|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 «Системное программное обеспечение»

**Тема практической работы:** «Docker»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИВБО-11-19 | Давлетбаева Алина Ринатовна |
| **Проверил:** | ассистент Воронцов Ю.А. |

2021 г.

**Содержание**

**[Часть 3.1. Образы](#_Toc73794359)** [3](#_Toc73794359)

[**Часть 3.2. Изоляция** 4](#_Toc73794360)

[**Часть 3.3. Работа с портами** 6](#_Toc73794361)

[**Часть 3.4. Именованные контейнеры, остановка и удаление** 10](#_Toc73794362)

[**Часть 3.5. Постоянное хранение данных** 12](#_Toc73794363)

[**Часть 3.5.1 Тома** 14](#_Toc73794364)

[**Часть 3.5.2 Монтирование директорий и файлов** 15](#_Toc73794365)

[**Часть 3.6 Переменные окружения** 15](#_Toc73794366)

[**Часть 3.7 Dockerfile** 16](#_Toc73794367)

[**Часть 3.8 Индивидуальное задание** 18](#_Toc73794368)

**Практическая работа №3**

**По дисциплине «Системное программное обеспечение»**

**На тему «Docker»**

**Цель работы:**

Получить навыки работы с docker.

# **Часть 3.1. Образы**

[**Docker**](https://www.docker.com/) –популярное приложение для контейнеризации программ. Такие контейнеры – это, по сути, виртуальные машины с высокой портативностью, которые более рационально используют ресурсы и в большей степени зависят от операционной системы.

Основной сайт сервиса — [play-with-docker.com](http://play-with-docker.com/), который после прохождения CAPTCHA пересылает на один из облачных хостов, где стартует 4-часовая сессия «игровой площадки» (playground). В ней вы можете создавать новые сущности (instances), т.е. узлы тестового Docker-кластера. Каждый из них — это инсталляция легковесного дистрибутива Alpine Linux (на данный момент версии 3.6.2 с ядром Linux 4.4.0) с редактируемым локальным IP-адресом. В них установлен Docker актуальной версии [17.06 Community Edition](https://habrahabr.ru/company/flant/blog/332160/).

1. Регистрируемся на id.docker.com
2. Заходим на labs.play-with-docker.com
3. Входим через Docker ID: id.docker.com

Login: liebeallina11

1. Нажимаем «Старт»
2. Создаём новый инстанс (будет доступен с момента создания в течении 4-х часов), кликнув по «Add new instance»:

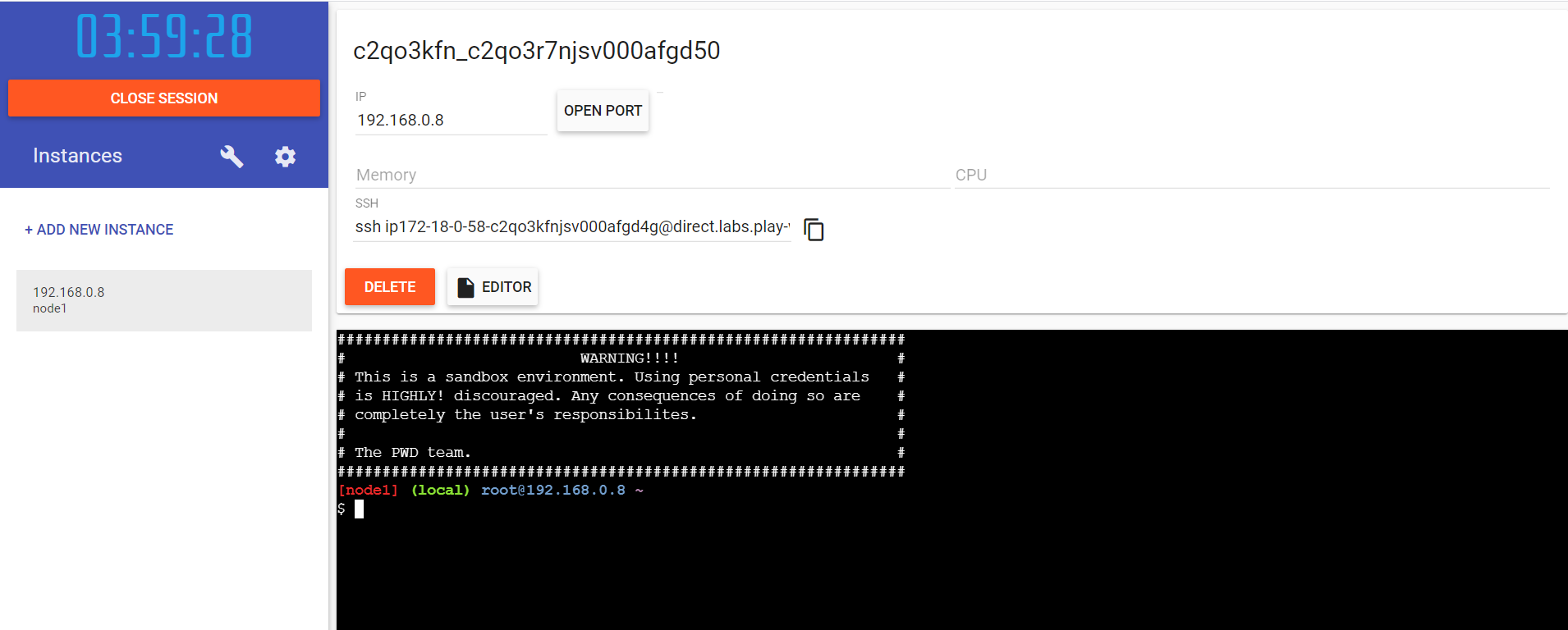
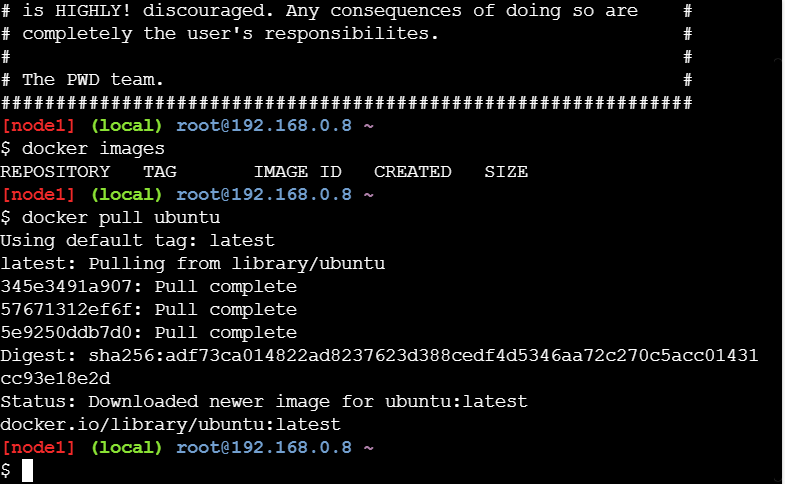
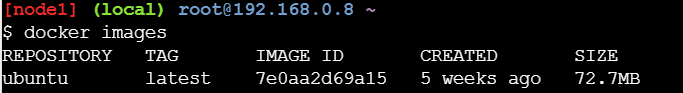


