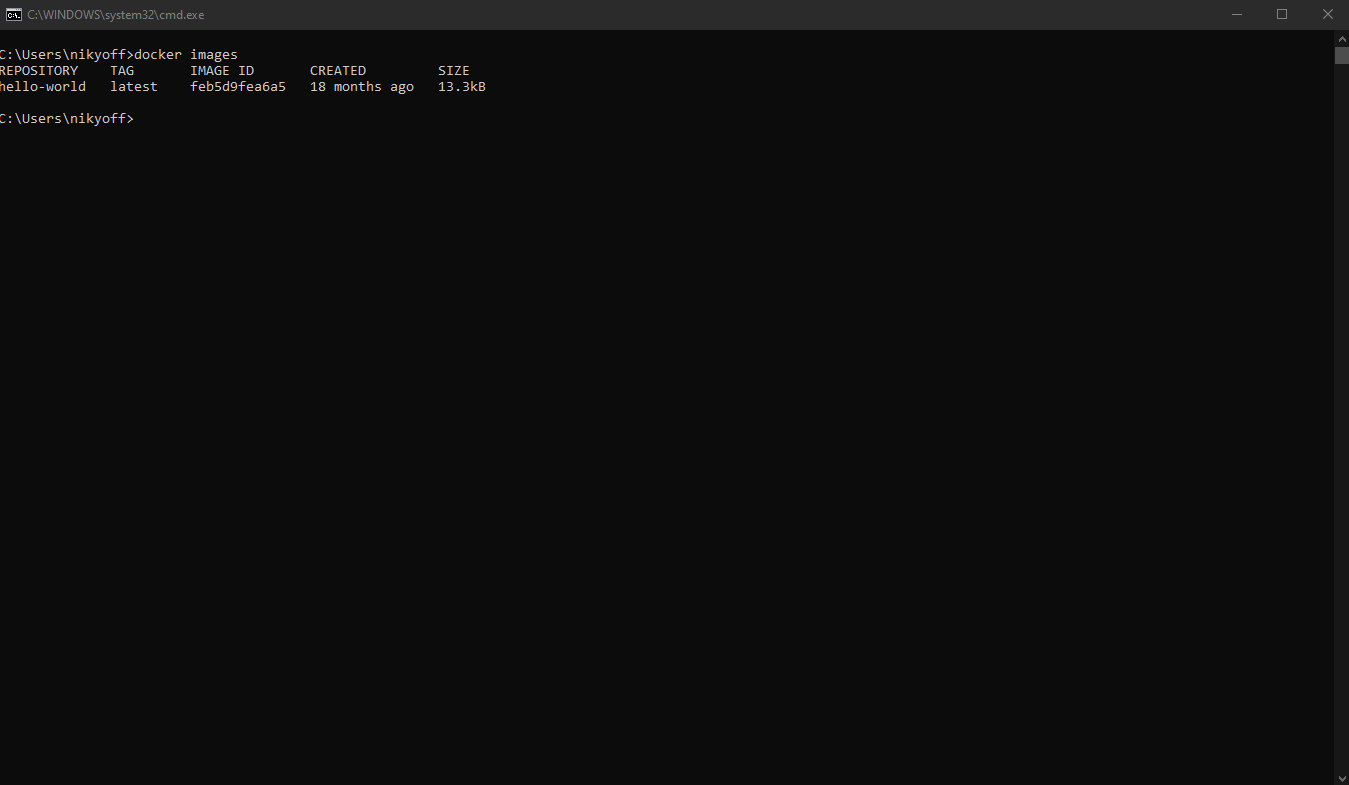


Рисунок  – Команда для просмотра образов

Попробуем удалить образ с помощью команды «docker image rm {id}» (рисунок 23).

