Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

*Факультет Программной инженерии и компьютерной техники*

Дисциплина «Технологии виртуализации»

**Лабораторная работа №4**

Знакомство с Docker

Группа: P3417

Выполнил: Ореховский А.

Проверил:

К.т.н. преподаватель Белозубов А.В.

Санкт-Петербург,

2020 г

Оглавление

[Задание 3](#_Toc62493752)

[Основные команды docker 5](#_Toc62493753)

[Ход работы 5](#_Toc62493754)

[Основной проект 5](#_Toc62493755)

[phpvirtyalbox 10](#_Toc62493756)

[NextCloud 10](#_Toc62493757)

[Выводы 12](#_Toc62493758)

# Задание

Подготовить отчет о ходе выполнения работы.

1. Проверить сетевой интерфейс на виртуальной машине. Установить – сетевой мост.
2. Установить Docker и VirtualBOX в ОС Ubuntu (возможно, Вам потребуется добавить пользователя в группу docker, чтобы вызвать эту команду без sudo)
3. Описать основные команды (ps, pull, build, logs, run, stop, kill, rm, rmi, cp, create, image, login, rename, servive)
4. Все действия выполнять в Терминале:

**$ docker --version**

**$ docker run Hello-Ваша\_Фамилия**

1. Создание Dockerfile
2. Создайте пустой каталог (MyLabs-Docker).
3. Войдите в каталог и создайте файл с именем Dockerfile
4. Скопируйте и вставьте в этот файл следующее содержимое и сохраните его.

FROM python:2.7-slim

# Укажем рабочий каталог /app

WORKDIR /app

# Скопируем текущее содержимое каталога в контейнер по адресу /app

ADD . /app

# Установить список пакетов из файла requirements.txt

RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Открыть 80 порт

EXPOSE 80

# Определение переменной окружения

ENV NAME World

# Команда запуска приложения

CMD ["python", "app.py"]

1. Создать файлы с содержимым:

**requirements.txt:**

Flask

Redis

**app.py:**

from flask import Flask

from redis import Redis, RedisError

import os

import socket

# Connect to Redis

redis = Redis(host="redis", db=0, socket\_connect\_timeout=2, socket\_timeout=2)

app = Flask(\_\_name\_\_)

@app.route("/")

def hello():

try:

visits = redis.incr("counter")

except RedisError:

visits = "<i>cannot connect to Redis, counter disabled</i>"

html = "<h3>Hello {name}!</h3>" \

"<b>Hostname:</b> {hostname}<br/>" \

"<b>Visits:</b> {visits}"

return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=80)

1. Описать что произойдет при первом запуске скрипта. Что происходит при повторных запусках?
2. Запустите скрипт и следующие команды команды

**$docker build -t friendlyhello**

**$ docker images**

1. Запустите приложение и откройте следующие порты (последние 4 цифры вашего ID в ИСУ) **4000** и 80

**$docker run -p 80:80 friendlyhello**

**$docker run -p 4000:80 friendlyhello**

Проверьте работу в браузере <http://localhost:4000> и http://localhost:80

1. Выполните команду

**$docker container ls**

1. Запустите программу app в фоновом режиме

**$docker run -d -p 4000:80 friendlyhello**

1. Выполните команду

**$docker container ls**

1. Завершите работу контейнера friendlyhello по ID и имени
2. Создать контейнер с приложением phpvirtualbox
3. Создать контейнер с приложением NextCloud

# Основные команды docker

* ps – список контейнеров.
* pull – скачивание образа или репозитория из реестра docker.
* build – создание образа из файла dockerfile.
* logs – список логов контейнера.
* run – запуск процесса в контейнере.
* stop – остановка одного или нескольких контейнеров.
* kill – удаление одного или нескольких контейнеров запущенных.
* rm – удаление одного или нескольких контейнеров.
* rmi - удаление одного или нескольких образов.
* cp – копирование файла/каталога между контейнером и локальной файловой системой.
* create – создание нового контейнера.
* image – управление образами.
* login – вход в реестр docker.
* rename – переименование контейнера.
* service – управление сервисами.

