Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина «Технологии виртуализации»

Отчёт

по лабораторной работе №3.1

Выполнил:
Барсуков М.А., группа P3415
Проверил:
преподаватель Адмакин М.А.

Санкт-Петербург

2025 г.

Содержание

Введение

В рамках данной лабораторной работы изучается технология контейнеризации приложений с использованием платформы Docker — одного из ключевых инструментов современной разработки, DevOps-практик и облачной инфраструктуры. Docker позволяет изолировать приложения и их зависимости в легковесных, переносимых контейнерах, обеспечивая воспроизводимость среды выполнения независимо от хостовой операционной системы. В отличие от традиционной виртуализации, контейнеризация не требует эмуляции аппаратного обеспечения и обеспечивает более высокую производительность и гибкость при развертывании программных решений.

Для выполнения работы использовалась операционная система Arch Linux, на которой были установлены Docker и сопутствующие инструменты. В ходе лабораторной работы были освоены основные команды Docker CLI, рассмотрены принципы создания и сборки собственных образов с помощью Dockerfile, а также изучены механизмы управления многоконтейнерными приложениями через Docker Compose. Особое внимание уделено практическому развертыванию популярных сервисов: веб-сервера (Nginx), СУБД (MariaDB), облачного хранилища (NextCloud) и веб-интерфейса управления виртуализацией (phpVirtualBox).

Также была выполнена регистрация на платформе Docker Hub — централизованном репозитории образов контейнеров, — и опубликован собственный образ, что демонстрирует возможность интеграции в цикл непрерывной доставки (CI/CD) и совместной разработки. Дополнительно проведена демонстрация запуска готовых приложений из публичного реестра, включая интерактивные веб-игры, что подчеркивает простоту и универсальность экосистемы Docker.

Цель работы

Изучить основы работы с Docker, основные команды, создание и запуск контейнеров, запуск и управление нескольких контейнеров, Регистрация на Docker Hub. Подготовить отчет о ходе выполнения работы.

Задачи работы

1. Установить Docker и настроить права доступа.
2. Проверить работоспособность Docker, выполнить команды CLI.
3. Создать Dockerfile для веб-сервера с пользовательской HTML-страницей и запустить контейнер на порту 8080.
4. Развернуть контейнер с СУБД MariaDB, подключиться к нему и создать базу данных с именем по фамилии студента.
5. Запустить контейнер с NextCloud, настроить хранение данных на хосте и зарегистрировать пользователей.
6. Развернуть контейнер с phpVirtualBox.
7. Создать и запустить многоконтейнерное приложение с помощью Docker Compose, объединяющее NextCloud, веб-сервер и базу данных, с доступом через порт 2022.
8. Зарегистрироваться на Docker Hub, опубликовать собственный образ.
9. Продемонстрировать запуск сторонних приложений из Docker Hub (например, игры 2048).
10. Систематизировать и описать основные команды Docker для управления образами и контейнерами.

Выполнение

1. Установка Docker и VirtualBox

Установлен пакет docker. Пользователь добавлен в группу docker для выполнения команд без sudo. Демон Docker запущен и включён в автозагрузку, VirtualBox также установлен, что показано на рисунке 1:

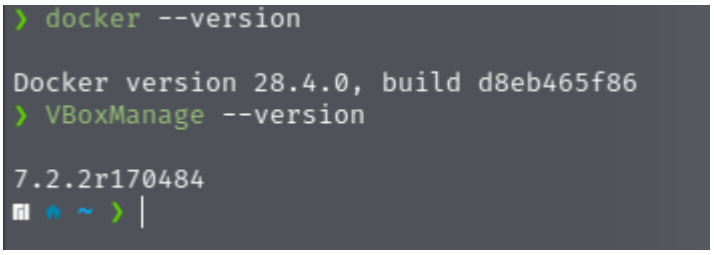

Рисунок 1 – Docker и VirtualBox установлены

2. Запуск тестового контейнера

Попробуем запустить тестовый контейнер *hello-barsukov*, как показано на рисунке 2:

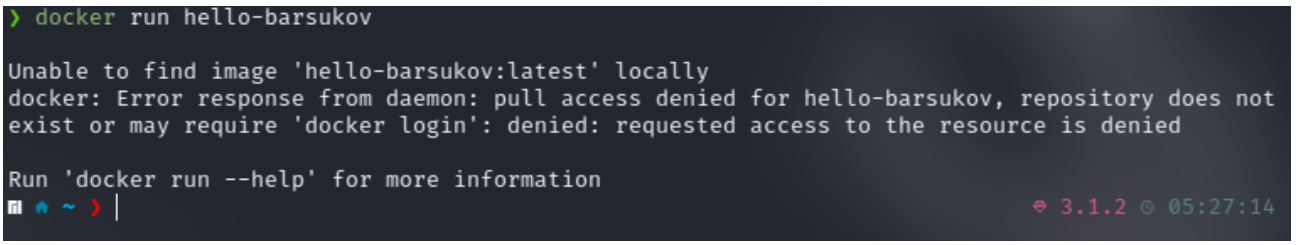

Рисунок 2 – Невозможно найти *hello-barsukov*

Мы получаем ошибку, так как образа *hello-barsukov* не существует как локально, так и в удаленном репозитории образов. Исправим это, создав образ для *hello-barsukov*, как показано на рисунке 3:

```
FROM alpine:latest
CMD ["echo", "Hello Barsukov"]
```