Рис.1. Добавление нового узла тестового Docker-кластера

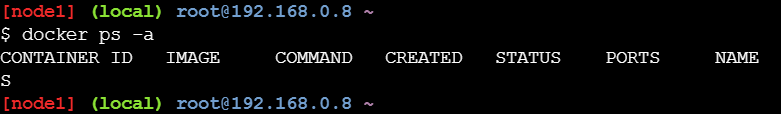
1. В разделах, где есть текст про открыть <http://0.0.0:8000/> нужно будет кликнуть на порт вверху страницы.
2. Загрузили образ docker pull Ubuntu



1. Проверка имеющихся образов docker images



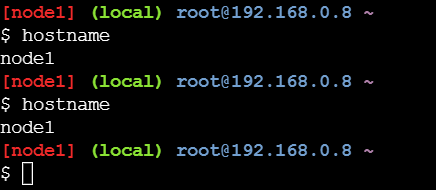
1. Проверка списка контейнеров docker ps –a



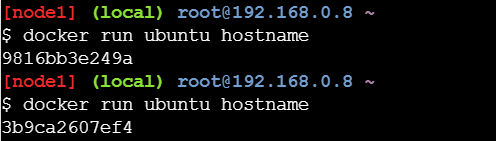
# **Часть 3.2. Изоляция**

1. Посмотрим информацию о хостовой системе, выполнив команду hostname.

**Хост** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *host* — «владелец, принимающий гостей») — любое [устройство](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE), предоставляющее сервисы формата «[клиент-сервер](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80)» в режиме сервера по каким-либо [интерфейсам](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и уникально определённое на этих интерфейсах. В более широком смысле под хостом могут понимать любой [компьютер](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), подключённый к [локальной](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) или [глобальной](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) [сети](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C).

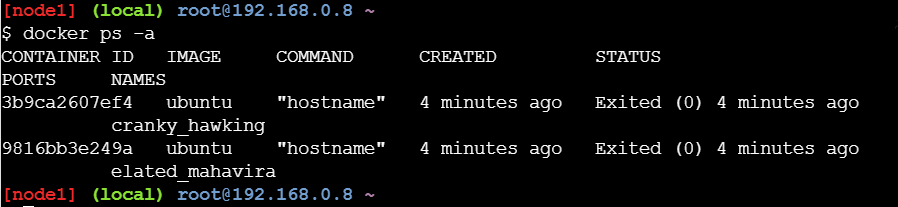


1. Посмотрим информацию о хостовой системе в контейнерах. Выполним два раза команду docker run ubuntu hostname:



При разных запусках команды в контейнерах, ответ будет отличаться, будет разный hostname. Так происходит, потому что из одного образа Ubuntu были запущены два изолированных контейнера, поэтому у них и был разный hostname.

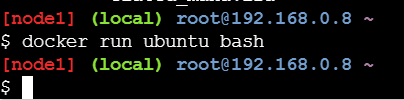
1. Выполним вновь docker ps –a. Появятся запущенные ранее контейнеры.



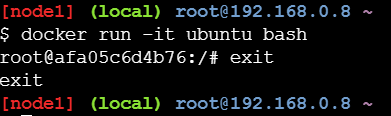
1. Запуск контейнеров командой:

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

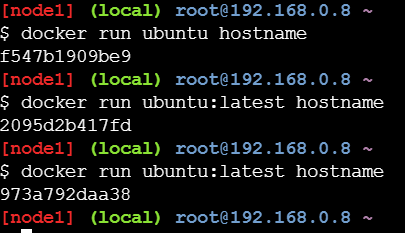
Запуск bash в контейнере: docker run Ubuntu bash. Ничего не произошло.



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

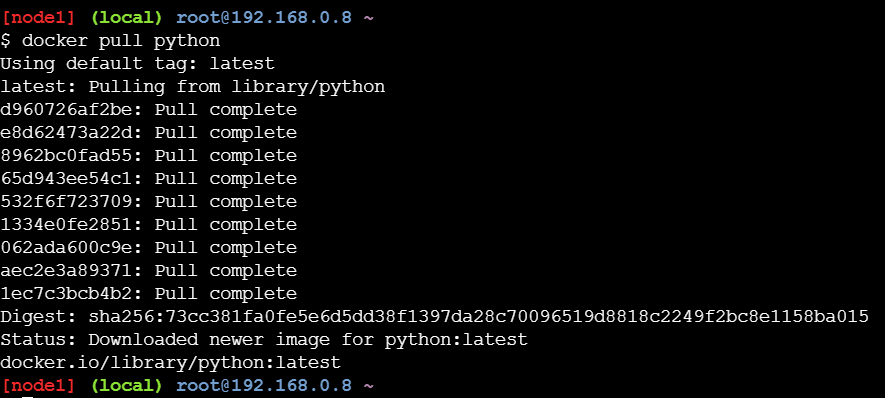


Выполняя запуск контейнера, указывая образ Ubuntu, неявно указывался образ ubuntu:latest. Следовательно команды docker run ubuntu hostname и docker run ubuntu:latest hostname равнозначны.

