

Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

Отчет по лабораторной работе № 6

«Контейнеризация»

по курсу «ОС Linux»

Студент
Группа ПМ-18

Полухина Е.Д.

Руководитель

Кургасов В.В.

Липецк 2020 г.

СОДЕРЖАНИЕ

Цель работы	3
Задание.....	4
Ход работы	5
1. Клонирование проекта.	5
2. Открытие проекта.....	6
3. Установка docker и docker-compose.	7
4. Создание Dockerfile.....	8
5. Создание файла docker-compose.yml.....	9
6. Заполнение базы данных.....	11
7. Создание файла docker-compose.yml для WordPress.	12
8. Создание каталогов для MySQL.....	14
9. Запуск Compose.	15
10. Просмотр сервисов.....	16
11. Настройка WordPress.....	17
Вывод	19

Цель работы

Изучить современные методы разработки ПО в динамических и распределенных средах на примере контейнеров Docker.

Задание

1. С помощью Docker Compose на своем компьютере поднять сборку nginx+php-fpm+postgres, продемонстрировать ее работоспособность, запустив внутри контейнера демо-проект на symfony. По умолчанию проект работает с sqlite-базой. Нужно заменить ее на postgres.
2. Заменить DATABASE_URL в .env на строку подключения к postgres.
3. Создать схему БД и заполнить ее данными из фикстур.

Проект должен открываться по адресу <http://demo-symfony.local/> (Код проекта должен располагаться в папке на локальном хосте). Для компонентов nginx, fpm есть готовые docker-образы, их можно и нужно использовать. Нужно расшарить папки с локального хоста, настроить подключение к БД. В .env переменных для postgres нужно указать путь к папке, где будет лежать база, чтобы она не удалялась при остановке контейнера. На выходе должен получиться файл конфигурации docker-compose.yml и .env файл с настройками переменных окружения.

4. Создание образа с Wordpress.

Ход работы

1. Клонирование проекта.

Клонируем себе тестовый проект с помощью команды «git clone <https://github.com/symfony/demo>». Затем перейдем в каталог с проектом с помощью команды «cd project1».

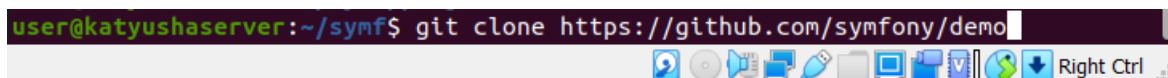
A terminal window with a dark background. The prompt is 'user@kasyushaserver:~/symf\$'. The command 'git clone https://github.com/symfony/demo' is entered and partially executed. The terminal has a taskbar at the bottom with various icons and the text 'Right Ctrl'.

Рисунок 1 – Клонирование проекта

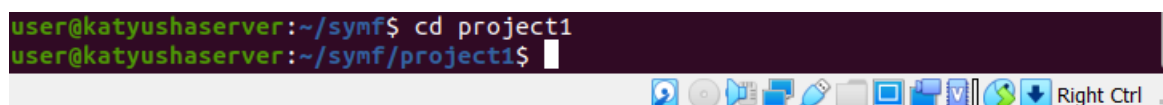
A terminal window with a dark background. The prompt is 'user@kasyushaserver:~/symf\$'. The command 'cd project1' is entered and executed. The prompt changes to 'user@kasyushaserver:~/symf/project1\$'. The terminal has a taskbar at the bottom with various icons and the text 'Right Ctrl'.

Рисунок 2 – Переход в каталог project1

2. Открытие проекта.

Запустим проект с помощью команды «`php -S 0.0.0.0:8000 -t public/`».

```
user@katyushaserver:~/symf/project1$ php -S 0.0.0.0:8000 -t public/  
[Wed Jan 6 11:17:39 2021] PHP 7.4.3 Development Server (http://0.0.0.0:8000) s  
tarted
```

Рисунок 3 – Запуск проекта

После этого в браузере будет доступен данный проект по адресу `http://localhost:8000`. Главное окно приложения имеет вид, представленный на рисунке 4.