# Ход работы

## Основной проект

Проверил интерфейс и установив docker и VirtualBox. Затем проверил что docker установился правильно, проверив его версию. Далее язапустил команду

# docker run Hello\_Orekhovskiy

Разумеется, ничего это команда не выполнила, так как ни локально, ни в репозитории docker не имеется образ с таким именем.

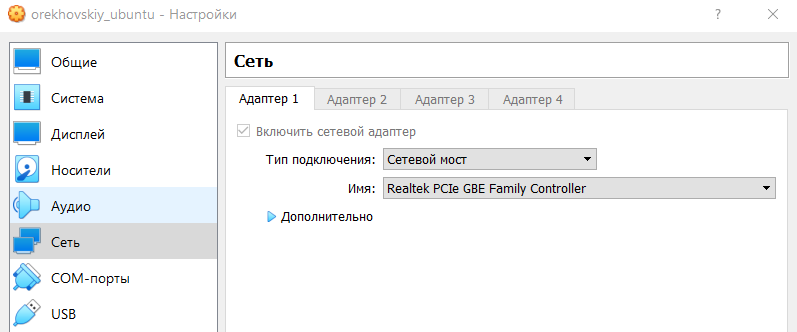


Рисунок 1. Тип подключения.

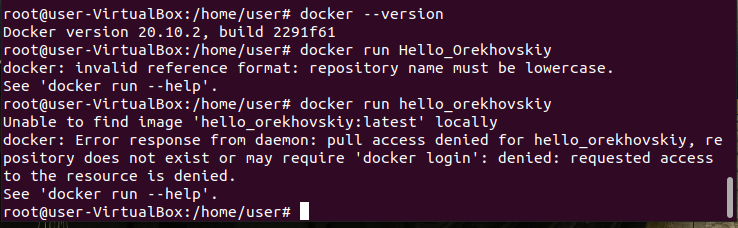


Рисунок 2. Проверка работы docker.

Далее я создал пустой каталог MyLabs-Docker, создал в нем Dockerfile, requirements.txt, а так же app.py.

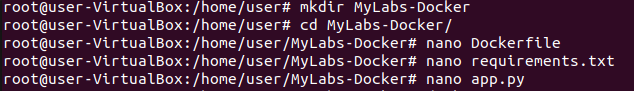


Рисунок 3.а. Процесс подготовки docker проекта.

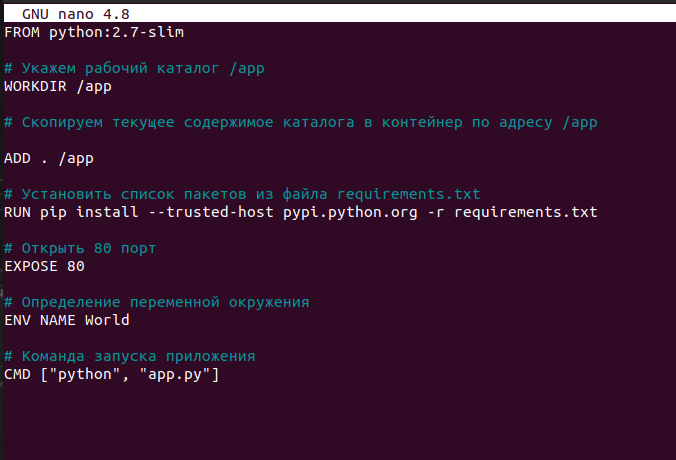


Рисунок 3.б. Содержимое файла Dockerfile

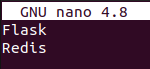


Рисунок 3.в. Содержимое файла requirements.txt



Рисунок 3.г. Содержание файла app.py.