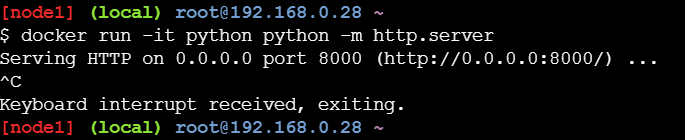


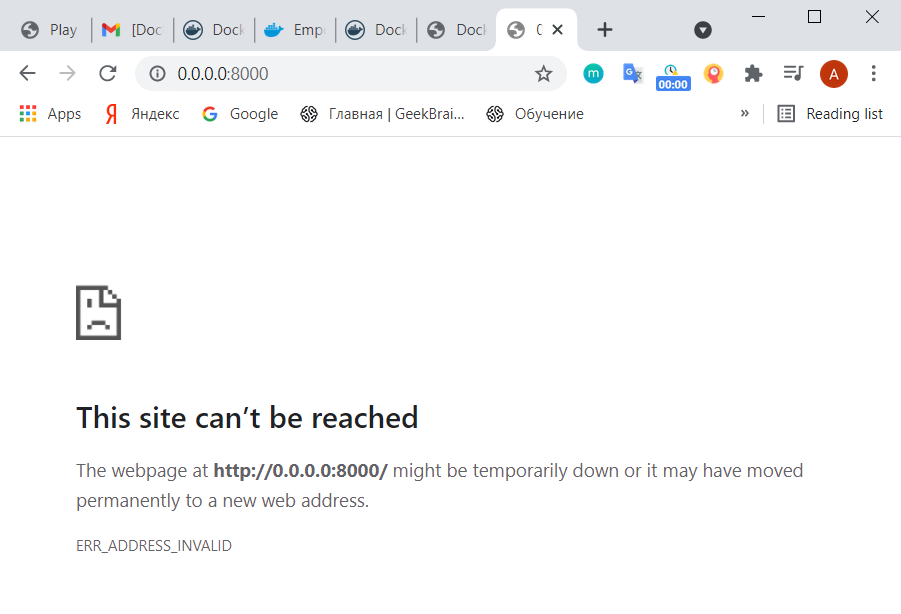
# **Часть 3.3. Работа с портами**

1. Загрузим образ python командой docker pull python



1. В качестве примера, запустим встроенный в Python модуль веб-сервера из корня контейнера, чтобы отобразить содержание контейнера. Docker run –it python –m http.server (у меня был перезапуск узла, поэтому хост поменялся на 28):