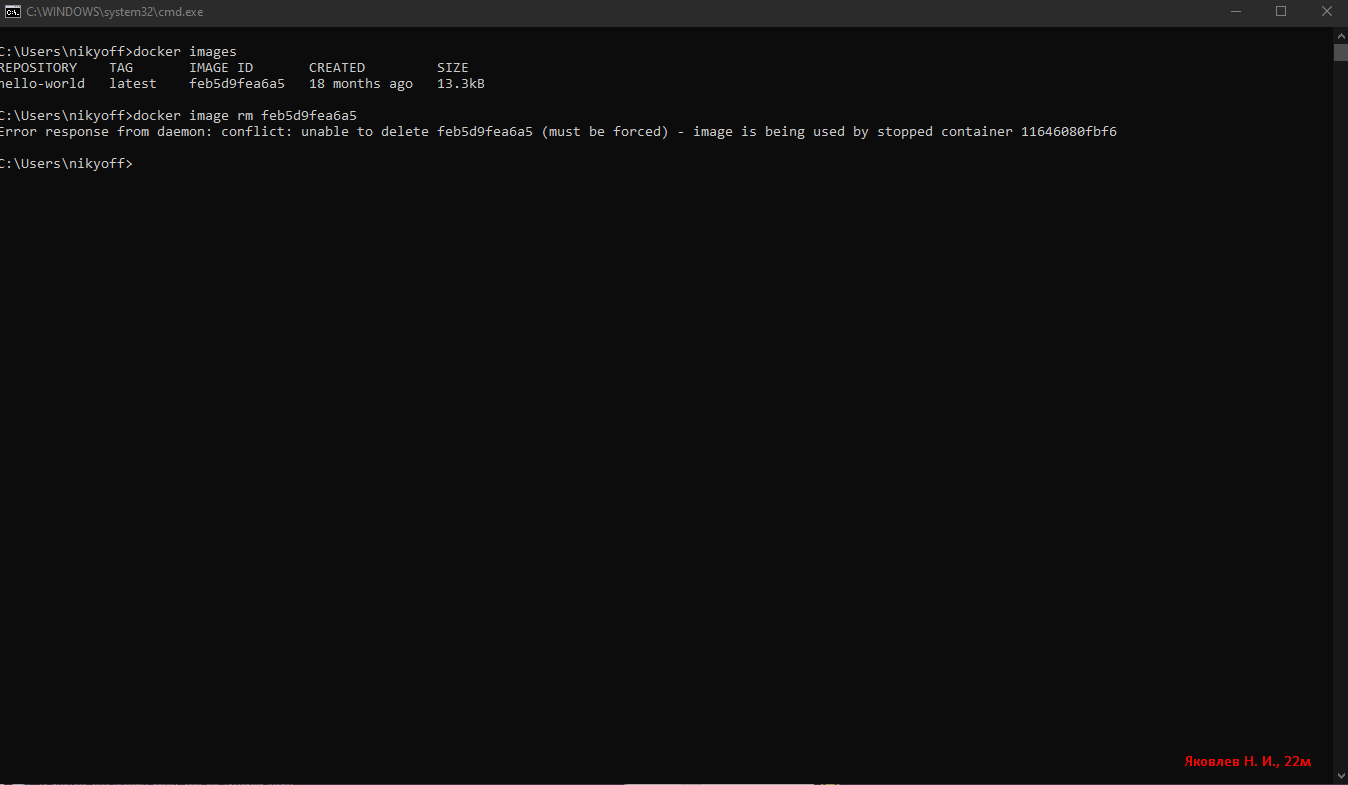


Рисунок  – Команда для удаления образа

В результате получим ошибку, выполним аналогичные действия из лабораторной работы в ОС Linux (рисунок 24).