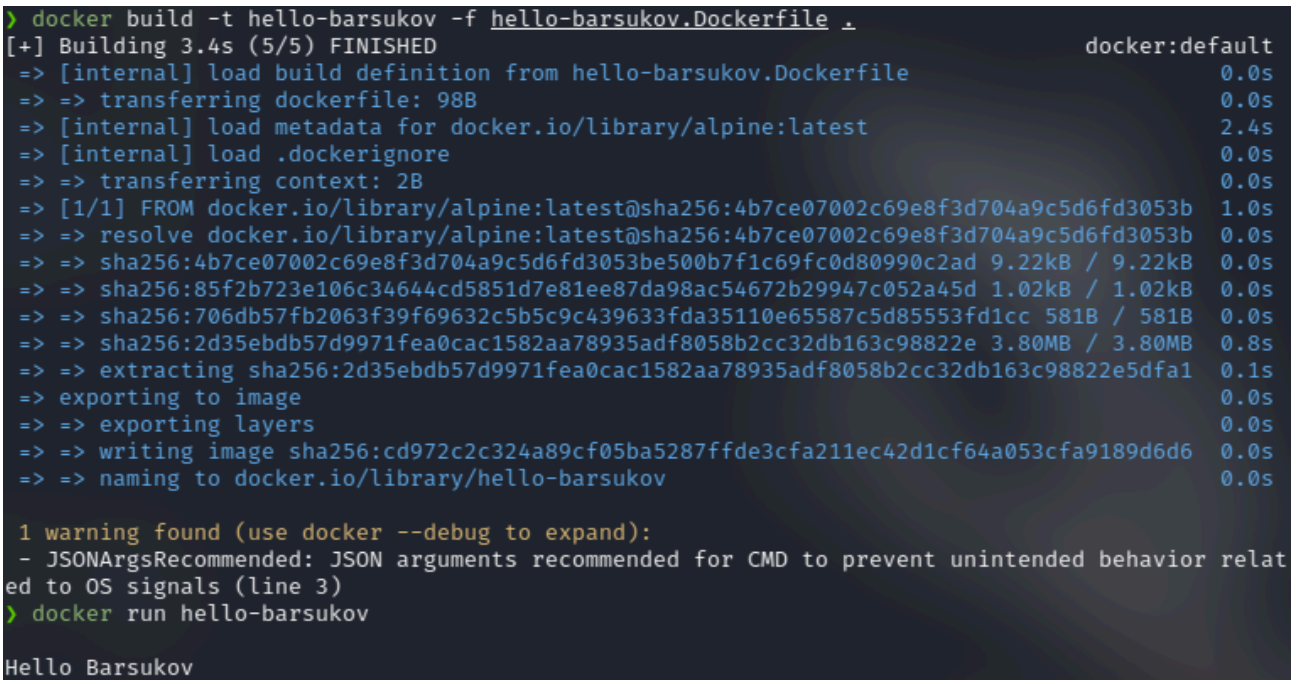

Рисунок 3 – Проверка работоспособности Docker и запуск первого контейнера

3. Контейнер с Nginx и пользовательской страницей

Создан каталог */home/www/html*, в нём размещён HTML-файл *index.html*. Написан *nginx.Dockerfile* на базе образа *nginx:latest*, который пробрасывает порт 8080 и копирует страницу, как показано на рисунке 4:

```
❯ cat /home/www/html/index.html
<!DOCTYPE html>
<html>
<head>
    <title>Технологии виртуализации ЛР3</title>
</head>
<body>
    <h1>Барсуков Максим Андреевич</h1>
    <p>Студент группы P3415</p>
</body>
</html>
❯ cat nginx.Dockerfile
FROM nginx:latest
LABEL maintainer="Барсуков Максим Андреевич"
WORKDIR /usr/share/nginx/html
COPY index.html .
EXPOSE 8080
CMD ["nginx", "-g", "daemon off;"]
❯
```

Рисунок 4 – Содержимое Dockerfile и HTML-страницы

Образ собран и запущен с монтированием тома, что видно на рисунке 5:

```
❯ docker build -t my-nginx -f nginx.Dockerfile .
[+] Building 9.2s (8/8) FINISHED                                                docker:default
 => [internal] load build definition from nginx.Dockerfile                                0.0s
 => => transferring dockerfile: 226B                                                      0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                           1.9s
 => [internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                           0.0s
 => [1/3] FROM docker.io/library/nginx:latest@sha256:f547e3d0d5d02f7009737b284abc87d808e  6.7s
 => => resolve docker.io/library/nginx:latest@sha256:f547e3d0d5d02f7009737b284abc87d808e  0.0s
 => => sha256:9d0e6f6199dcb6e045dad103064601d730fcfaf8d1bd357d969fb70bd5 8.75kB / 8.75kB  0.0s
 => => sha256:0e86847a3920c130e5f7e66bdf15b66b9ed5426b80ecf1b2f8e7960794620a 629B / 629B  2.4s
 => => sha256:f547e3d0d5d02f7009737b284abc87d808e4252b42dceea361811e9f 10.23kB / 10.23kB  0.0s
 => => sha256:12549785f32b3daca6f1c39e7d756226eeb0e8bb20b9e2d8a03d484160 2.29kB / 2.29kB  0.0s
 => => sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d19 29.78MB / 29.78MB  4.6s
 => => sha256:a0a6ab1415585a8dc626877b8de231086ab4d45ea8709d35b8ac7ec2 29.97MB / 29.97MB  5.6s
 => => sha256:1bace208328954ff24f16c62a7492c4b3a4d9451a10f23631417f3feebe3f4 955B / 955B  2.9s
 => => sha256:89df300a082a0f5d71c8aff1944dc57a49544592d8d804fe7c61b7e844aa80 404B / 404B  3.6s
 => => sha256:35fb9ffa6621d9044c7062eb2fe4b181ce919447e2575a8577dcedbd7a 1.21kB / 1.21kB  4.2s
 => => sha256:5545b08f9d26518aae35920f1c1b7cdf962b32b69e9feba182bfb3e0b9 1.40kB / 1.40kB  4.6s
 => => extracting sha256:38513bd7256313495cdd83b3b0915a633cfa475dc2a07072ab2c8d191020ca5  0.9s
 => => extracting sha256:a0a6ab1415585a8dc626877b8de231086ab4d45ea8709d35b8ac7ec24bea303  0.7s
 => => extracting sha256:0e86847a3920c130e5f7e66bdf15b66b9ed5426b80ecf1b2f8e7960794620a6  0.0s
 => => extracting sha256:1bace208328954ff24f16c62a7492c4b3a4d9451a10f23631417f3feebe3f44  0.0s
 => => extracting sha256:89df300a082a0f5d71c8aff1944dc57a49544592d8d804fe7c61b7e844aa80b  0.0s
 => => extracting sha256:35fb9ffa6621d9044c7062eb2fe4b181ce919447e2575a8577dcedbd7aa5aa6  0.0s
 => => extracting sha256:5545b08f9d26518aae35920f1c1b7cdf962b32b69e9feba182bfb3e0b90dcc2  0.0s
 => [internal] load build context                                                         0.0s
 => => transferring context: 280B                                                         0.0s
 => [2/3] WORKDIR /usr/share/nginx/html                                                   0.0s
 => [3/3] COPY index.html .                                                               0.0s
 => exporting to image                                                                    0.5s
 => => exporting layers                                                                   0.5s
 => => writing image sha256:66ac364fcbe395eafe1771ad994271d1f914d4186fa7651f54d59a2e3b99  0.0s
 => => naming to docker.io/library/my-nginx                                               0.0s
❯
❯ docker run -d -p 8080:80 -v /home/www/html:/usr/share/nginx/html:ro --name my-webserver my-nginx
5895f660345e96b2dc7628524b0e985f697e8fdc245d39f00e447c1352302c78
```