Файл dockerfile помогает создать необходимы нам образ. В файле requirements.txt указаны необходимые модули для корректной работы файла app.py. В файле app.py сначала мы подключаемся к Redis, затем мы описываем endpoint. По идее, если подключение к Redis было произведено успешно, то мы должны вывесть страницу приветствия, на которой берется имя хоста из переменной среды, а так же присутствует счетчик из Redis, то есть при каждом новом обращении к данному endpoint-у счетчик длжен увеличиваться.

Проверить это можно запустив образ. Собственно, я создал этот образ и проверил его наличие, а затем запустил данный образ на порту 80. Так ка была установлена библиотека для связи с Redis, но не сам он, то была ошибка подключения к нему. Так же было и на порту 5123.

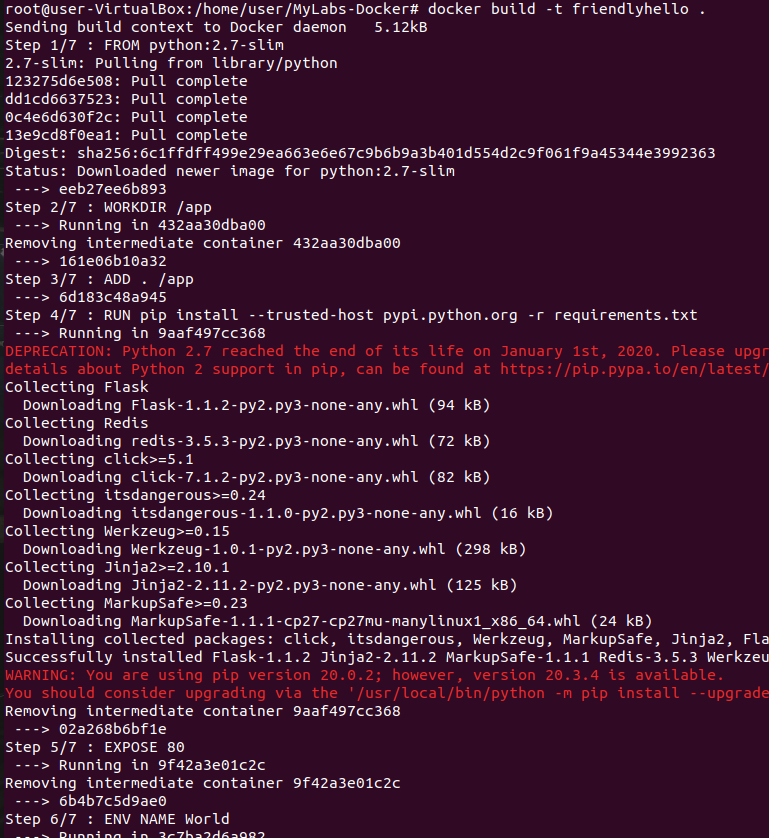


Рисунок 4.а. Процесс создания образа.

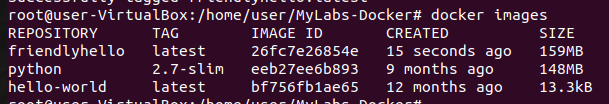


Рисунок 4.б. Список доступных образов.

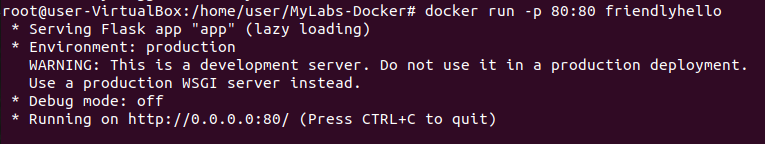


Рисунок 4.в. Запуск образа на порту 80.