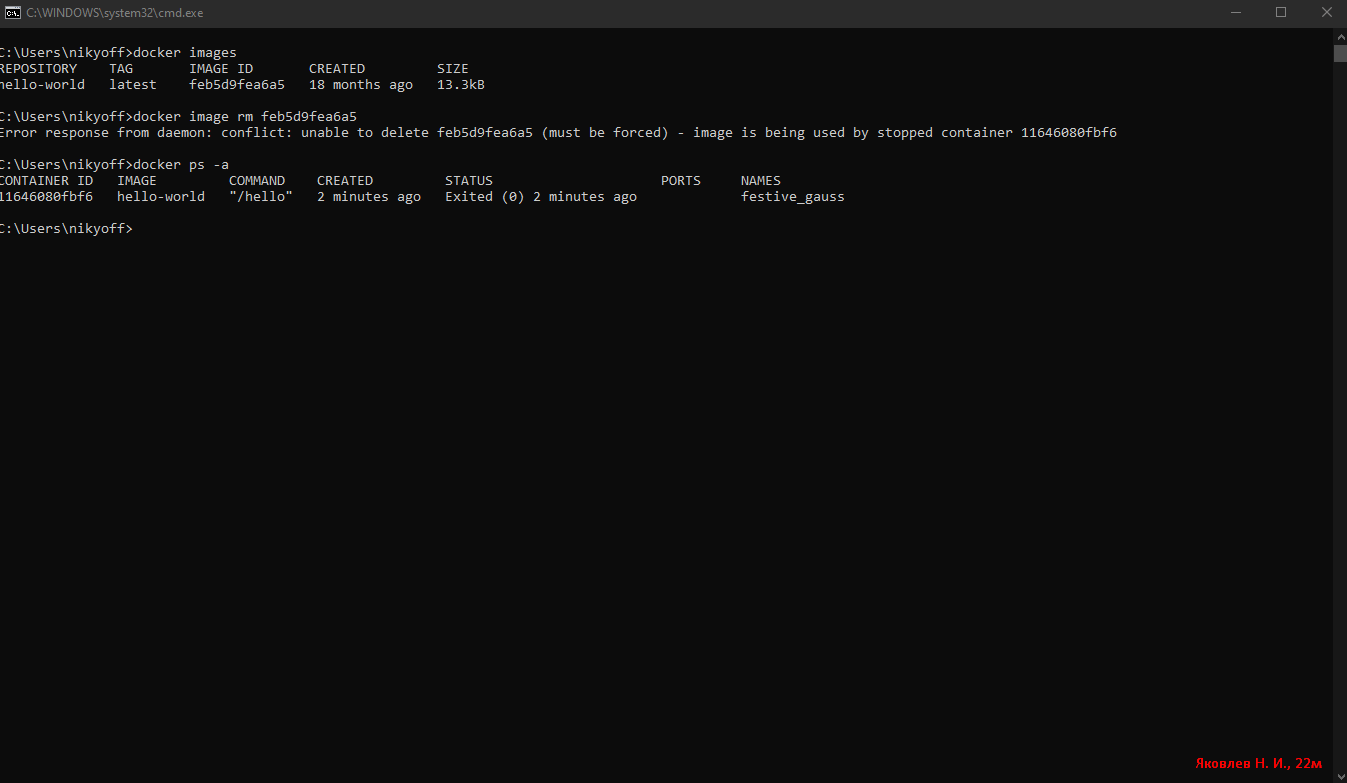


Рисунок  – Команда для просмотра контейнером

Затем удалим контейнер с помощью команды «docker container rm {id}» (рисунок 25).

