МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт Компьютерных наук
Кафедра Прикладной математики

ЛАБОРАТОРНАЯ РАБОТА

По дисциплине: «Операционные системы Linux».
На тему: «Контейниризация».

Студент ПМ-22 (группа) _____ (подпись, дата) Борисов А. В. (фамилия, инициалы)

Руководитель

к. т. н. (ученая степень, ученое звание) _____ (подпись, дата) Кургасов В. В. (фамилия, инициалы)

Липецк 2024

Цель работы:

Ознакомление с программным обеспечением удаленного доступа к распределенным системам обработки данных.

Задачи:

Изучить теоретический материал и выполнить предложенные практические

задания.

В результате необходимо:

- Знать назначение и возможности Docker;

- Знать особенности установки и настройки Docker;

- Владеть инструментом для определения и запуска многоконтейнерных приложений

Docker – Docker Compose

Ход работы:

1. **Установка Docker**

```
exerted@fedora:~$ sudo dnf -y install dnf-plugins-core
sudo dnf-3 config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo
[sudo] пароль для exerted:
Fedora 40 - x86_64 - Updates                    6.5 kB/s | 7.1 kB     00:01
Fedora 40 - x86_64 - Updates                    1.1 MB/s | 4.1 MB     00:03
Последняя проверка окончания срока действия метаданных: 0:00:07 назад, Вт 10 дек 2024 11:51:36.
Пакет dnf-plugins-core-4.10.0-1.fc40.noarch уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
Добавление репозитория из: https://download.docker.com/linux/fedora/docker-ce.repo
exerted@fedora:~$ sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Docker CE Stable - x86_64                       9.2 kB/s | 3.5 kB     00:00
Пакет docker-ce-3:27.4.0-1.fc40.x86_64 уже установлен.
Пакет docker-ce-cli-1:27.4.0-1.fc40.x86_64 уже установлен.
Пакет containerd.io-1.7.24-3.1.fc40.x86_64 уже установлен.
Пакет docker-buildx-plugin-0.19.2-1.fc40.x86_64 уже установлен.
Пакет docker-compose-plugin-2.31.0-1.fc40.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
exerted@fedora:~$ sudo systemctl enable --now docker
```

2. **Создание структуры проекта**

```
exerted@fedora:~$ tree docker
docker
├── docker-compose.yml
├── html
│   └── index.php
├── mysql
│   └── data
├── nginx
│   └── conf.d
│       └── default.nginx
├── php
│   ├── Dockerfile
│   └── php.ini
└── proxy
    └── docker-compose.yml

8 directories, 6 files
```

3. **Устанавливаем Nginx и напишем compose файл в корневой папке проекта**

```yaml
version: '3.0'

services:
  nginx:
    image: nginx:latest
    container_name: container_nginx
    ports:
      - "80:80"
    networks:
      - frontend
      - backend

networks:
    frontend:
      driver: bridge
    backend:
      driver: bridge
```

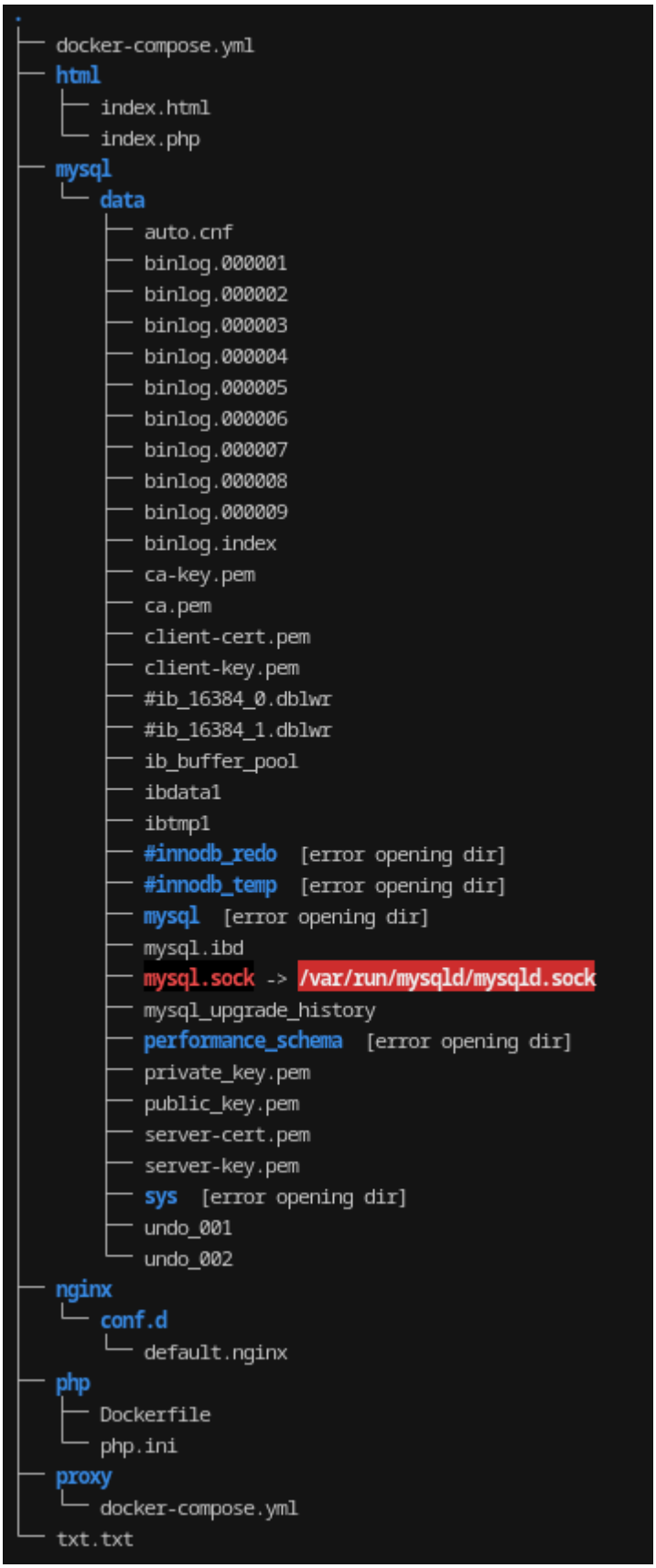
4. **Проверим и перейдем по http://localhost**