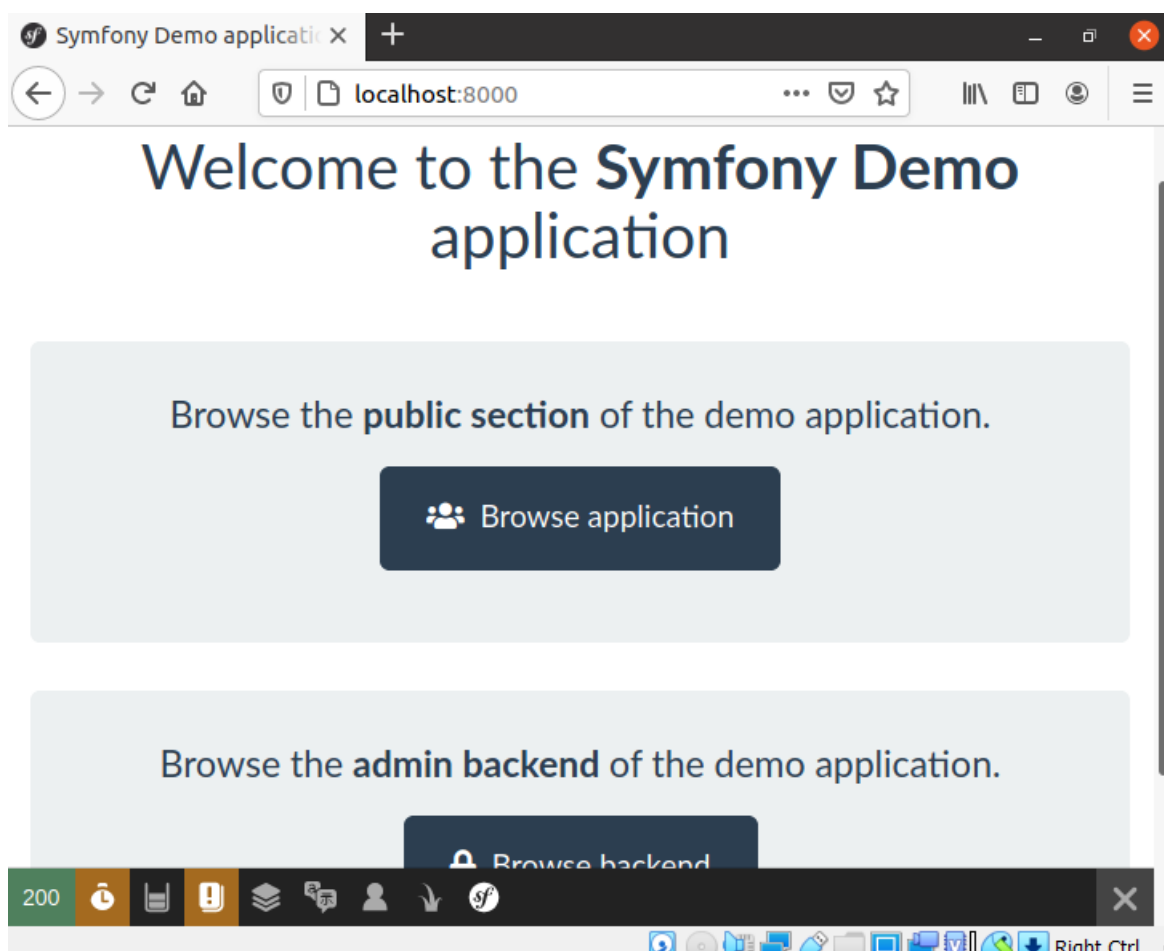


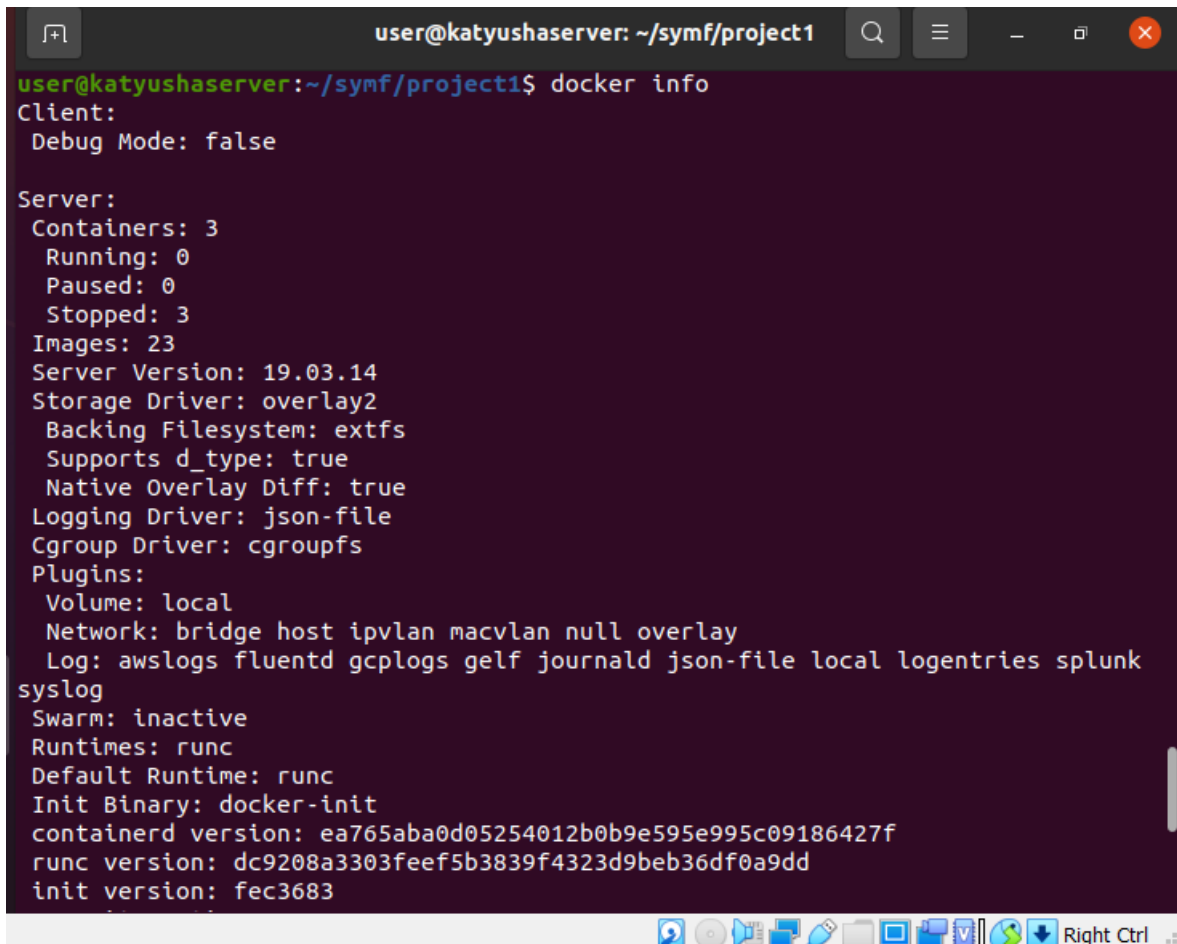
Рисунок 4 – Открытие проекта по адресу `http://localhost:8000`

3. Установка docker и docker-compose.

Установим docker и docker-compose посредством следующих команд:

- a) `sudo apt-get install docker-ce`
- b) `sudo curl -L https://github.com/docker/compose/releases/download/1.25.0-rc4/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose`
- c) `sudo chmod +x /usr/local/bin/docker-compose`
- d) `sudo ln -s /usr»/local/bin/docker-compose /usr/bin/docker-compose`

С помощью команды «`docker info`» проверим, действительно ли установился docker.



```
user@katyushaserver: ~/symf/project1
user@katyushaserver:~/symf/project1$ docker info
Client:
 Debug Mode: false

Server:
 Containers: 3
  Running: 0
  Paused: 0
  Stopped: 3
 Images: 23
 Server Version: 19.03.14
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Native Overlay Diff: true
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk
 syslog
 Swarm: inactive
 Runtimes: runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: ea765aba0d05254012b0b9e595e995c09186427f
 runc version: dc9208a3303feef5b3839f4323d9beb36df0a9dd
 init version: fec3683
```

Рисунок 5 – Результат выполнения команды `docker info`

4. Создание Dockerfile.

В папке с проектом создаем файл Dockerfile и заполняем его следующим содержимым:

```
FROM richarvey/nginx-php-fpm
WORKDIR /var/www/html/demo
COPY composer.json ./
RUN composer install
COPY . .
EXPOSE 8000
CMD ["php", "-S", "0.0.0.0:8000", "-t", "public/"]
```

На рисунке **Error! Reference source not found.** показано содержимое файла Dockerfile.