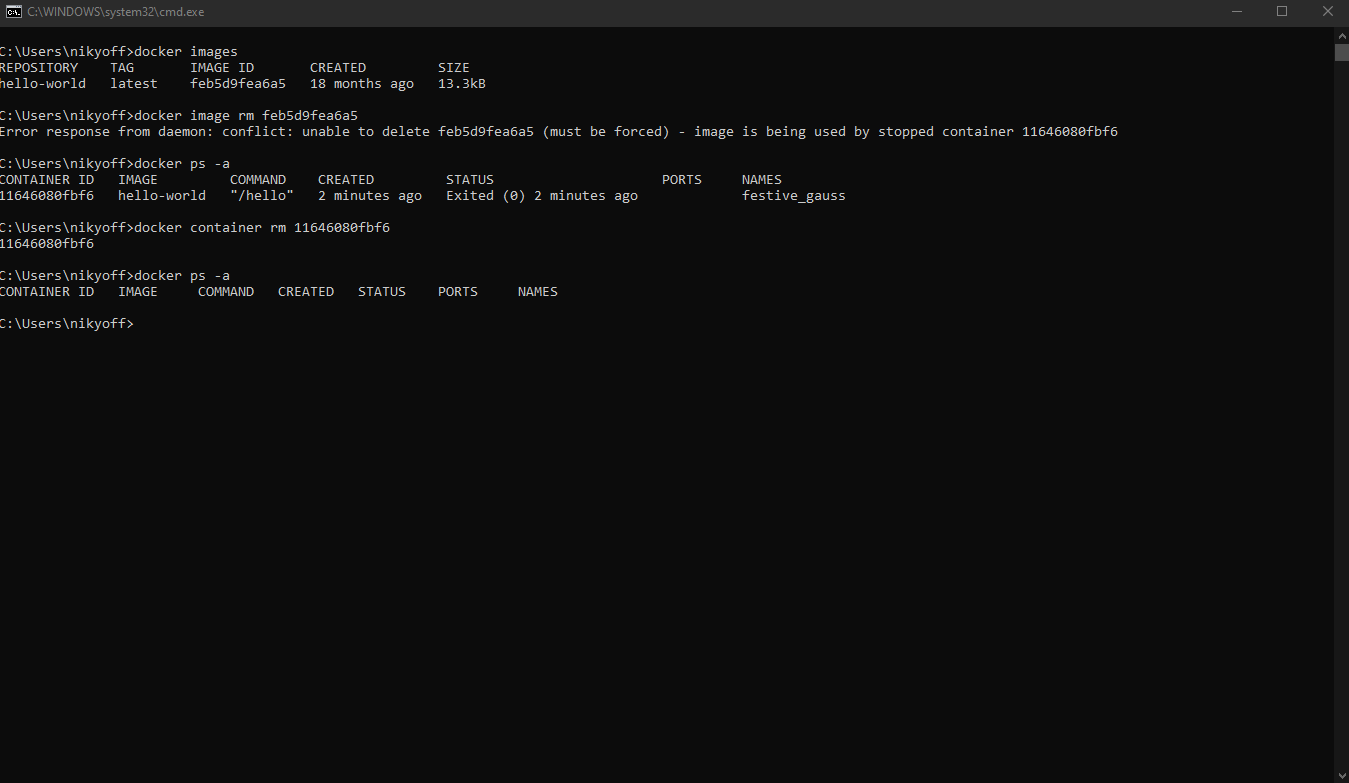


Рисунок  – Команда для удаления контейнера

Теперб удалим ранее созданный образ «hello-world» (рисунок 26).