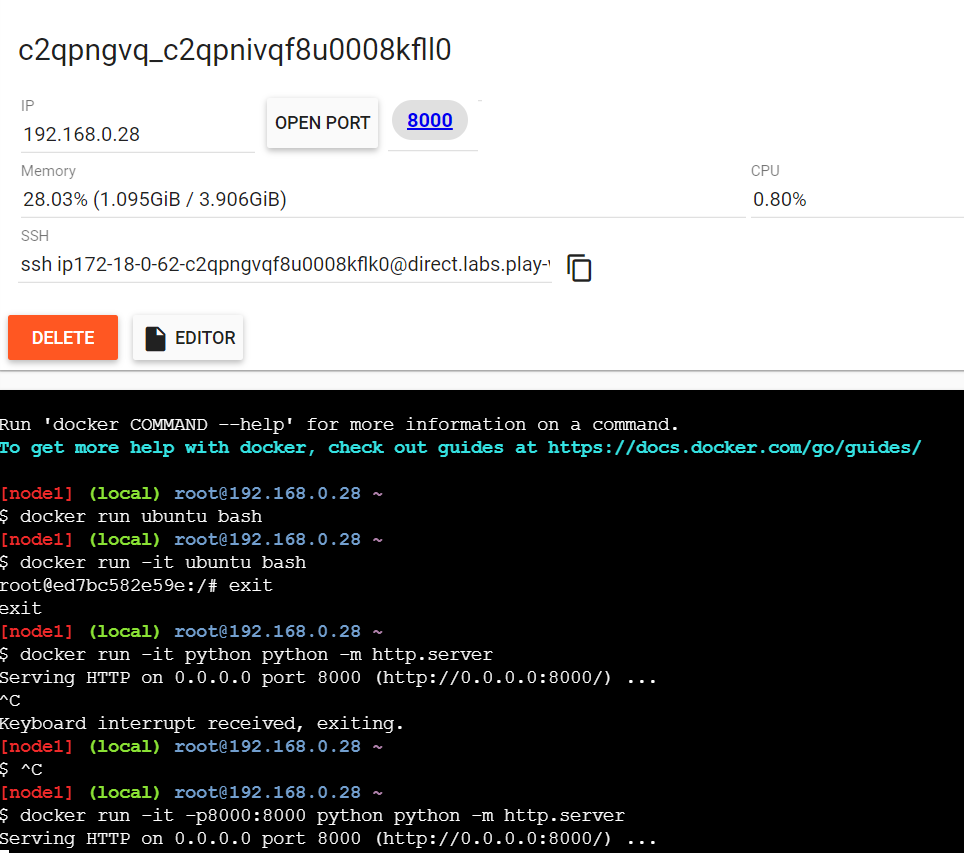


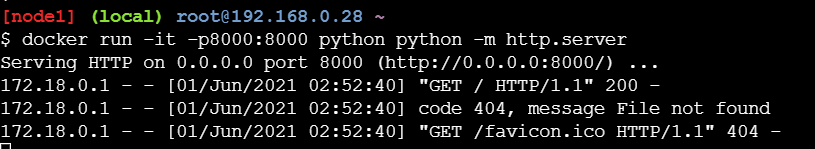


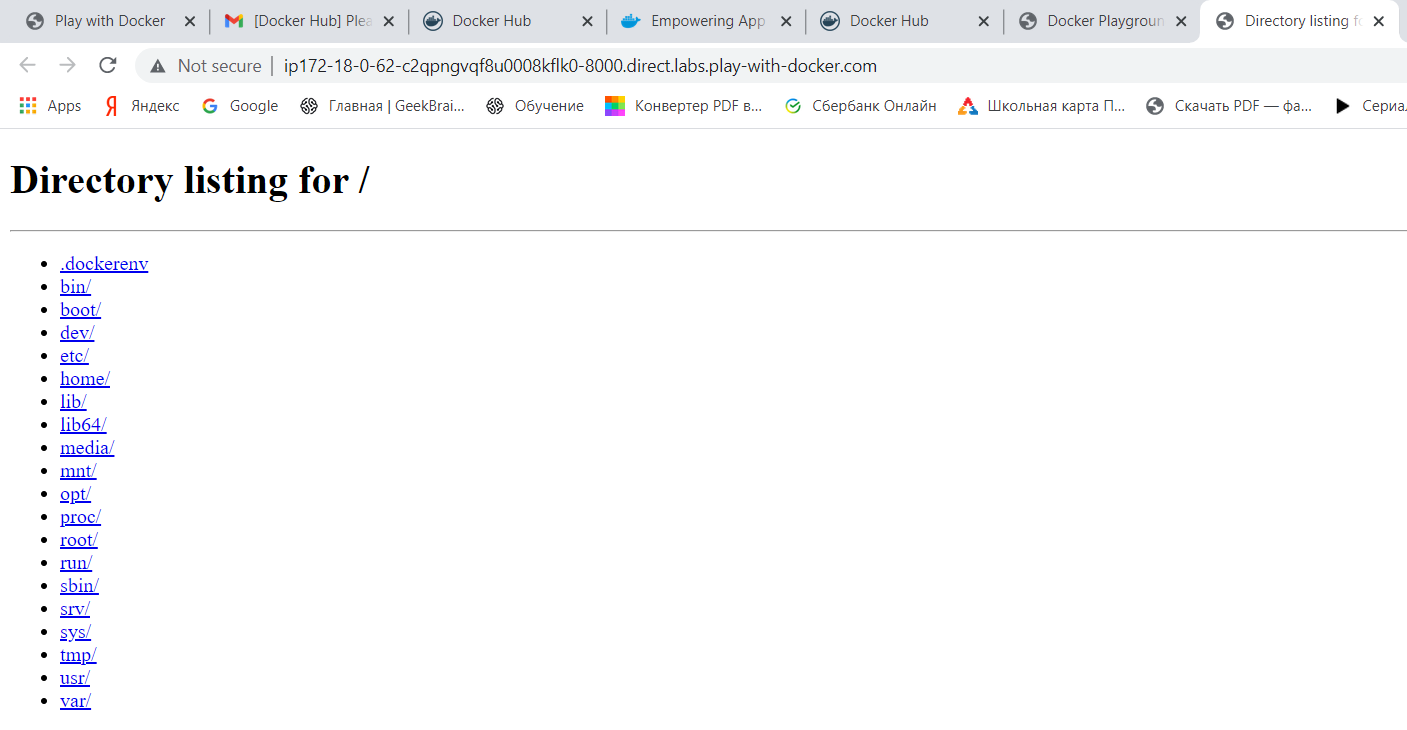
Для проброса портов используется флаг –p hostPort:containerPort

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/> открывается содержимое корневой директории в контейнере.