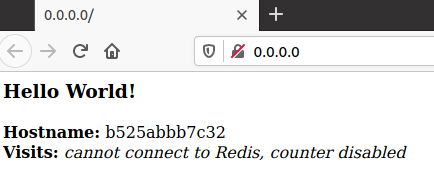


Рисунок 4.г. Попытка обращения к endpoint-у

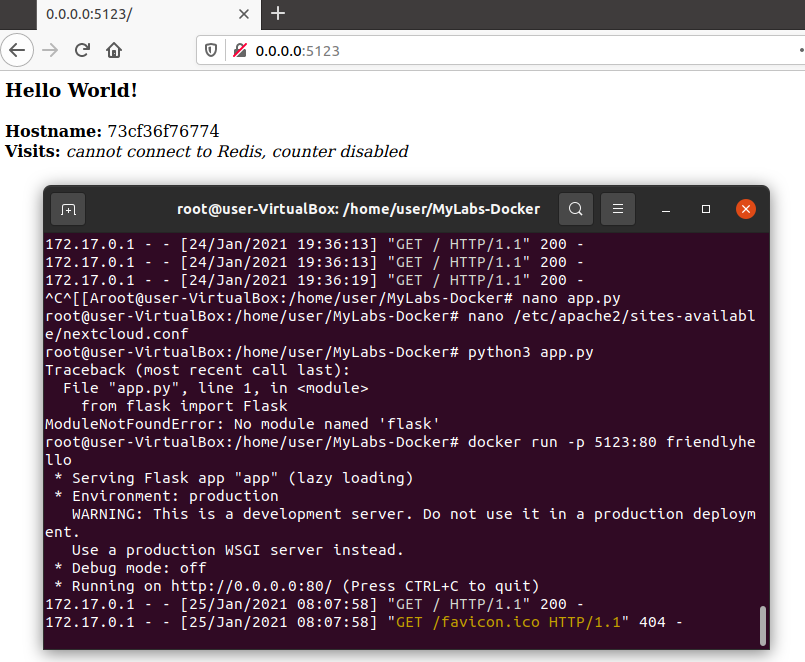


Рисунок 4.д. Запуск на другом порте и попытка подключения к нему.

Далее, согласно заданию, я выполнил команду просмотра всех контейнеров docker-а, предварительно остановив выполнения образа. далее я повторил эту команду, но уже с запущенным в фоне образом.



Рисунок 5.а. Проверка всех контейнеров.

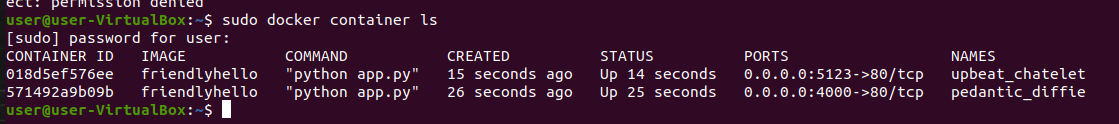


Рисунок 5.б. Проверка всех контейнеров, с запущенным в фоне образом.

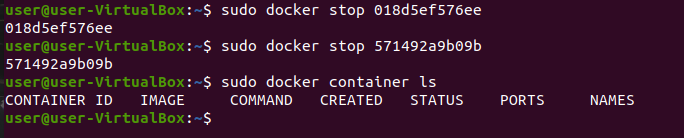


Рисунок 5.в. Остановка запущенных образов по их Ид.

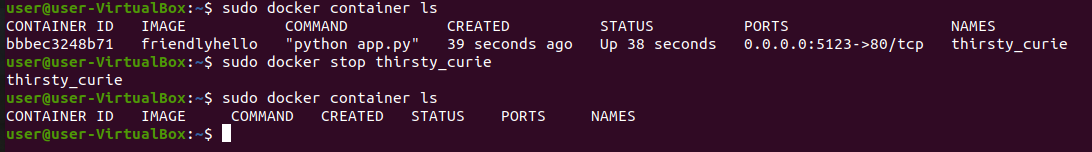


Рисунок 5.г. Остановка запущенного образа по его имени.