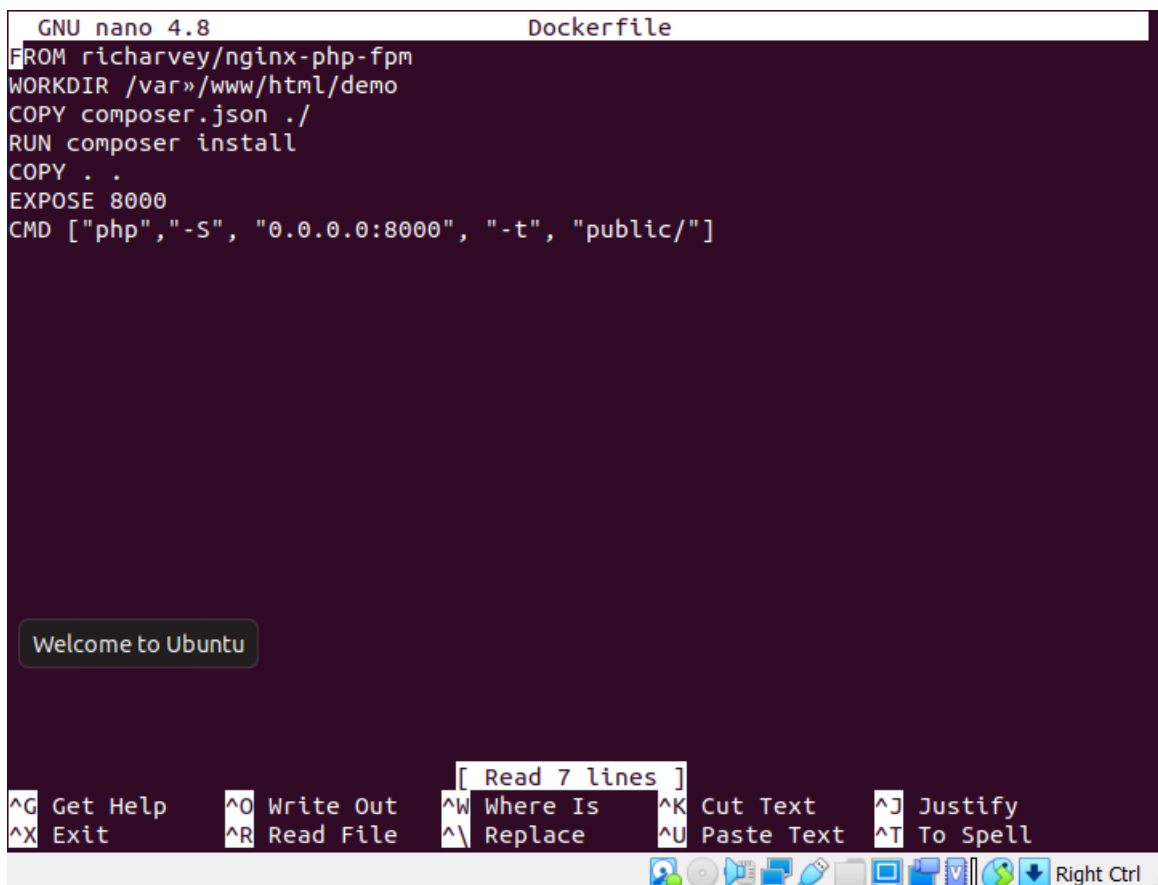


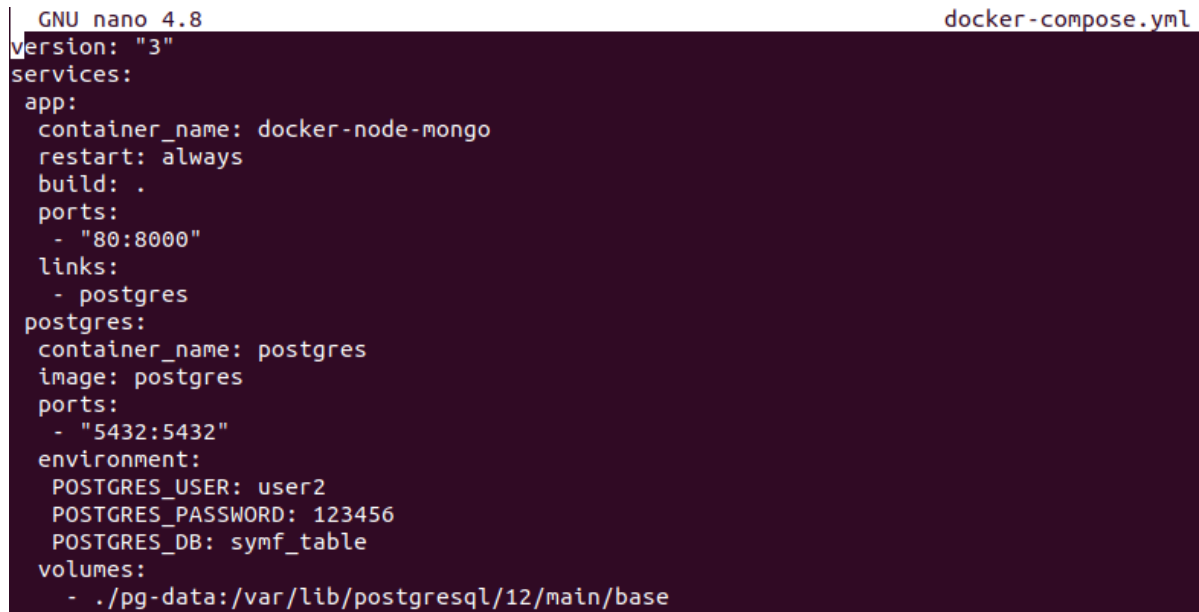
Рисунок 6 – Содержимое файла Dockerfile

5. Создание файла docker-compose.yml.