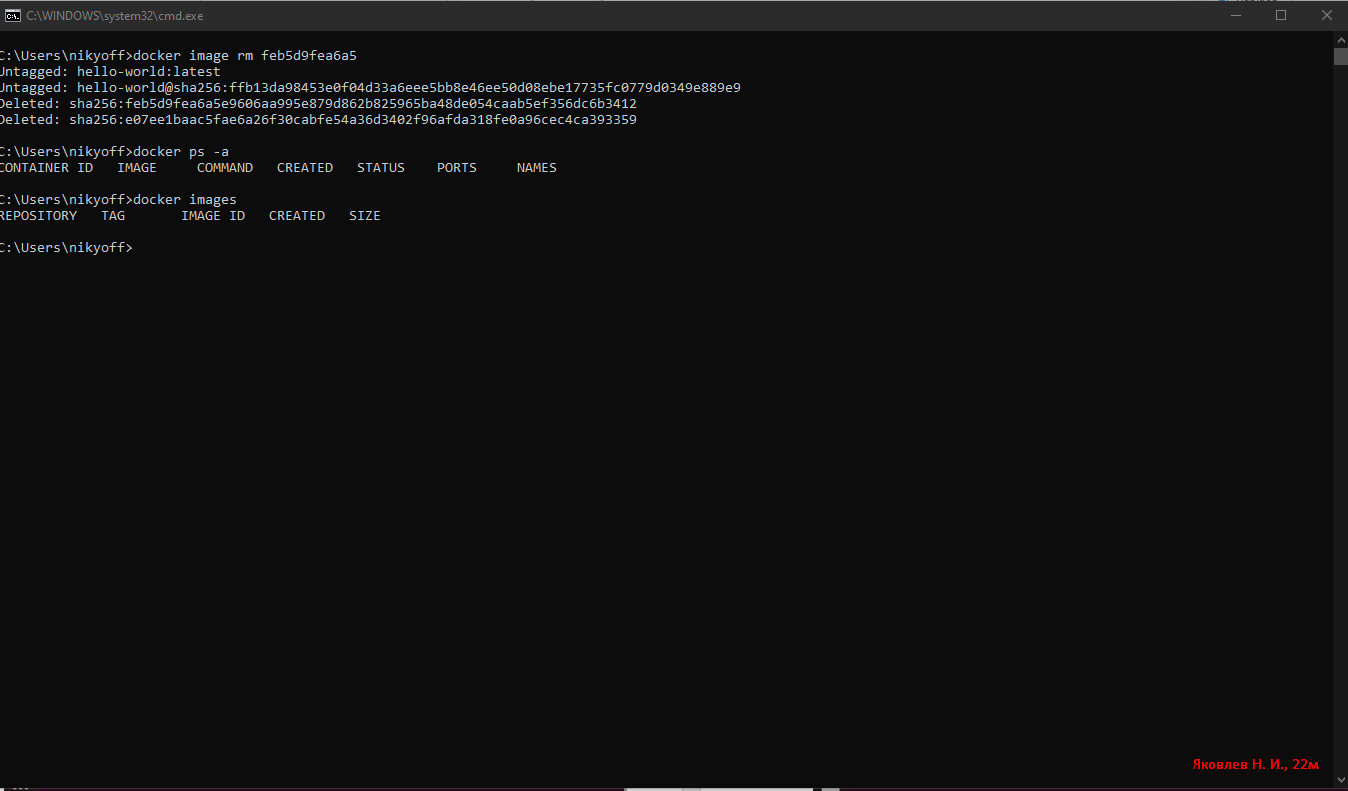


Рисунок  – Удаление образа

Установим образ ОС «alpine» с помощью команды «docker pull alpine» (рисунок 27).