## phpvirtyalbox

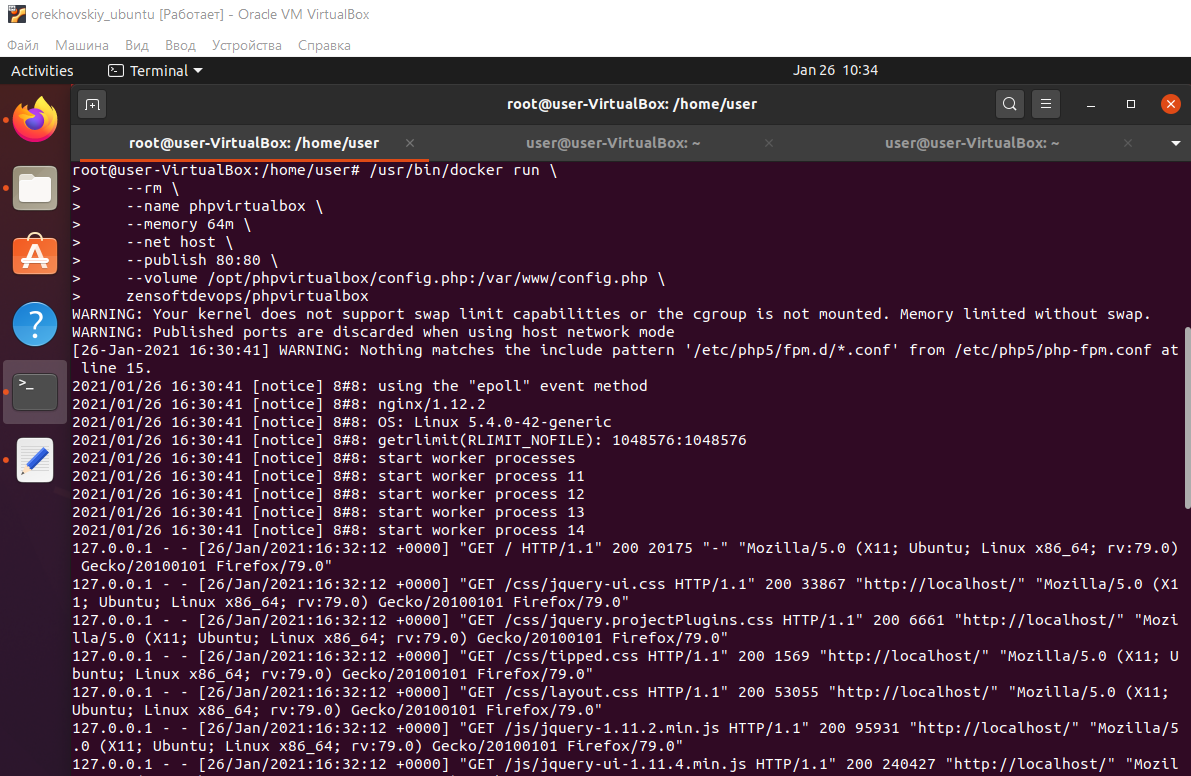


Рисунок 6.a. Запуск контейнера phpvirtualbox.