В папке с проектом создаем файл docker-compose.yml с помощью команды «nano docker-compose.yml» и заполняем его следующим содержимым:

```
version: "3"
services:
  app:
    container_name: docker-node-mongo
    restart: always
    build: .
    ports:
      - "80:8000"
    links:
      - postgres
  postgres:
    container_name: postgres
    image: postgres
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: 123456
      POSTGRES_DB: symf_table
    volumes:
      - ./pg-data:/var/lib/postgresql/12/main/base
```

На рисунке 7 показано содержимое файла docker-compose.yml.



```
GNU nano 4.8                                     docker-compose.yml
version: "3"
services:
  app:
    container_name: docker-node-mongo
    restart: always
    build: .
    ports:
      - "80:8000"
    links:
      - postgres
  postgres:
    container_name: postgres
    image: postgres
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: user2
      POSTGRES_PASSWORD: 123456
      POSTGRES_DB: symf_table
    volumes:
      - ./pg-data:/var/lib/postgresql/12/main/base
```

Рисунок 7 – Содержимое файла docker-compose.yml

6. Заполнение базы данных.

Заполняем созданную базу данных данными с помощью команд:

```
php bin/console doctrine:schema:create
```

```
php bin/console doctrine:fixtures:load
```

После этого вводим команду «docker-compose up» и ожидаем выполнения скачивания образов и установки зависимостей composer.

После этого видим в браузере наше приложение по адресу localhost.local.

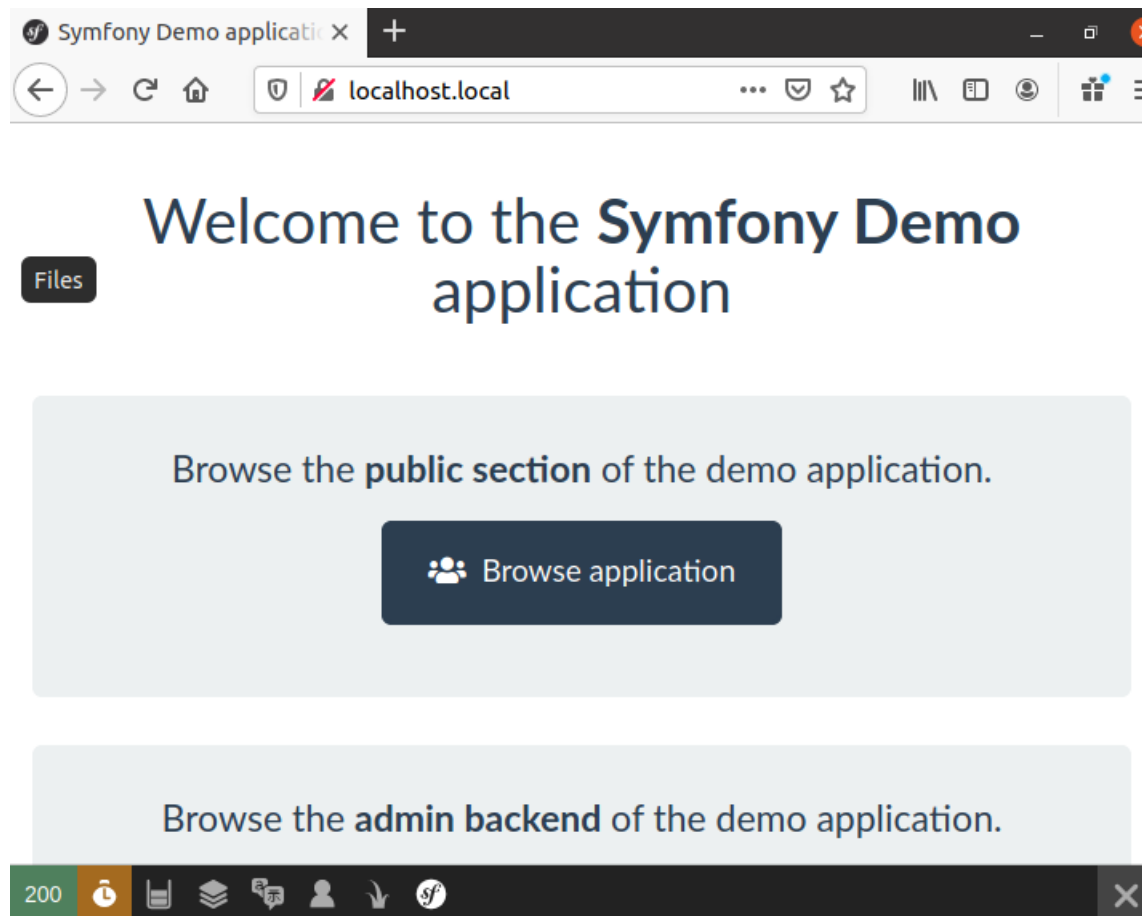


Рисунок 8 – Открытие проекта по адресу http://localhost.local

7. Создание файла docker-compose.yml для WordPress.