5. **Передача в контейнер html-файла с произвольным текстом, с помощью volumes**

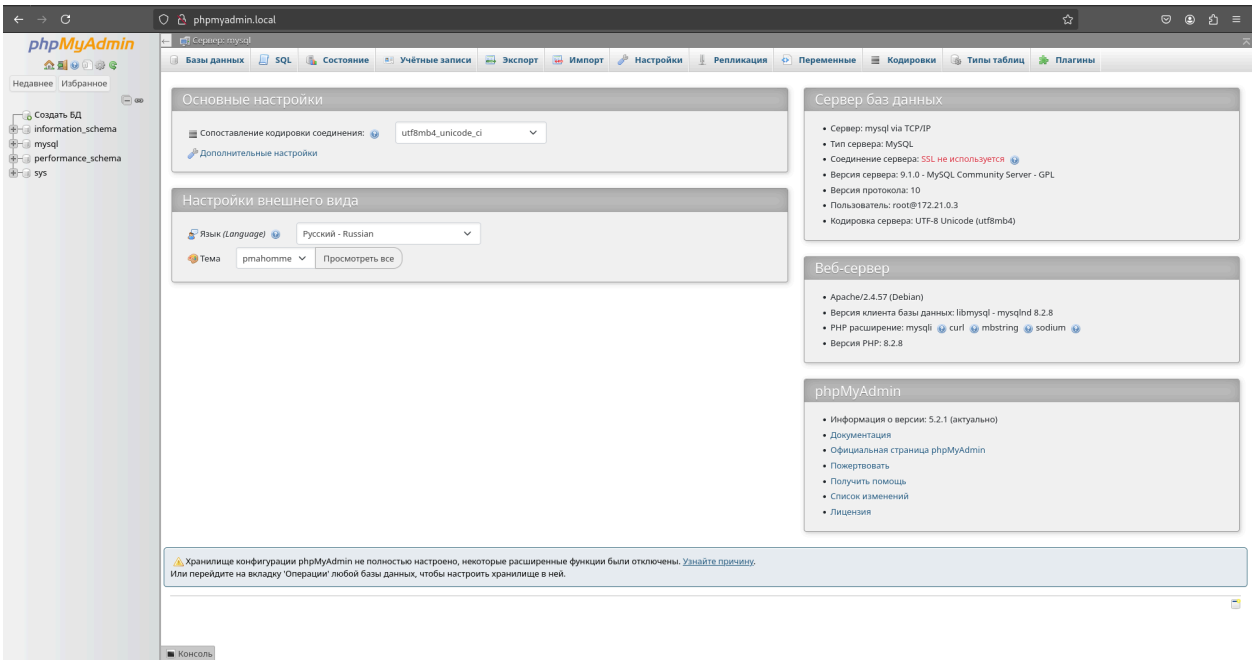
```yaml
version: '3.0'

services:
  nginx:
    image: nginx:latest
    container_name: container_nginx
    ports:
      - "80:80"
    volumes:
      - ./html/:/usr/share/nginx/html/
    networks:
      - frontend
      - backend

networks:
    frontend:
      driver: bridge
    backend:
      driver: bridge
```