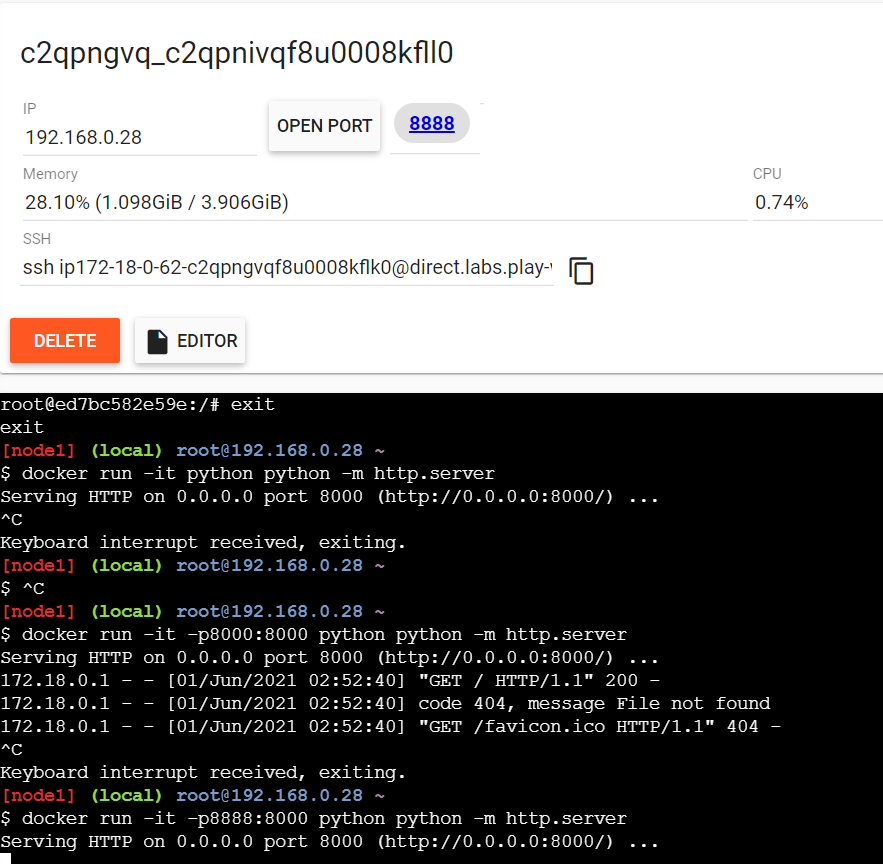


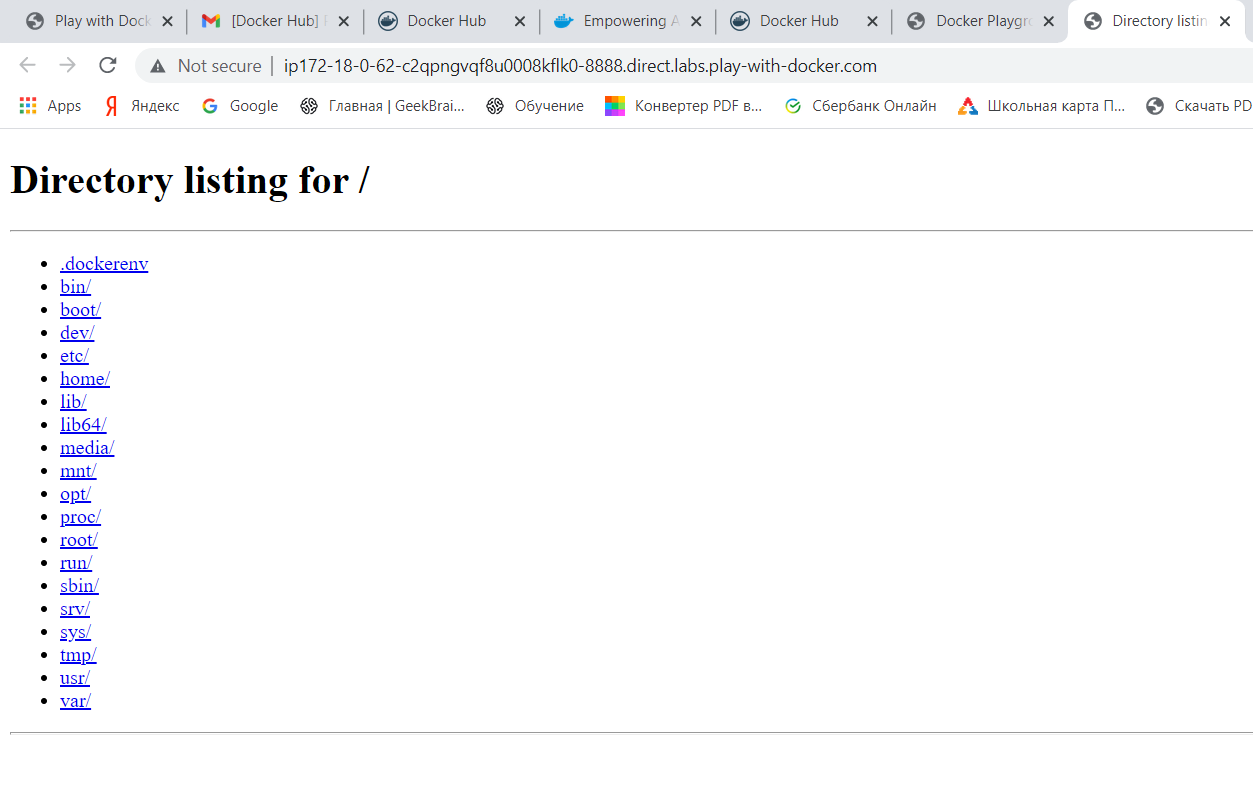


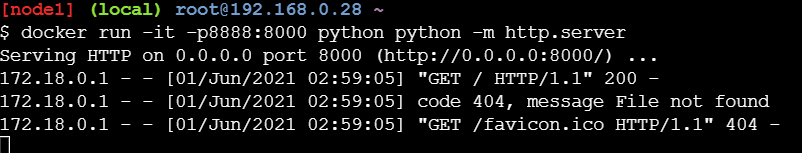


Для того, чтобы доступный в контейнере на порту 8000 веб-сайт в хостовой системе открывался на порту 8888, необходимо указать флаг –p 8888:8000:

docker run –it –p8888:8000 python python –m http.server

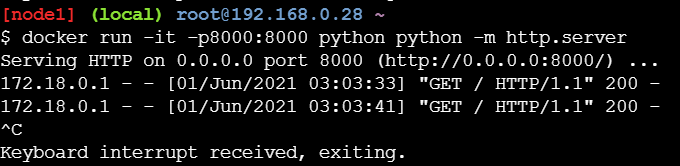






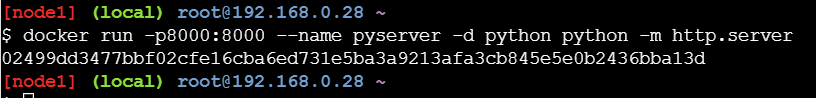
# **Часть 3.4. Именованные контейнеры, остановка и удаление**

1. Запустить контейнер: docker run –it –p8000:8000 python python –m http.server. Завершить.



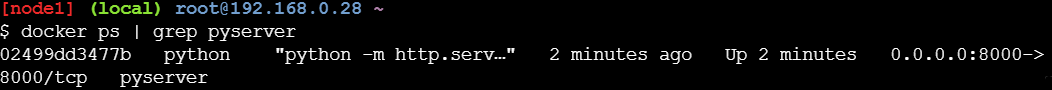
Для того, чтобы запустить контейнер в фоне, нужно добавить флаг –d/--detach. Также определим имя контейнеру, добавив флаг --name.

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



1. Убедиться, что контейнер всё ещё запущен: docker ps | grep pyserver.

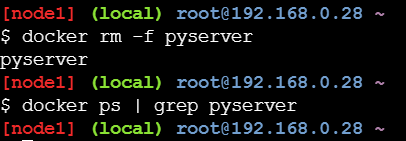
**grep** — [утилита](https://ru.wikipedia.org/wiki/%D0%A3%D1%82%D0%B8%D0%BB%D0%B8%D1%82%D1%8B) [командной строки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8), которая находит на вводе строки, отвечающие заданному [регулярному выражению](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), и выводит их, если вывод не отменён специальным ключом.

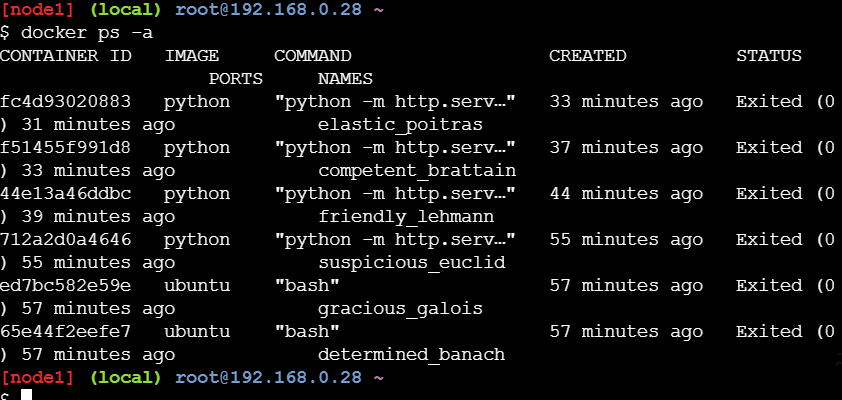


1. Для того, чтобы остановить выполнение контейнера, существует команда docker stop pyserver.

Удаление docker rm pyserver.

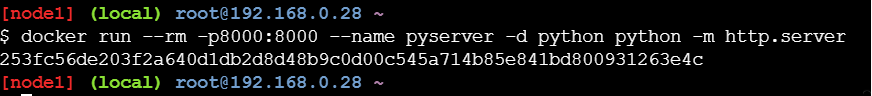
Для остановки и удаления контейнера можно воспользоваться командой docker rm –f pyserver вместо выполнения двух отдельных команд. После удаления контейнер с таким именем можно будет создать заново.