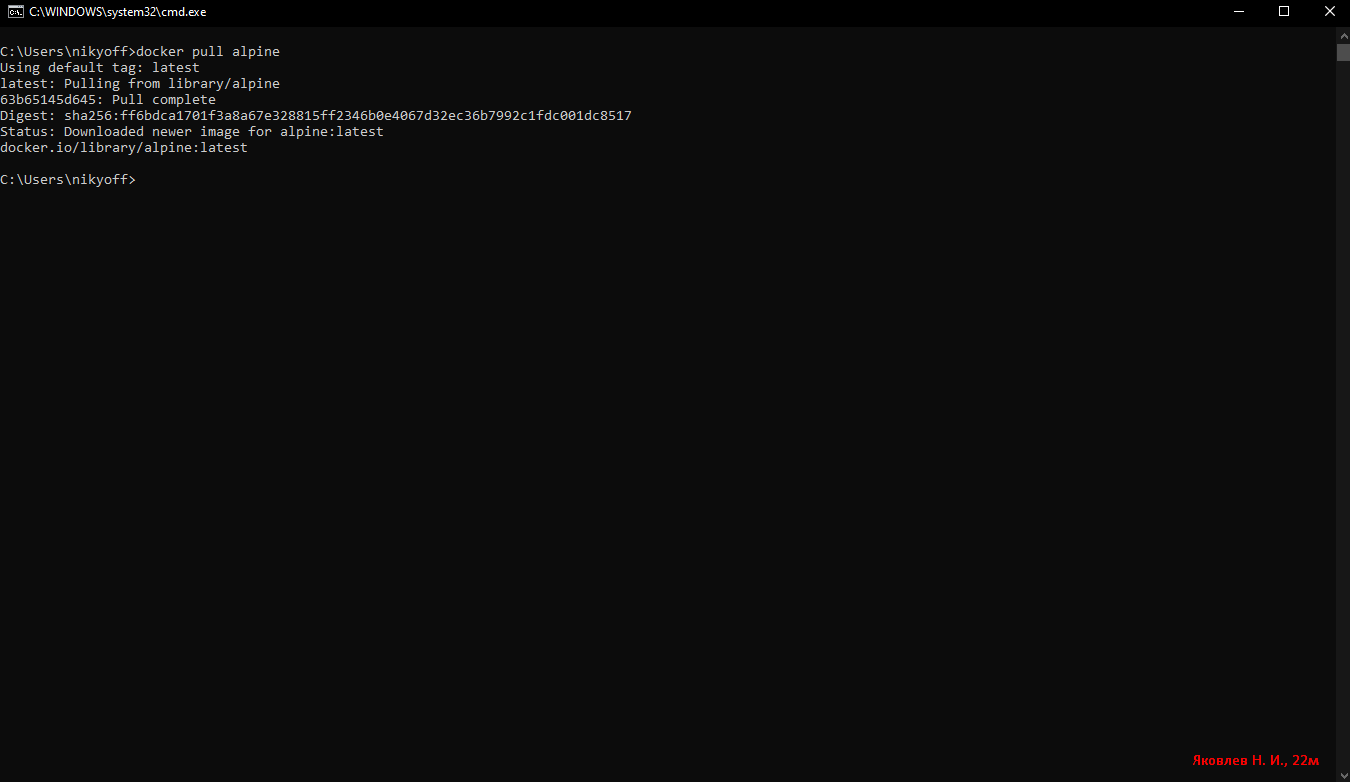


Рисунок  – Команда для установки образа

Запустим созданный образ с помощью команды «docker run -it alpine». После запуска контейнера, мы можем вызывать команды-косноли из нашего терминала внутри контейнера, выполним команды из последнего раздела лабораторной работы (рисунок 28).