Рисунок 5 – Сборка и запуск образа

Вывод браузера по адресу http://localhost:8080 с заголовком «Барсуков Максим Андреевич» виден на рисунке 6:

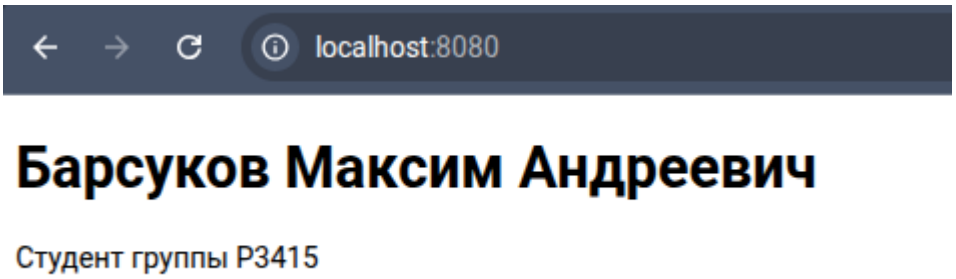

Рисунок 6 – Результат работы веб-сервера в браузере

4. Контейнер с MariaDB

Запущен контейнер *mariadb* с монтированием тома */home/DB* и пробросом порта 3306, как показано на рисунке 7:

```
❯ docker run -d --name my-mariadb \
  -v /home/DB:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=admin \
  -p 3306:3306 \
  mariadb:12.1.1-rc

Unable to find image 'mariadb:12.1.1-rc' locally
12.1.1-rc: Pulling from library/mariadb
4b3ffd8ccb52: Pull complete
97d12f886595: Pull complete
145d03c0878f: Pull complete
13dd8409c1da: Pull complete
407392161996: Pull complete
28621b2bf47e: Pull complete
2ca321759995: Pull complete
1e67950216d6: Pull complete
Digest: sha256:6c5b9de5ef15cd5e512ea4bfa8ec7bee5db0d85571c4a4d647793455a390b117
Status: Downloaded newer image for mariadb:12.1.1-rc
dbb6ff849fb6b412a43de11f331c2a1830fcef5a1e8b762ed8edf500f31fe674
```

Рисунок 7 – Запуск *mariadb*

Через *docker exec* выполнено подключение к СУБД и создана база данных с именем *barsukov*, как показано на рисунке 8:

```
❯ docker exec -it mariadb-container mariadb -u root -padmin -e
 "CREATE DATABASE barsukov; SHOW DATABASES;"
+--------------------+
| Database           |
+--------------------+
| barsukov           |
| information_schema |
| mariadb            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
❯
```

Рисунок 8 – Создание базы данных в MariaDB через терминал контейнера

В итоге база данных сохранилась как отдельная папка внутри хостовой директории */home/DB*, которая подключена (смонтирована) внутрь контейнера, как показано на рисунке 9:

```
❯ ls -la /home/DB
итого 158476
drwxr-xr-x 7  999 adm       4096 ноя  3 06:26 .
drwxr-xr-x 5 root root      4096 ноя  3 06:24 ..
-rw-rw---- 1  999 adm    4907008 ноя  3 06:26 aria_log.00000001
-rw-rw---- 1  999 adm         52 ноя  3 06:26 aria_log_control
drwx------ 2  999 adm       4096 ноя  3 06:26 barsukov
-rw-rw---- 1  999 adm          9 ноя  3 06:26 ddl_recovery.log
-rw-rw---- 1  999 adm        722 ноя  3 06:26 ib_buffer_pool
-rw-rw---- 1  999 adm   12582912 ноя  3 06:25 ibdata1
-rw-rw---- 1  999 adm  100663296 ноя  3 06:25 ib_logfile0
-rw-rw---- 1  999 adm   12582912 ноя  3 06:26 ibtmp1
drwx------ 2  999 adm       4096 ноя  3 06:25 mariadb
-rw-r--r-- 1  999 adm         14 ноя  3 06:25 mariadb_upgrade_info
-rw-rw---- 1  999 adm          0 ноя  3 06:25 multi-master.info
-rw------- 1  999 adm        118 ноя  3 06:25 .my-healthcheck.cnf
drwx------ 2  999 adm       4096 ноя  3 06:25 mysql
drwx------ 2  999 adm       4096 ноя  3 06:25 performance_schema
drwx------ 2  999 adm      12288 ноя  3 06:25 sys
-rw-rw---- 1  999 adm      24576 ноя  3 06:26 tc.log
-rw-rw---- 1  999 adm   10485760 ноя  3 06:25 undo001
-rw-rw---- 1  999 adm   10485760 ноя  3 06:25 undo002
-rw-rw---- 1  999 adm   10485760 ноя  3 06:25 undo003
```