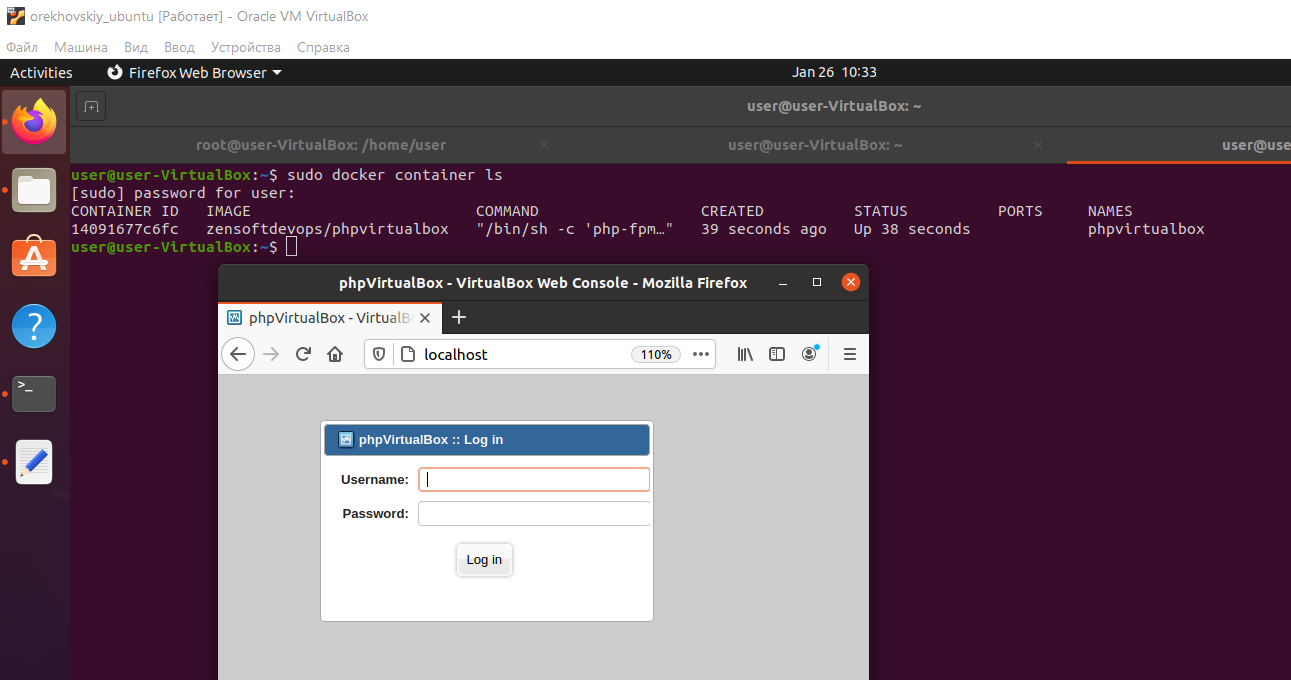


Рисунок 6.б. Запущенный phpvirtualbox

## NextCloud

Для начала, я создал файл docker-compose.yml, а затем запустил его.

version: ‘3’

networks:

net:

ipam:

config:

- subnet: 192.168.101.1/24

volumes:

db:

app:

services:

db:

image: mysql/mysql-server

networks:

- net

volumes:

- db:/var/lib/mysql

restart: always

app:

depends\_on:

- db

image: nextcloud

networks:

- net

ports:

- 80:80

volumes:

- app:/var/www/html

restart: always

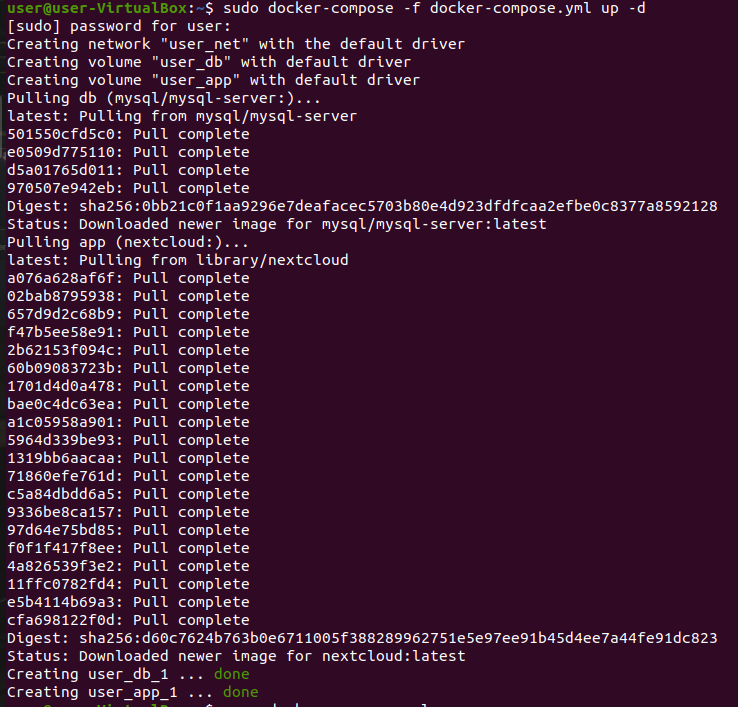


Рисунок 7.a. Запуск NextCloud.