1. 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



# **Часть 3.5. Постоянное хранение данных**

1. Запустить контейнер, в котором веб-сервер будет отдавать содержимое директории /mnt: docker run –p8000:8000 –name pyserver –rm –d python

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

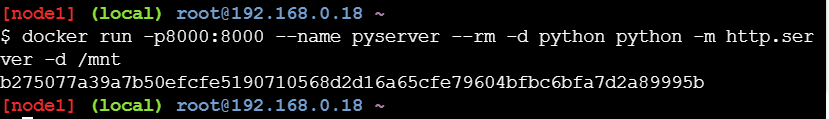
Ответ: -p8000:8000 – отвечает за проброс портов;

--name – создание имени контейнеру,

--rm – контейнер удаляется после завершения работы;

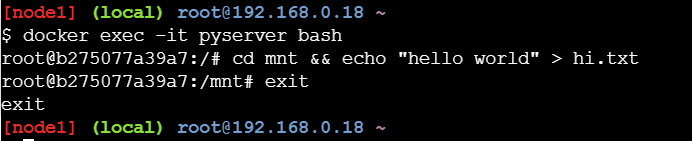
-m – ограничение памяти

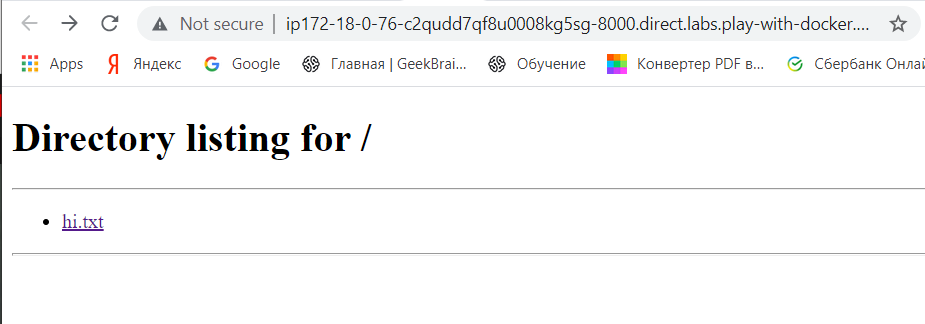
Команда run.

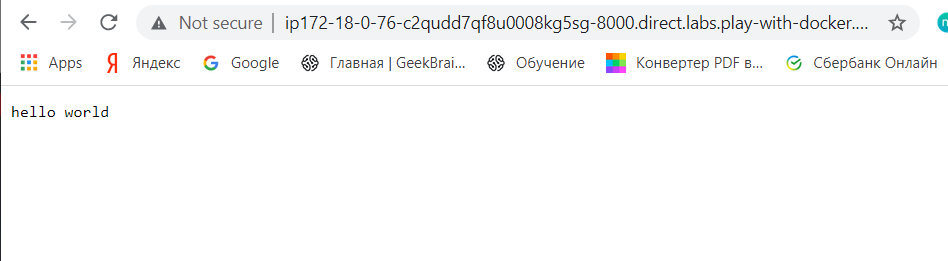


Для того, чтобы попасть в уже запущенный контейнер, существует команда docker exec –it pyserver bash – мы попадём в оболочку bash в контейнере. Попав в контейнер, выполнить команду cd mnt && echo “hello world” > hi.txt, а затем выйти из контейнера, введя команду exit или нажав комбинацию клавиш Ctrl+D.

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

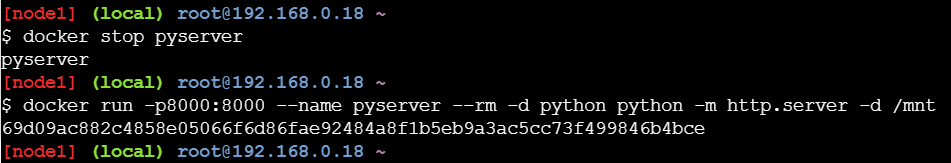




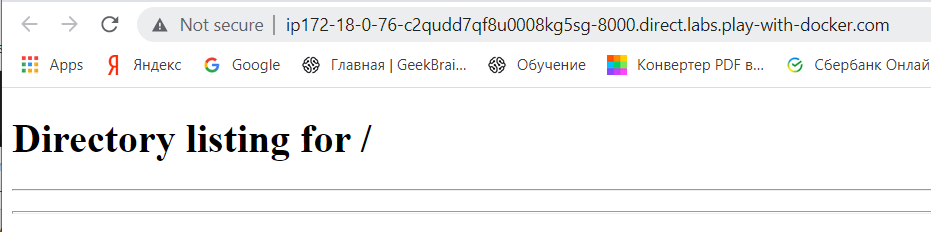


Остановим контейнер: docker stop pyserver, а затем снова зупстим:

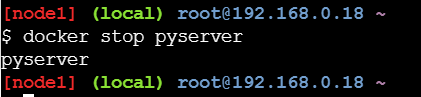
Docker run –p8000:8000 –name pyserver –rm –d python python –m http.server –d / mnt



Файл fi.txt пропал – мы запустили другой контейнер, а старый был удалён после завершения работы (флаг --rm).



Остановим контейнер: docker stop pyserver.

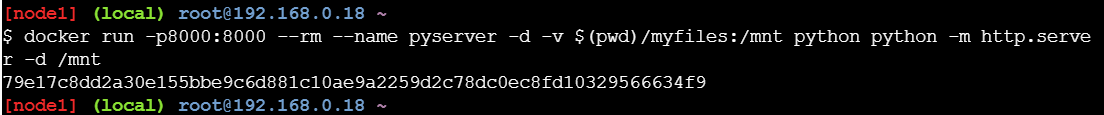


Для того, чтобы не терялись какие-то данные существует механизм монтирования.

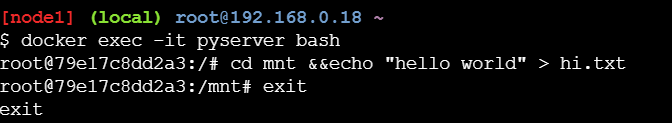
# **Часть 3.5.1 Тома**

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

Создаём контейнер с примонтированным томом.



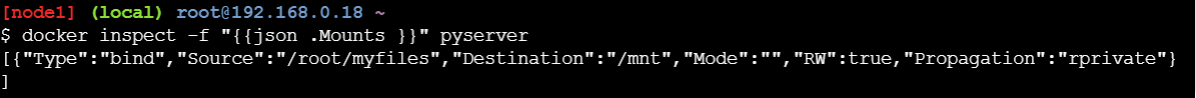
Если создать теперь файл (выполнить docker exec –it pyserver bash и внутри контейнера выполнить cd mnt && echo “hello world” > hi.txt), то даже после удаления контейнера данные в этом томе будут сохранены.