Рисунок 9 – Содержимое */home/DB*

5. Контейнер с NextCloud

Запущен официальный образ *nextcloud* с пробросом порта 8088 и монтированием каталога */home/www/NextCloud*, как показано на рисунке 10:

```
❯ docker run -d \
  --name my-nextcloud \
  -v /home/www/NextCloud:/var/www/html \
  -p 8088:80 \
  nextcloud:latest
22781828a96a7842ff92e043aa0b0e461bb0ba14ab85a31a078df62a2aeceb05
❯
```

Рисунок 10 – Создание контейнера с NextCloud

Логин и пароль от NextCloud – admin:admin, как показано на рисунке 11:

Рисунок 11 – Настройка NextCloud

Через веб-интерфейс зарегистрированы два пользователя: студент (Барсуков М.А.) и преподаватель (Адмакин М.А.), как показано на рисунках 12 и 13:

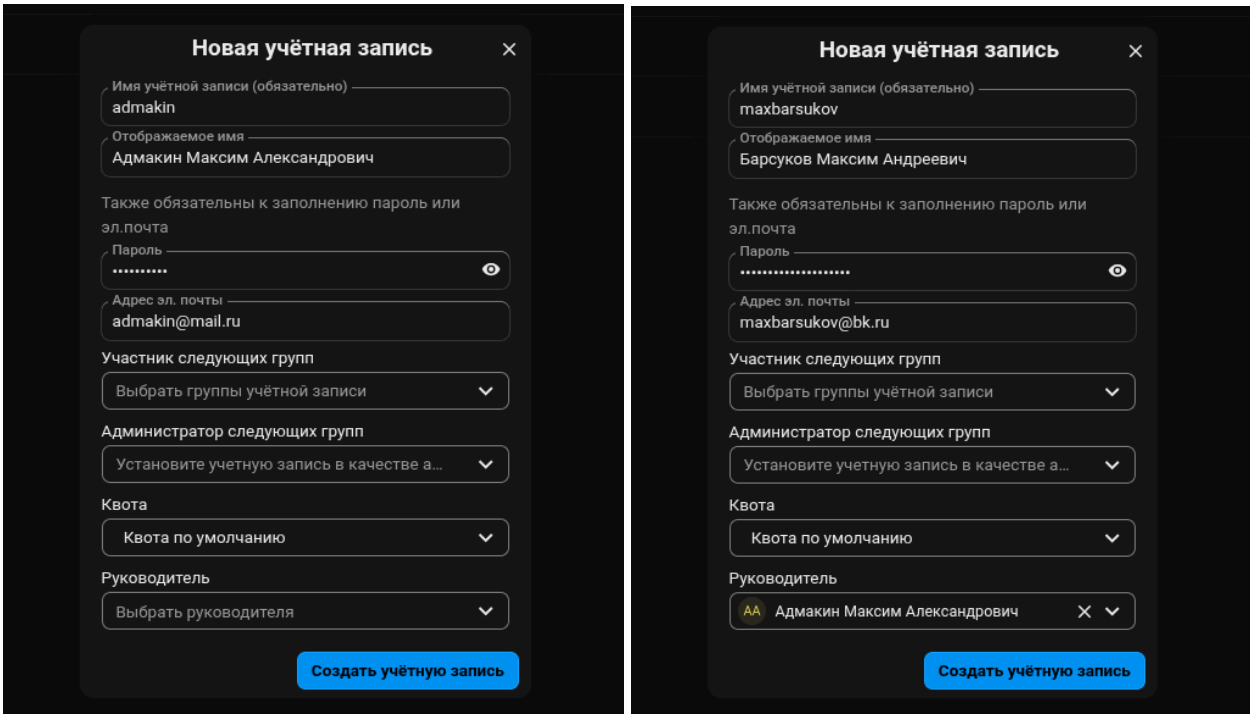

Рисунок 12 – Создание пользователей

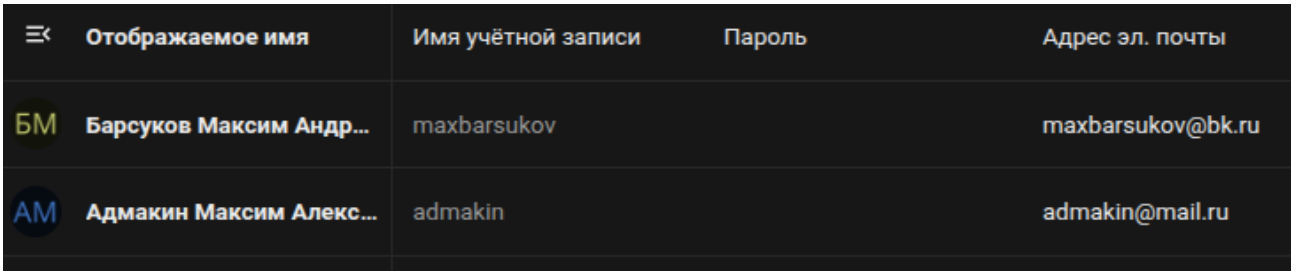

Рисунок 13 – Раздел «Пользователи» в NextCloud

6. Контейнер с phpvirtualbox

Сначала запустим *vboxwebsrv*, как показано на рисунке 14:

```
❯ vboxwebsrv -H 0.0.0.0 -A null -b
Oracle VirtualBox web service Version 7.2.2
Copyright (C) 2007-2025 Oracle and/or its affiliates
00:00:00.000799 main     VirtualBox web service 7.2.2 r170484 linux.amd64 (Sep 11 2025 11:31:33) release log
00:00:00.000801 main     Log opened 2025-11-03T03:46:19.726787000Z
00:00:00.000801 main     Build Type: release
00:00:00.000802 main     OS Product: Linux
00:00:00.000803 main     OS Release: 6.16.8-1-MANJARO
00:00:00.000803 main     OS Version: #1 SMP PREEMPT_DYNAMIC Fri, 19 Sep 2025 16:09:36 +0000
00:00:00.000822 main     DMI Product Name: ZenBook UX425EA_UX425EA
00:00:00.000826 main     DMI Product Version: 1.0
00:00:00.000829 main     Firmware type: UEFI
00:00:00.000995 main     Secure Boot: Disabled
00:00:00.001020 main     Host RAM: 15683MB (15.3GB) total, 6334MB (6.1GB) available
00:00:00.001022 main     Executable: /usr/lib/virtualbox/vboxwebsrv
00:00:00.001023 main     Process ID: 51379
00:00:00.001023 main     Package type: LINUX_64BITS_GENERIC (OSE)
❯
```

Рисунок 14 – Запуск веб-сервиса VirtualBox

Логин:пароль от *phpVirtualBox* – admin:admin, как показано на рисунке 15:

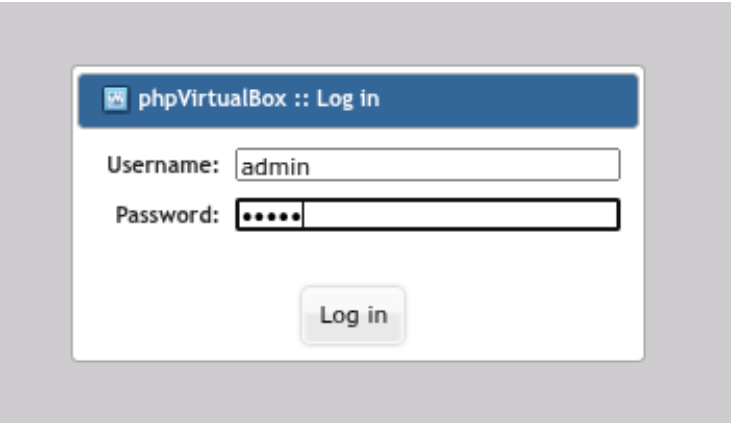

Рисунок 15 – Начальный экран phpVirtualBox

Можно увидеть те же виртуальные машины, что и в VirtualBox, и запустить их, как показано на рисунках 16 и 17:

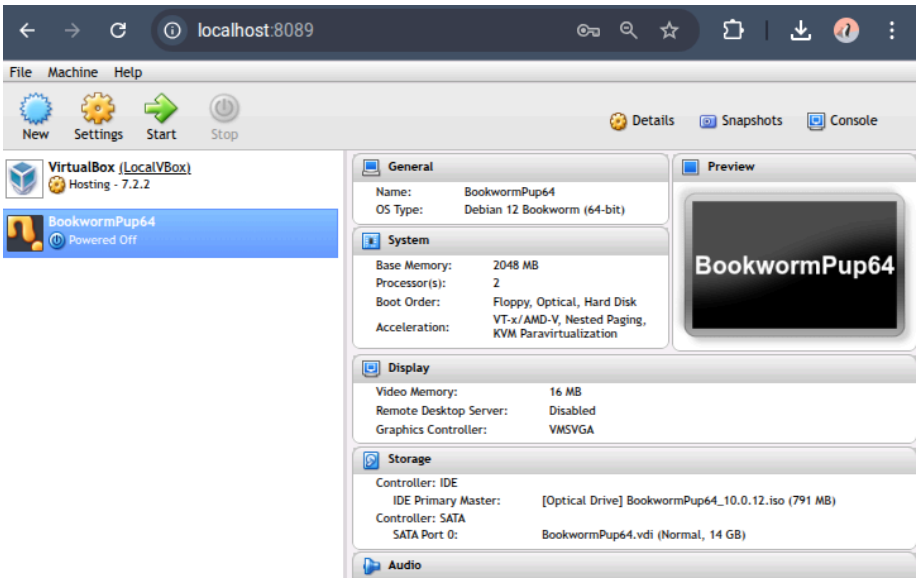

Рисунок 16 – Список VM в phpVirtualBox

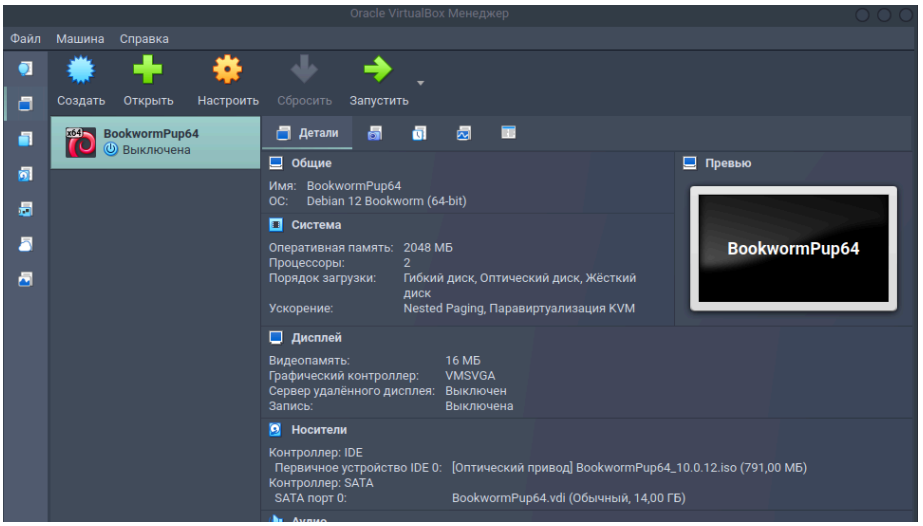

Рисунок 17 – Список VM в VirtualBox

7. Настройка окружения с помощью Docker Compose

Создан файл *docker-compose.yml*, описывающий три сервиса: *db* (MariaDB), *app* (NextCloud), *web* (Nginx), как показано на рисунке 18:

```
❯ show-files --color docker-compose.yml
============= docker-compose.yml =============
services:
  web:
    build:
      context: .
      dockerfile: nginx.Dockerfile
    container_name: nginx-webserver
    restart: unless-stopped
    ports:
      - "8080:80"
    volumes:
      - /home/www/html:/usr/share/nginx/html:ro
    networks:
      - nextcloud-network

  db:
    image: mariadb:12.1.1-rc
    container_name: nextcloud-mariadb
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: admin
      MYSQL_DATABASE: mariadb
    volumes:
      - /home/DB:/var/lib/mysql
    networks:
      - nextcloud-network

  nextcloud:
    image: nextcloud:latest
    container_name: nextcloud-app
    restart: unless-stopped
    ports:
      - "2022:80"
    environment:
      MYSQL_HOST: db
      MYSQL_DATABASE: mariadb
      MYSQL_USER: root
      MYSQL_PASSWORD: admin
    volumes:
      - /home/www/NextCloud:/var/www/html
    depends_on:
      - db
      - web
    networks:
      - nextcloud-network

networks:
  nextcloud-network:
    driver: bridge
```

Рисунок 18 – Содержимое docker-compose.yml

Вывод команды docker-compose ps после запуска показан на рисунке 19:

```
❯ docker-compose ps
NAME                IMAGE               COMMAND                  SERVICE     CREATED          STATUS          PORTS