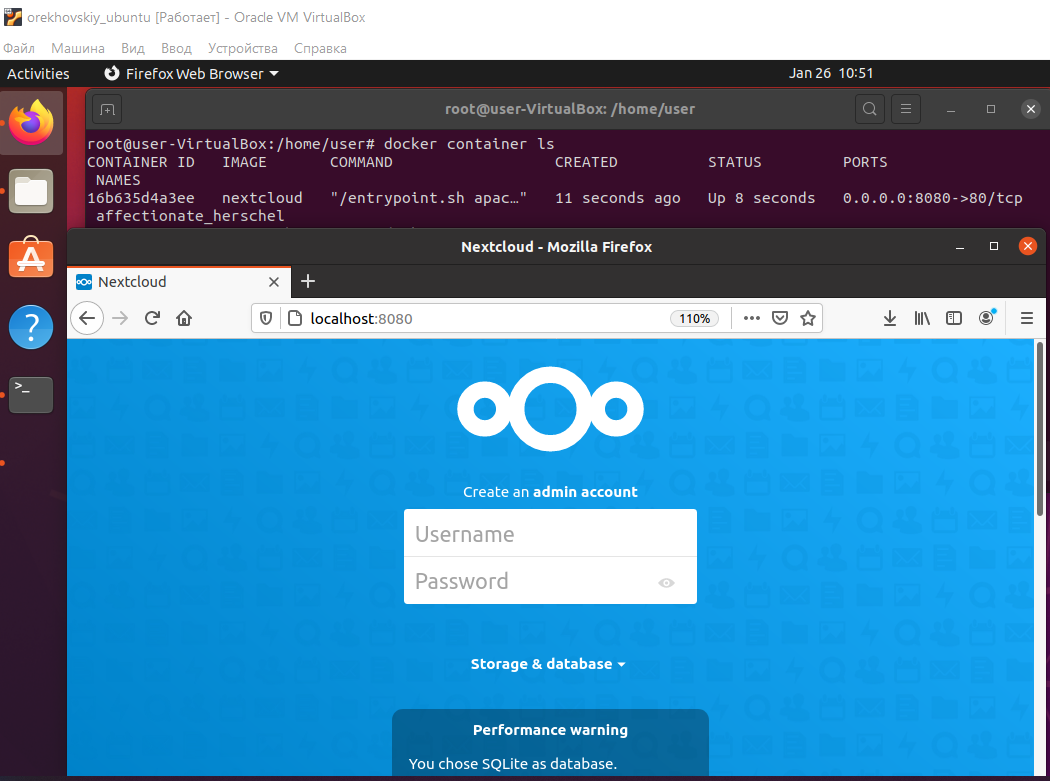


Рисунок 7.б. Запущенный NextCloud.

# Выводы

Сам по себе docker является хорошим средством, которое упростит жизнь в плане развертывания приложений. Как и полагает программам контейнеризации докер помогает быстро запускать готовые продукты, автоматизировать их, гарантирует запуск с одинаковыми условиями на любой среде. После выполнения данной работы, у меня появились начальные умения работы с docker. Что самое интересное, наибольшее затруднение вызвала у меня работа с файлом python, так как это первый мой опыт работы с данным языком. Кто бы знал, что табуляция может быть такой коварной! В будущем буду иметь в виду, что после изменения файла необходима пересобрать образ, чтобы изменения вступили в силу, хотя это и очевидно.