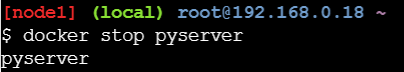
Чтобы узнать, где хранятся данные, выполню команду:

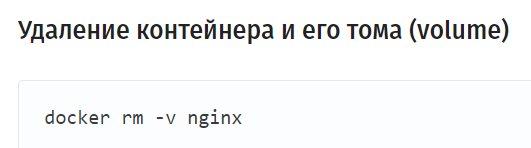
Docker inspect –f “{{json .Mounts}}” pyserver, в поле Source будет храниться путь до тома на хостовой машине.

**JSON** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *JavaScript Object Notation*, обычно произносится как [/ˈdʒeɪsən/](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D0%9C%D0%A4%D0%90_%D0%B4%D0%BB%D1%8F_%D0%B0%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0) JAY-sən) — [текстовый формат](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82) [обмена данными](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D0%B5%D0%BD_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8), основанный на [JavaScript](https://ru.wikipedia.org/wiki/JavaScript). Как и многие другие текстовые форматы, JSON легко читается людьми.



Остановим контейнер, созданный на предыдущем шаге: docker stop pyserver



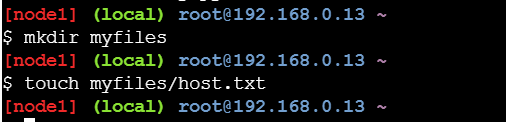


# **Часть 3.5.2 Монтирование директорий и файлов**

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

Создаю директорию mkdir myfiles

В директории создам файл host.txt: touch myfiles/host.txt, которые буду монтировать.

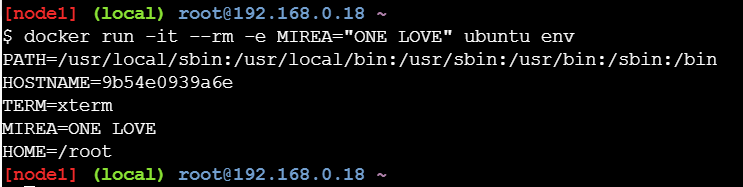


Запустим контейнер:   
Перенос строки осуществляется комбинацией “ \ ” + enter

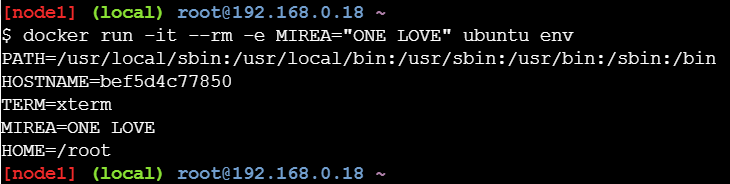


# **Часть 3.6 Переменные окружения**

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



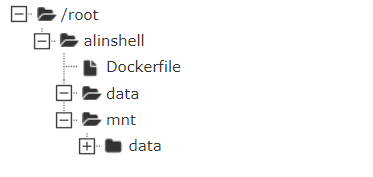
Проверка выводом всех переменных окружения, определённых в контейнере с помощью утилиты env:

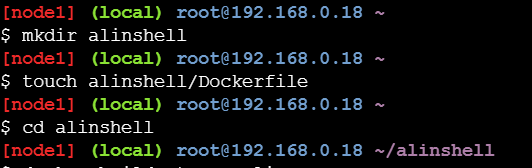
docker run –it --rm --e MIREA=”ONE LOVE” ubuntu env

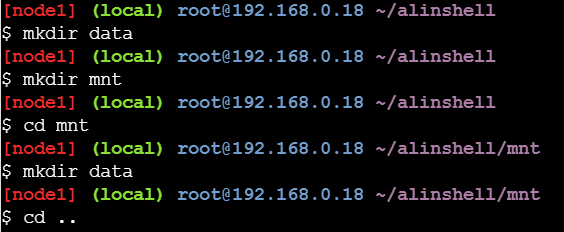
# **Часть 3.7 Dockerfile**

Собрать образ, в который будут установлены дополнительные пакеты, примонтировать директорию и установить команду запуска. Для это создаётся файл Dockerfile (без разрешения).

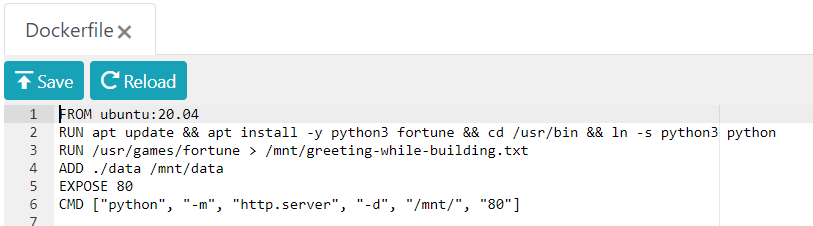
Создадим соответствующие директории и файлы:

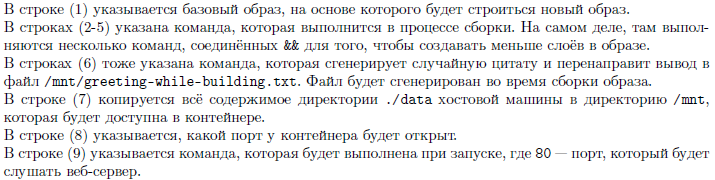




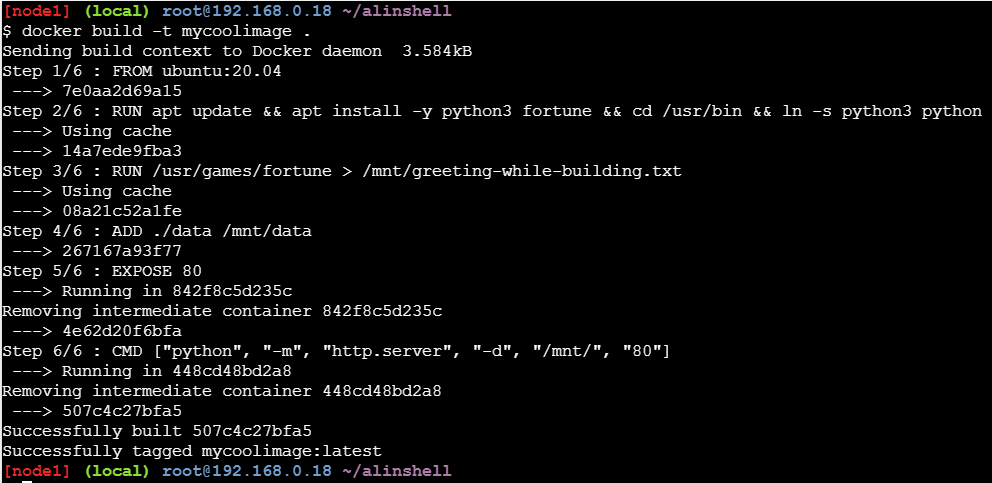


Создадим Dockerfile и запишем в него соответствующие значения:

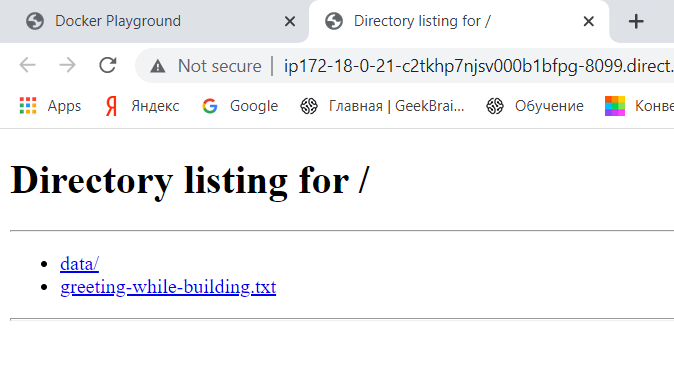


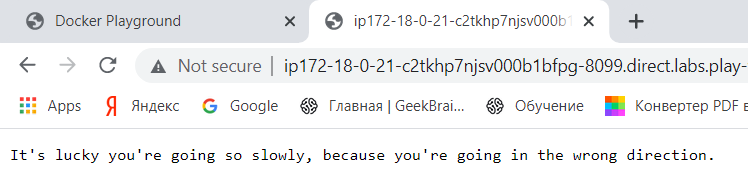


Создадим образ:



Запустим образ:

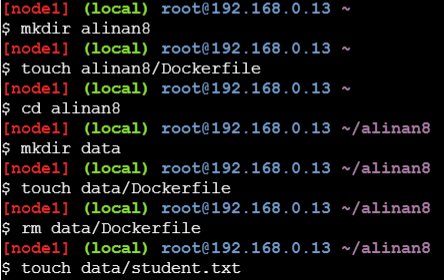


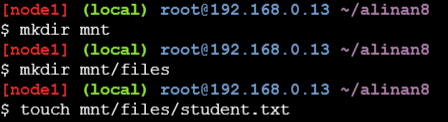


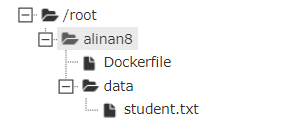


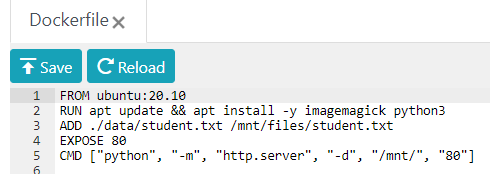
# **Часть 3.8 Индивидуальное задание**

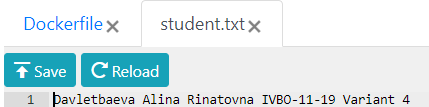
Создадим необходимые файлы и папки:



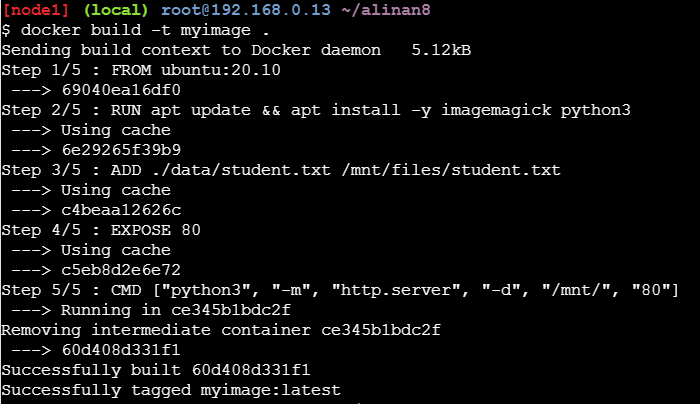




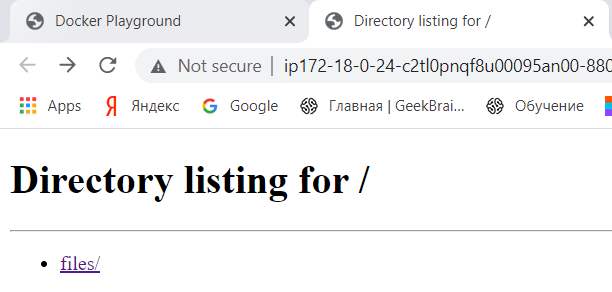


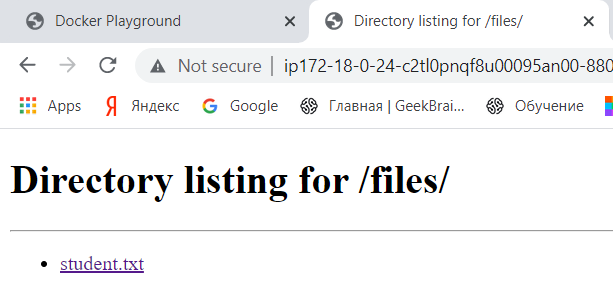


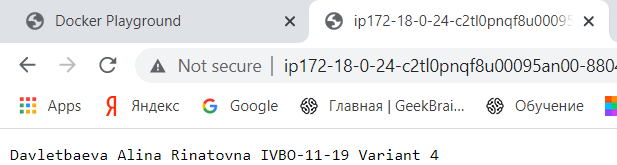
Создадим и запустим образ:

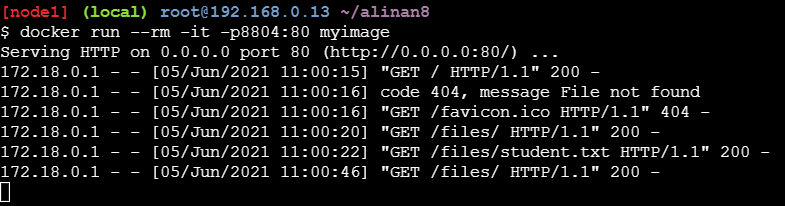


Запуск на порту хостовой машины - 8004









**Вывод:** в ходе выполнения работы мною были получены навыки работы с функционалом Docker, созданием собственного контейнера, загрузки и просмотра образов, монтирования директорий и файлов.