Создаем каталог wordpress. Создаем в нем файл docker-compose.yml с помощью команды «nano docker-compose.yml» и заполняем его следующим содержимым:

```
version: '3.3'
services:
  wordpress:
    image: wordpress:latest
    restart: always
    links:
      - db:mysql
    ports:
      - "80:80"
    working_dir: /var/www/html
    volumes:
      - "/opt/wp-content:/var/www/html/wp-content"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

  db:
    image: mysql:5.7
    restart: always
    volumes:
      - "/opt/mysql:/var/lib/mysql"
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
```

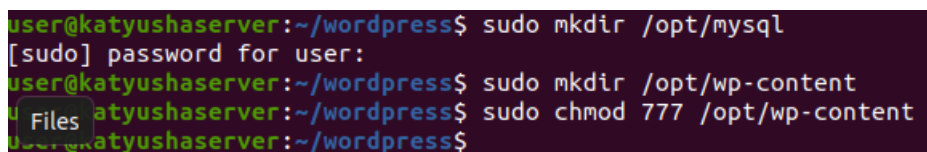
```
GNU nano 4.8                                docker-compose.yml                Modified
version: '3.3'
services:
  wordpress:
    image: wordpress:latest
    restart: always
    links:
      - db:mysql
    ports:
      - "80:80"
    working_dir: /var/www/html
    volumes:
      - "/opt/wp-content:/var/www/html/wp-content"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
  db:
    image: mysql:5.7
    restart: always
File Name to Write: docker-compose.yml
^G Get Help      M-D DOS Format  M-A Append      M-B Backup File
^C Cancel        M-M Mac Format  M-P Prepend     ^T To Files
```

Рисунок 9 – Содержимое файла docker-compose.yml

8. Создание каталогов для MySQL.

Создаем каталоги для локального хранения файлов плагинов, контента и базы данных MySQL с помощью команд «`sudo mkdir /opt/mysql`» и «`sudo mkdir /opt/wp-content`».

После этого изменяем права доступа на каталог wp-content с помощью команды «`sudo chmod 777 /opt/wp-content`».

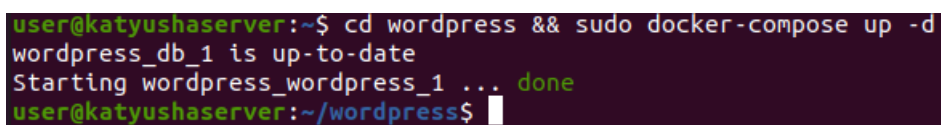
A screenshot of a terminal window with a dark background. The prompt is 'user@katyushaserver:~/wordpress\$'. The first command is 'sudo mkdir /opt/mysql', followed by a password prompt '[sudo] password for user:'. The second command is 'sudo mkdir /opt/wp-content'. The third command is 'sudo chmod 777 /opt/wp-content'. The prompt returns to 'user@katyushaserver:~/wordpress\$'.

```
user@katyushaserver:~/wordpress$ sudo mkdir /opt/mysql
[sudo] password for user:
user@katyushaserver:~/wordpress$ sudo mkdir /opt/wp-content
user@katyushaserver:~/wordpress$ sudo chmod 777 /opt/wp-content
user@katyushaserver:~/wordpress$
```

Рисунок 10 – Создание каталогов для MySQL и изменение прав доступа

9. Запуск Compose.

Запускаем Compose с помощью команды «cd wordpress && sudo docker-compose up -d».



```
user@katyushaserver:~$ cd wordpress && sudo docker-compose up -d
wordpress_db_1 is up-to-date
Starting wordpress_wordpress_1 ... done
user@katyushaserver:~/wordpress$
```

Рисунок 11 – Запуск Compose

10. Просмотр сервисов.

После запуска Compose с помощью команды «`sudo docker ps`» смотрим существующие сервисы.

```
user@katiushaserver:~/wordpress$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
5e9028e92473   wordpress:latest "docker-entrypoint.s..." 2 minutes ago  Up 59 seconds  0.0.0.0:80->80/tcp      wordpress_wordpress_1
78a9fc72fa27   mysql:5.7      "docker-entrypoint.s..." 2 minutes ago  Up 2 minutes   3306/tcp, 33060/tcp     wordpress_db_1
user@katiushaserver:~/wordpress$
```

Рисунок 12 – Просмотр сервисов

11. Настройка WordPress.

Переходим в браузере по адресу `http://localhost.local` и выполняем настройку WordPress.