nextcloud-app       nextcloud:latest    "/entrypoint.sh apac…"   nextcloud   3 minutes ago    Up 2 minutes    0.0.0.0:2022->80/tcp, [::]:2022->80/tcp
nextcloud-mariadb   mariadb:12.1.1-rc   "docker-entrypoint.s…"   db          3 minutes ago    Up 2 minutes    3306/tcp
nginx-webserver     31-web              "/docker-entrypoint.…"   web         3 minutes ago    Up 2 minutes    8080/tcp, 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
❯
```

Рисунок 19 – Статус запущенных сервисов через Docker Compose

Страницу NextCloud по адресу http://localhost:2022 можно увидеть на рисунке 20:

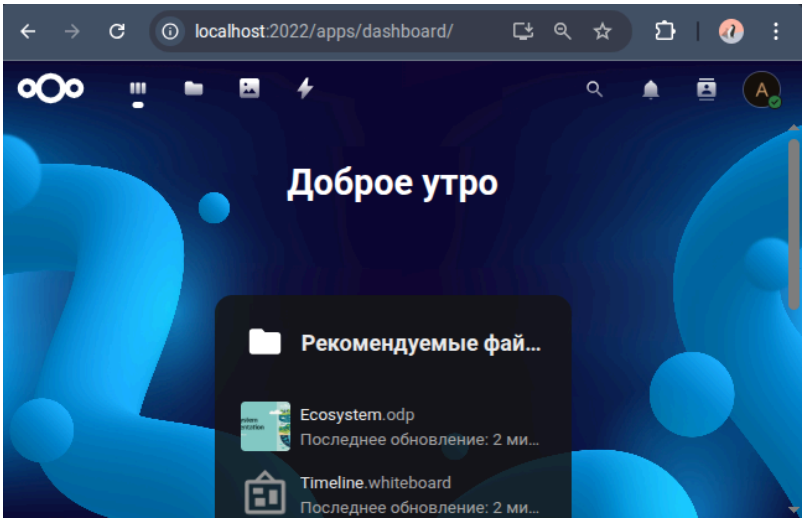

Рисунок 20 – NextCloud, доступный через порт 2022

Судя по содержимому нового контейнера mariadb — инициализация прошла успешно, что видно на рисунке 21:

```
MariaDB [(none)]> use mariadb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mariadb]> SHOW TABLES;
+-----------------------------+
| Tables_in_mariadb           |
+-----------------------------+
| oc_accounts                 |
| oc_accounts_data            |
| oc_activity                 |
| oc_activity_mq              |
| oc_addressbookchanges       |
| oc_addressbooks             |
| oc_appconfig                |
| oc_appconfig_ex             |
| oc_authorized_groups        |
| oc_authtoken                |
| oc_bruteforce_attempts      |
| oc_calendar_invitations     |
| oc_calendar_reminders       |
| oc_calendar_resources       |
| oc_calendar_resources_md    |
| oc_calendar_rooms           |
| oc_calendar_rooms_md        |
```

Рисунок 21 – Таблицы NextCloud в MariaDB

8. Регистрация на Docker Hub

Выполнена регистрация на hub.docker.com, что видно на рисунке 22:

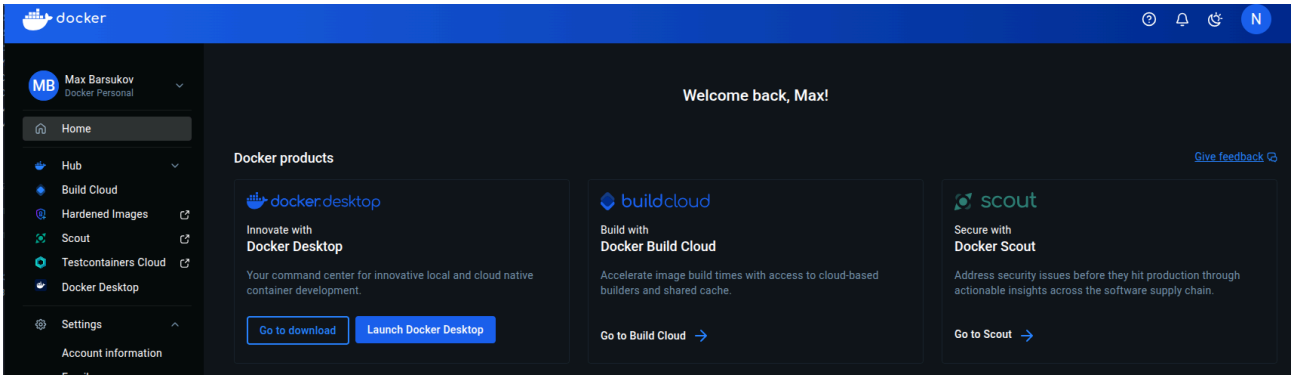

Рисунок 22 – Мой профиль на Docker Hub

9. Публикация образа на Docker Hub

Образ NextCloud помечен тегом и отправлен в публичный репозиторий, как показано на рисунках 23 и 24:

```
❯ docker tag nextcloud nyapsilon/nextcloud:v1.0
❯ docker push nyapsilon/nextcloud:v1.0
The push refers to repository [docker.io/nyapsilon/nextcloud]
72c5cbcb60d9: Mounted from library/nextcloud
569e3380c6c9: Mounted from library/nextcloud
ea88b097d252: Mounted from library/nextcloud
d39eaa5c1ce0: Mounted from library/nextcloud
aefebd58a16b: Mounted from library/nextcloud
f31beee0dec1: Mounted from library/nextcloud
c5d35f8473a3: Mounted from library/nextcloud
7717bbb8b074: Mounted from library/nextcloud
5f70bf18a086: Mounted from library/nextcloud
6da6ce241629: Mounted from library/nextcloud
e0be901b9885: Mounted from library/nextcloud
b0b0c4c00ce3: Mounted from library/nextcloud
eab84bfa29de: Mounted from library/nextcloud
bf8e40ac3c46: Mounted from library/nextcloud
cbf34dbfe55c: Mounted from library/nextcloud
5b90ee3c1b2c: Mounted from library/nextcloud
5e86fb8dd799: Mounted from library/nextcloud
5267a9f8a6b2: Mounted from library/nextcloud
a9cd4deb2ac4: Mounted from library/nextcloud
64013f07709c: Mounted from library/nextcloud
ec1c6a2202b4: Mounted from library/nextcloud
a70a53678e39: Mounted from library/nextcloud
d7c97cb6f1fe: Mounted from library/nginx
v1.0: digest: sha256:41a440846ad4ebc147b49f7347927f7944d141324ba05eb82e3d9685a488ad27 size: 5122
```

Рисунок 23 – Push образа в Docker Hub

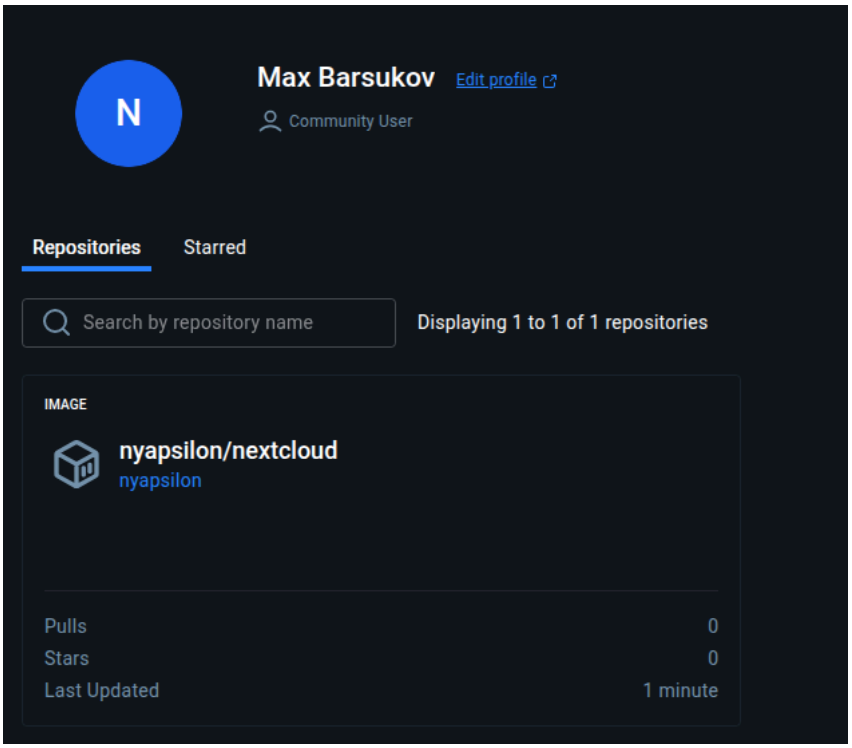

Рисунок 24 – Образ, опубликованный в публичном репозитории Docker Hub

10. Запуск приложения из Docker Hub (игра Super Mario)

Запущен публичный образ игры *docker-supermario*. Приложение доступно в браузере по порту 8082, как показано на рисунках 25-26:

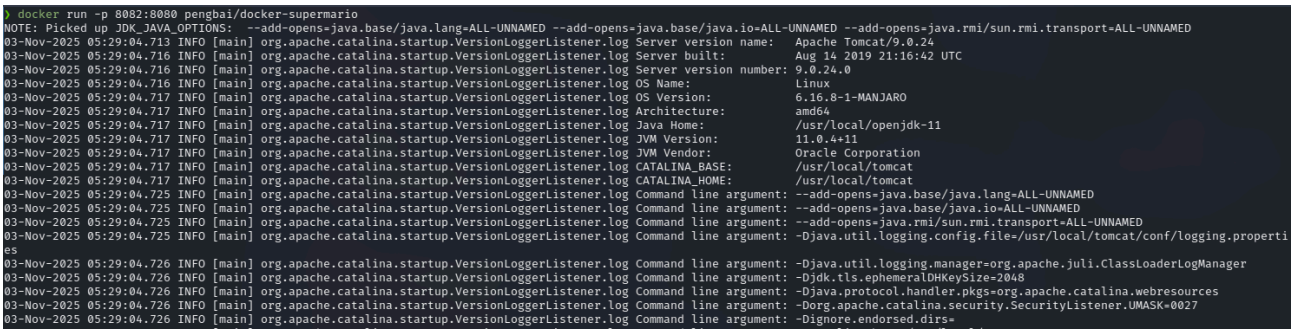

Рисунок 25 – Демонстрация запуска стороннего приложения из Docker Hub

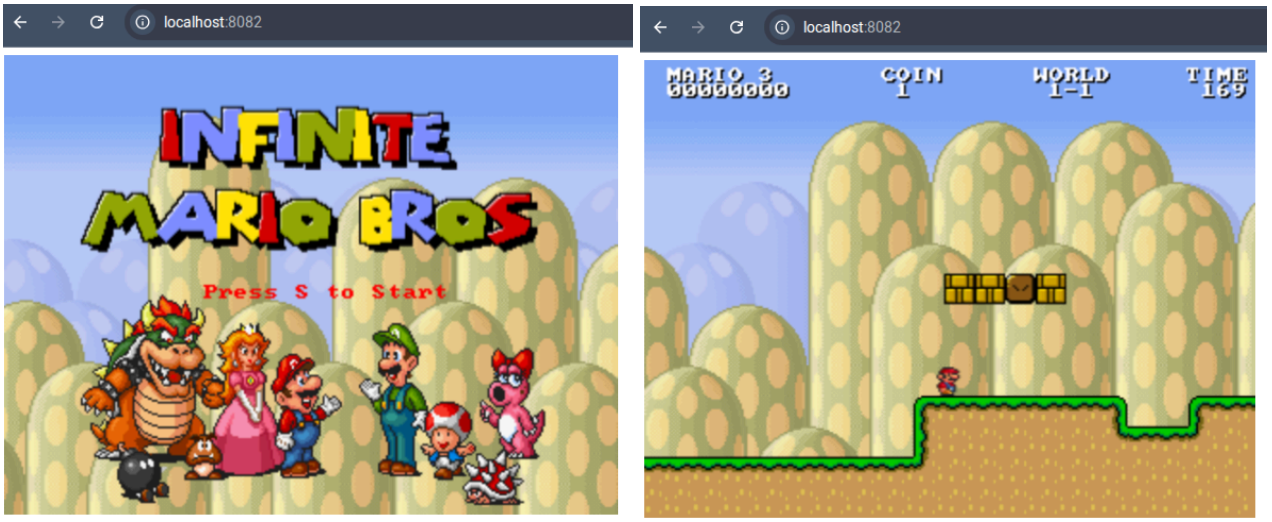

Рисунок 26 – Игра docker-supermario в браузере

11. Основные команды Docker

В ходе выполнения лабораторной работы были применены следующие команды Docker для управления контейнерами, образами и многоконтейнерными приложениями.

Команды для управления контейнерами:

- docker create — создаёт контейнер из образа без его запуска.
- docker start — запускает ранее созданный или остановленный контейнер.
- docker run — создаёт и сразу запускает новый контейнер из образа.
- docker ps — отображает список только запущенных контейнеров.
- docker ps -a — отображает все контейнеры, включая остановленные.
- docker inspect — выводит подробную информацию о контейнере (сетевые настройки, тома, переменные окружения и др.).
- docker logs — показывает журнал (логи) работы контейнера.
- docker stop — корректно останавливает контейнер, отправляя сигнал завершения (SIGTERM, затем при необходимости SIGKILL).
- docker kill — немедленно завершает работу контейнера принудительно (SIGKILL).
- docker rm — удаляет остановленный контейнер из системы.
- docker exec — выполняет указанную команду внутри уже запущенного контейнера.

Команды для управления образами:

- docker build — собирает образ на основе инструкций из файла Dockerfile.
- docker pull — загружает образ из реестра (например, с Docker Hub).
- docker push — отправляет локальный образ в удалённый реестр.
- docker images (или docker image ls) — отображает список всех локальных образов.
- docker image inspect — выводит детальную информацию о метаданных и конфигурации образа.
- docker rmi (или docker image rm) — удаляет указанный образ из локального хранилища.
- docker login — выполняет аутентификацию в Docker Hub или другом реестре контейнеров.

Команды Docker Compose:

- docker-compose up — создаёт и запускает все сервисы, описанные в файле docker-compose.yml.
- docker-compose down — останавливает и удаляет контейнеры, сети и тома, созданные через docker-compose up.
- docker-compose ps — отображает статус всех сервисов в проекте.
- docker-compose logs — выводит журналы всех запущенных сервисов.
- docker-compose build — пересобирает образы, указанные в конфигурации Compose.

Прочие команды:

- docker system df — отображает использование дискового пространства Docker (образы, контейнеры, тома).
- docker volume ls — перечисляет все тома, созданные Docker.
- docker network ls — перечисляет все сети, управляемые Docker.
- docker version — показывает версии Docker Engine и других компонентов.
- docker info — выводит подробную информацию о конфигурации и состоянии Docker-окружения.

Заключение

В ходе выполнения лабораторной работы были успешно освоены ключевые аспекты работы с платформой Docker — от базовых операций с контейнерами до развертывания многокомпонентных приложений.

Практически реализованы все поставленные задачи: создан и запущен собственный образ веб-сервера Nginx с пользовательской HTML-страницей, развернута СУБД MariaDB, запущен и настроен сервис облачного хранилища NextCloud с регистрацией пользователей. Также продемонстрирована работа с phpVirtualBox как примером веб-интерфейса для управления виртуализацией.

С помощью Docker Compose было собрано единое приложение, состоящее из NextCloud, Nginx и MariaDB, с корректной настройкой сетевого взаимодействия. Выполнена регистрация на Docker Hub, создан и опубликован собственный образ.

Таким образом, цель лабораторной работы — изучение основ Docker и приобретение практических навыков развертывания, настройки и публикации контейнеризованных приложений — была достигнута в полном объеме.