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: P34102

Выполнил:

Лапин А.А.

Юнусов Р.Э.

Проверил:

к.т.н. преподаватель Белозубов А.В.

Санкт-Петербург

2024г.

Оглавление

[Введение 3](#_Toc184071562)

[Описание работы и инструментов 4](#_Toc184071563)

[Основные компоненты 4](#_Toc184071564)

[Используемые инструменты 5](#_Toc184071565)

[Цели и задачи 5](#_Toc184071566)

[Текст задания 6](#_Toc184071567)

[Выполнение 7](#_Toc184071568)

[Шаг 1. Установка Docker и VirtualBox в ОС c Unix системой 7](#_Toc184071569)

[Установка Docker на Debian 7](#_Toc184071570)

[Шаг 2. Проверка установки Docker. Запуск Docker Hello 9](#_Toc184071571)

[Шаг 3. Скачать и запустить контейнер с приложением Nginx 10](#_Toc184071572)

[Шаг 4. Скачать и запустить контейнер с приложением MariaDB 14](#_Toc184071573)

[Шаг 5. Скачать и запустить контейнер с приложением NextCloud 17](#_Toc184071574)

[Шаг 6. Скачать и запустить контейнер с приложением phpvirtualbox 21](#_Toc184071575)

[Шаг 7. Создание проекта Docker Compose для NextCloud 24](#_Toc184071576)

[Шаг 8. Регистрация в Docker Hub 30](#_Toc184071577)

[Шаг 9. Выложить свой проект на DockerHub. 31](#_Toc184071578)

[Шаг 10. Запуск проекта из DockerHub 31](#_Toc184071579)

[Шаг 11. Основные использованные команды Docker 33](#_Toc184071580)

[Заключение 35](#_Toc184071581)

[Список литературы 36](#_Toc184071582)

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 c 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 c 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

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок . 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

1. 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

1. 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

1. 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 перелогиниться, чтобы применились изменения в группах пользователей:

newgrp docker # Применить изменения группы без перелогина

Шаг 2. Проверка установки Docker. Запуск Docker Hello

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок . 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

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . 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

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

Рисунок . 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.

1. Создание рабочей директории на хосте

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

mkdir -p ~/nginx/html

1. 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>Добро пожаловать!</h1>

    <p>Это страница по умолчанию, созданная Алексем Лапиным.</p>

</body>

</html>

Сохраните этот файл в каталоге ~/nginx/html.

1. Проверка работы веб-сервера

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

Изображение выглядит как текст, снимок экрана, компьютер, программное обеспечение

Автоматически созданное описание

Рисунок . Страница по умолчанию Nginx

1. Управление контейнером

Проверка запущенных контейнеров:

docker ps

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Проверка запущенных контейнеров

Остановка контейнера:

docker stop my-nginx

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Остановка контейнера

Запуск остановленного контейнера:

docker start my-nginx

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Запуск контейнера

**Анализ результатов**

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

1. Развертывание 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

1. 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

1. Проверка статуса контейнера:

docker ps

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Проверка статуса контейнера MariaDB

1. Подключение к базе данных через терминал:

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

После ввода пароля вы попадете в интерактивную консоль MariaDB.

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

CREATE DATABASE lapin;

1. Проверка созданной базы данных:

SHOW DATABASES;

1. Предоставление прав пользователю:

GRANT ALL PRIVILEGES ON lapin.\* TO 'lapin'@'%';

FLUSH PRIVILEGES;

1. Выход из консоли MariaDB:

EXIT;

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок . 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

1. 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 - монтирование 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.

1. Проверка статуса контейнера:

docker ps

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок . Проверка статуса контейнера NextCloud

1. Первоначальная настройка NextCloud:
   1. Откроем браузер и перейдем по адресу `http://localhost:8088`
   2. При первом запуске вы увидите страницу создания учетной записи администратора

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Рисунок . Страница первоначальной настройки NextCloud

1. 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:
   1. Имя пользователя: belozubov
   2. Отображаемое имя: Белозубов А. В.
   3. Пароль: [надежный пароль]
5. Нажмем "Создать"

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Добавление нового пользователя

1. Проверим работоспособность:

Загрузим картинку для проверки хранилища

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок . 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. Развертывание 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

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Рисунок . Проверка версии VirtualBox

1. 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

1. Запустим VirtualBox Web Service:

vboxwebsrv -H 0.0.0.0 -v

1. 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

1. Запустим контейнер:

docker-compose up -d

1. Проверим статус контейнера:

docker ps

1. 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:

Изображение выглядит как текст, снимок экрана, компьютер, программное обеспечение

Автоматически созданное описание

Рисунок . 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. Развертывание 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}

1. 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;

}

}

1. 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

1. 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

1. 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

1. Сетевая конфигурация:

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

1. Безопасность:

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

1. Отказоустойчивость:

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

1. Запуск проекта:

docker-compose up -d

1. Проверка статуса контейнеров:

docker-compose ps

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок . Статус запущенных контейнеров

1. 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

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок . 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. Развертывание 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

1. 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
2. Проверка функциональности:

* Успешная регистрация администратора
* Проверена загрузка и хранение файлов
* Подтверждена работоспособность веб-интерфейса
* Система доступна через порт 2022

1. Выводы:

* Создана полноценная, масштабируемая инфраструктура
* Достигнута высокая степень изоляции компонентов
* Обеспечена отказоустойчивость через автоматический перезапуск контейнеров

Шаг 8. Регистрация в Docker Hub

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок . Регистрация в Docker Hub

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок . Вход в Docker Hub

Шаг 9. Выложить свой проект на DockerHub.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок . Выгрузка проекта на DockerHub

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок . Репозиторий на DockerHub

Шаг 10. Запуск проекта из DockerHub

Я выбрал игру 2048.

Изображение выглядит как электроника, текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок . Скачивание игры 2048 с DockerHub

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок . 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)

1. Работа с Docker Hub:

* Успешно выполнена регистрация на платформе
* Освоен процесс публикации собственных образов
* Получен опыт использования публичных образов

1. Практическое применение:

* Успешно развернут контейнер с игрой 2048
* Настроено проксирование портов для доступа к приложению
* Продемонстрирована работоспособность контейнеризированного приложения

1. 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
   * Развернут 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. Развертывание 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.