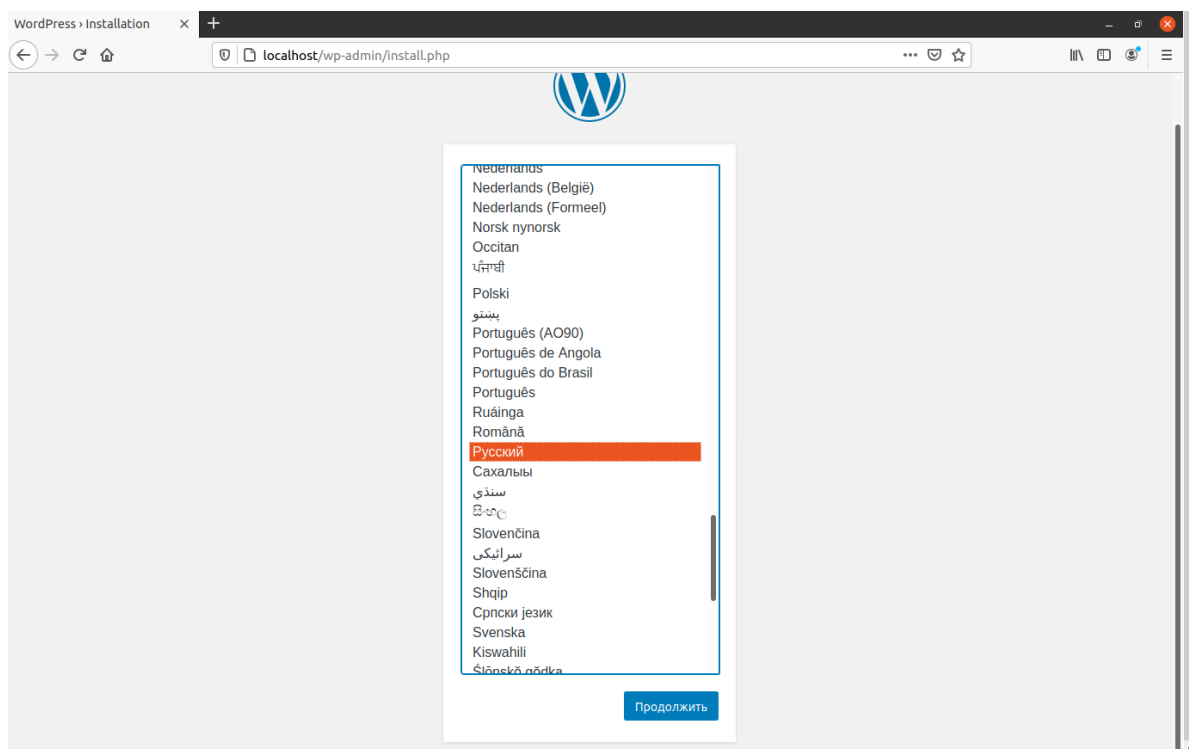


Рисунок 13 – Настройка WordPress

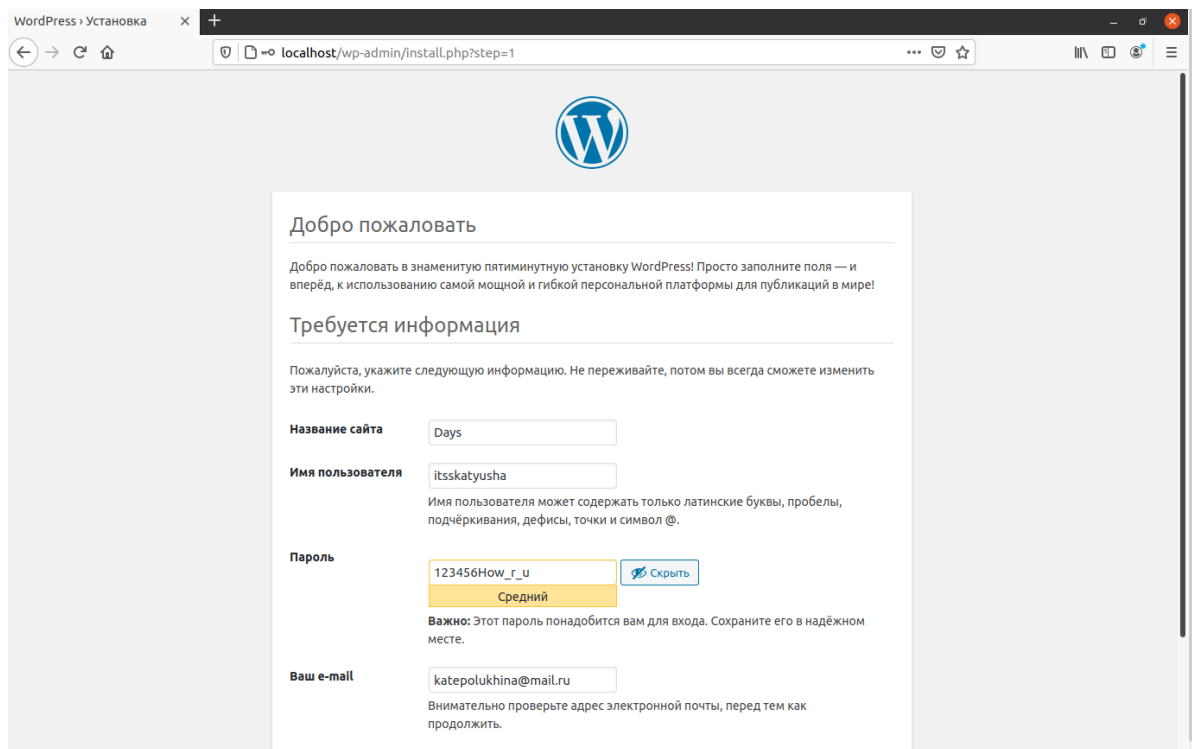


Рисунок 14 – Настройка WordPress

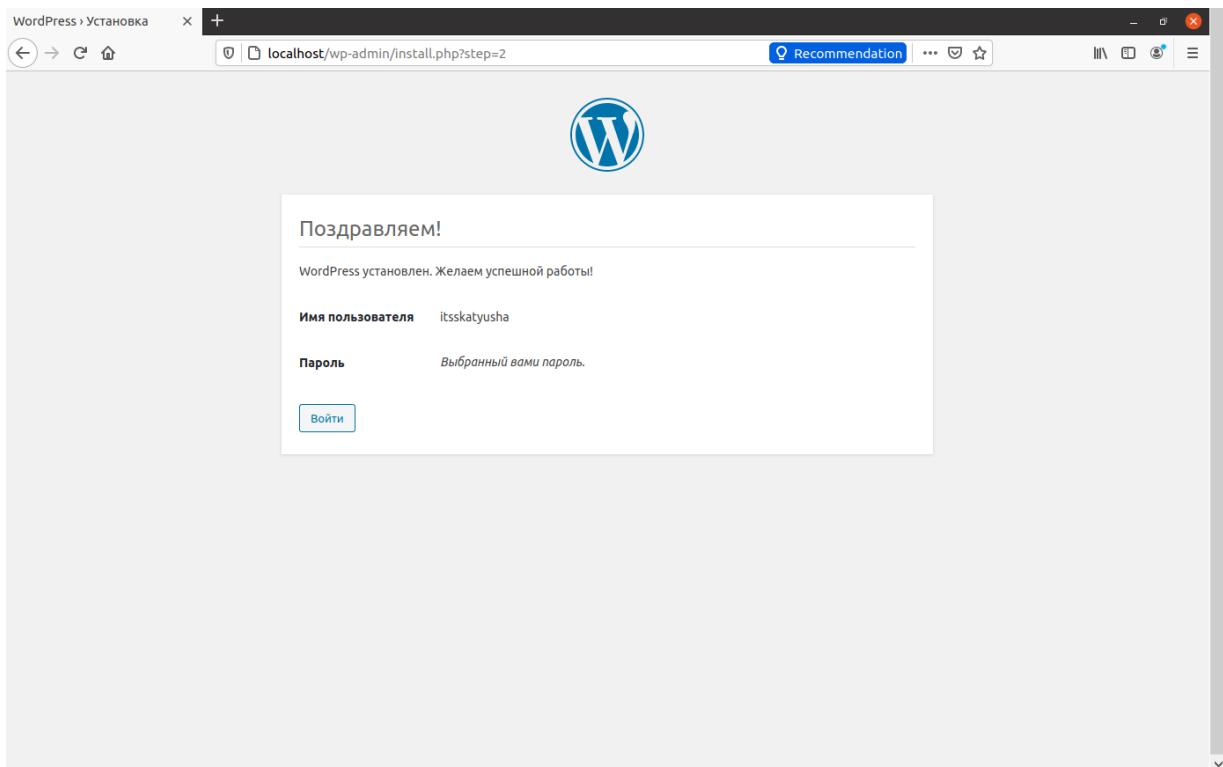


Рисунок 15 – Успешная установка WordPress

Вывод

В результате выполнения лабораторной работы я получила знания по контейнеризации. Получила навыки по клонированию проектов. Также научилась заполнять такие файлы как «Dockerfile», «docker-compose.yml», «.env». Поняла, что нужно указывать путь к папке, где будет лежать база данных, чтобы она не удалялась при остановке контейнера.

Научилась открывать нужный проект по адресу <http://localhost>. Освоила создание каталогов для MySQL, docker-compose.yml для WordPress и создала образ с WordPress.