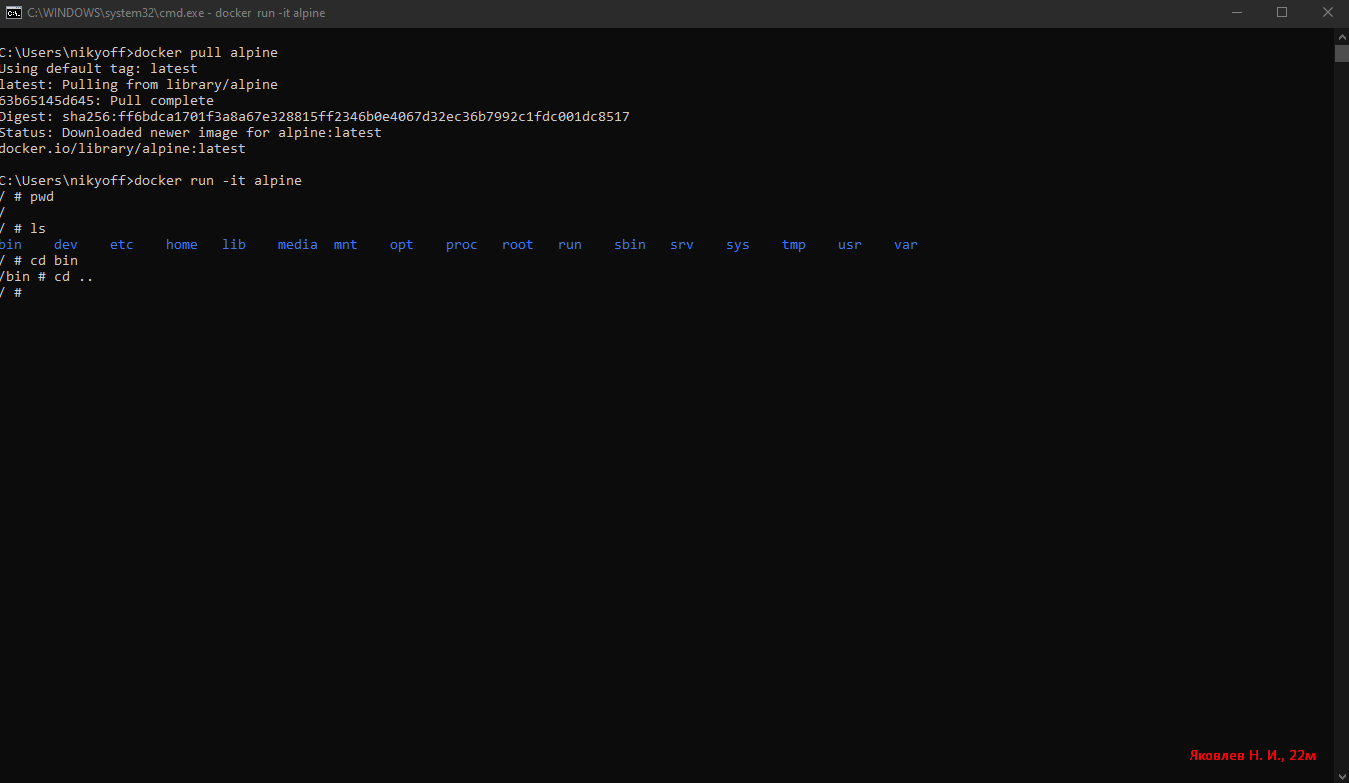


Рисунок  – Работа с терминалом внутри контейнера

Остановим контейнер с помощью команды «docker container stop {id}» (рисунок 29).

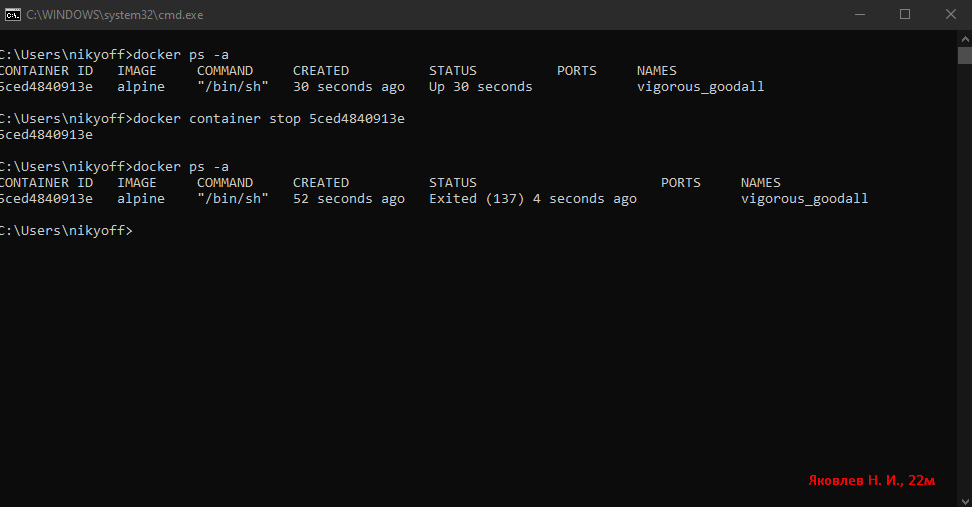


Рисунок  – Команда для остановки контейнера

Вывод

В данной лабораторной работе, мы познакомились с программой docker, изучили как скачивать образы и базовые возможности работы с ними, а также познакомились с базовыми возможностями работы с контейнерами.