Some text

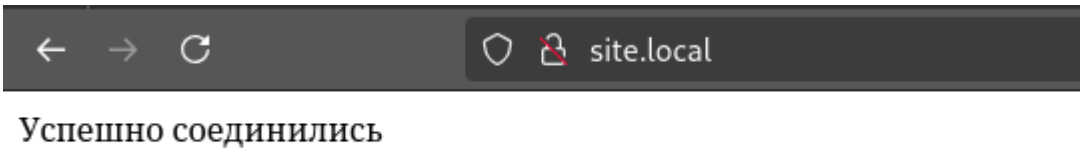
6. **Конечная структура проекта:**



7. Переход по phpmyadmin.local



8. Переход по site.local



Успешно соединились

9. Собрать образ с Wordpress

```yaml
version: '3.0'
services:
  nginx:
    image: nginx
    environment:
      - VIRTUAL_HOST=site.local
    depends_on:
      - wordpress
    volumes:
      - ./nginx/conf.d/default.nginx:/etc/nginx/conf.d/default.conf
      - ./html/:/var/www/html/
    networks:
      - frontend
      - backend
  wordpress:
    image: wordpress:latest
    volumes:
      - ./html:/var/www/html
    depends_on:
      - mysql
    environment:
      HOST: mysql:3306
      USER: root
      PASS: root
      NAME: wordpress
    networks:
      - frontend
      - backend
  mysql:
```

Вывод:

В ходе лабораторной работы была разработана и протестирована программа сортировки элементов массива на языке Rust. Для написания программы был использован текстовый редактор vim, а компиляция выполнена с помощью утилиты rustc. Было получено представление о разработке на системе Linux.

Контрольные вопросы:

1. **Назовите отличия использования контейнеров по сравнению с виртуализацией.**

A) Меньшие накладные расходы на инфраструктуру

C) Невозможность запуска GNU/Linux- и Windows-приложений на одном хосте

2. **Назовите основные компоненты Docker**

   Все кроме гипервизоров.

3. **Какие технологии используются для работы с контейнерами?**

   Контрольные группы и пространства имен.

4. **Найдите соответствие между компонентом и его описанием:**

   Реестры (репозитории) — сетевые хранилища образов

   Контейнеры - изолированные при помощи технологий операционной системы пользовательские окружения, в которых выполняются приложения.

   Образы — доступные только для чтения шаблоны приложений

5. **В чем отличие контейнеров от виртуализации?**

   Контейнеры: работают на уровне операционной системы, используют общее ядро, поэтому легковесны и быстро запускаются. Приложения внутри контейнеров изолированы, но не требуют отдельной ОС.

   Виртуализация: использует гипервизор для создания виртуальных машин (ВМ), каждая из которых имеет свою

операционную систему. Это требует больше ресурсов, а время запуска ВМ больше, чем у контейнера.

6. **Перечислите основные команды утилиты Docker с их кратким описанием.**

   docker run

   docker ps

   docker stop

   docker rm

   docker pull

   docker push

   docker build

   docker exec

7. **Каким образом осуществляется поиск образов контейнеров?**

   Для поиска образов используется команда docker search, которая отправляет запрос в публичные реестры, такие как Docker Hub. Также можно просматривать реестры через веб-интерфейсы.

8. **Каким образом осуществляется запуск контейнера?**

   Запуск контейнера осуществляется командой docker run. Эта команда:

   Загружает указанный образ из реестра (если его нет локально).

   Создает экземпляр контейнера.

   Настраивает изоляцию (сеть, файловую систему, ресурсы).

   Запускает приложение внутри контейнера.

9. **Что значит управлять состоянием контейнеров?**

Управление состоянием контейнеров подразумевает выполнение следующих операций:

Запуск: docker start запускает остановленный контейнер.

Остановка: docker stop завершает работу контейнера.

Перезапуск: docker restart перезапускает контейнер.

Удаление: docker rm удаляет контейнер.

Мониторинг: docker ps позволяет отслеживать активные контейнеры.

10. **Как изолировать контейнер?**

Изоляция контейнера достигается с помощью следующих технологий:

Linux Namespaces: изолируют процессы, файловую систему, сеть и другие ресурсы.

cgroups: ограничивают использование ресурсов (CPU, память, диск, сеть).

Сетевые настройки: использование отдельных виртуальных сетей для контейнеров.

Частные файловые системы: каждый контейнер работает в своей файловой системе.

11. **Опишите последовательность создания новых образов, назначение Dockerfile.**

Dockerfile: это текстовый файл, содержащий инструкции для сборки образа.

Последовательность:

Создайте файл Dockerfile.

Укажите базовый образ (FROM).

Определите инструкции для копирования файлов, установки зависимостей, выполнения команд (RUN, COPY).

Постройте образ с помощью команды docker build.

Используйте созданный образ для запуска контейнеров.

Назначение Dockerfile: автоматизация создания образов для контейнеров.

12. **Возможно ли работать с контейнерами Docker без одновременного движка?**

Нет, Docker Engine необходим для работы с контейнерами. Он выполняет задачи по управлению образами, контейнерами и их сетями, предоставляя API и интерфейс командной строки.

13. **Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes.**

Назначение Kubernetes:

Kubernetes управляет развертыванием, масштабированием, обновлением и поддержанием состояния контейнеров в кластере. Это облегчает управление большими системами и обеспечивает автоматизацию.

Основные объекты:

Pod: минимальная единица Kubernetes, содержащая один или несколько контейнеров.

Service: обеспечивает сетевой доступ к Pod'ам.

Deployment: управляет развертыванием и обновлением Pod'ов.

Namespace: разделяет ресурсы в кластере для изоляции.

Volume: предоставляет постоянное хранилище для контейнеров.