

# Липецкий государственный технический университет

Факультет автоматизации и информатики

Автоматизированные системы управления

## ЛАБОРАТОРНАЯ РАБОТА №5 – ЧАСТЬ 2

по курсу ДПО Интаро - Linux

«Контейнеризация»

Студент

Митина М. В.

Группа     ПИ-20-1

Руководитель  
доц.

Кургасов В.В.

Липецк 2022 г.

## Цель работы

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

## Задание кафедры

1. С помощью Docker Compose на своем компьютере поднять сборку nginx+phpfpm+postgres, продемонстрировать ее работоспособность, запустив внутри контейнера демо-проект на symfony (Исходники взять отсюда <https://github.com/symfony/demo>). По умолчанию проект работает с sqlite-базой. Нужно заменить ее на postgres:

1. Создать новую БД в postgres;
2. Заменить DATABASE\_URL в /.env на строку подключения к postgres;
3. Создать схему БД и заполнить ее данными из фикстур, выполнив в консоли ( `php bin/console doctrine:schema:create` `php bin/console doctrine:fixtures:load`)). Проект должен открываться по адресу `http://demo-symfony.local/` (Код проекта должен располагаться в папке на локальном хосте) контейнеры с fpm и nginx должны его подхватывать. 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 постгреса 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

Дополнительные требования: Postgres также должен работать внутри контейнера. В .env переменных нужно указать путь к папке на локальном хосте, где будут лежать файлы БД, чтобы она не удалялась при остановке контейнера.

Выполняется на UBUNTU.

## Задание 2:

Шаг №1. Установка Nginx Для начала необходимо установить один лишь Nginx. Что требует создания compose-файла включая директиву ports, иначе порт будет доступен только внутри контейнера и nginx через браузер уже будет недоступен.

Шаг №2. Передача в контейнер html-файлов. В этом нам поможет volumes, которая говорит, что происходит монтирование локальной папки в контейнер по указанному адресу. При монтировании папка по указанному адресу внутри контейнера заменяется папкой с локального компьютера. Необходимо создать папку html на одном уровне с docker-compose.yml и добавить в нее файл index.html с произвольным текстом «Ваш текст», после чего пересоздадим контейнер (docker-compose up -d).

Шаг 3. Web-разработка. Создать папку проху и в ней сборку docker-compose.yml для обращения по домену и пробросу такого домена на основной контейнер. И сборку nginx, php, mysql и phpmyadmin с использованием проху сети.

Шаг 4. Имеется работающий Web-сервер. Создайте образ с одним из движков (WordPress, Joomla). Папка для хранения внешних данных с курсами должна быть Вами определена.

## Оглавление

Ход работы .....	5
Установка необходимых пакетов .....	5
Установка Docker и docker compose: .....	5
Установка php v8.1. ....	5
Установка Composer.....	6
Установка symfony: .....	6
Настройка проекта.....	8
Запуск сервера symfony и проекта demo.....	8
Настройка docker, сборка проекта.....	14
Задание 2 .....	20
Вывод.....	25
Контрольные вопросы .....	26
Приложения .....	32

## Ход работы

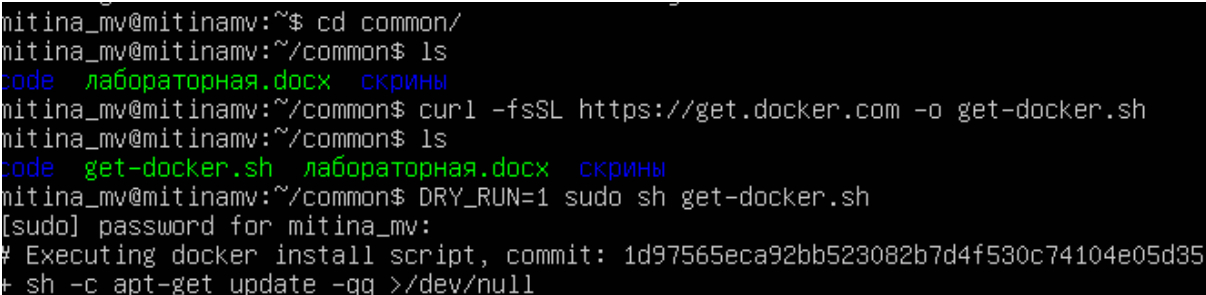
В этот раз лабораторная работа выполняется на Ubuntu, поэтому для начала в Virtual Box мне нужно создать машину с Ubuntu. Скачиваю образ Ubuntu server и произвожу все необходимые настройки. Кроме того, для удобства я сразу добавила общую папку между виртуальной машиной и хостовой, внутри ВМ она называется common. В ней выполняется вся лабораторная.

## Установка необходимых пакетов

### Установка Docker и docker compose:

```
curl -fsSL https://get.docker.com -o get-docker.sh
DRY_RUN=1 sudo sh ./get-docker.sh
```

Мы скачиваем файл со скриптом для установки docker и всех необходимых пакетов, а после запускаем этот скрипт на выполнение. Шаг аналогичен установке на Debian.



```
mitina_mv@mitinamv:~$ cd common/
mitina_mv@mitinamv:~/common$ ls
code лабораторная.docx скрины
mitina_mv@mitinamv:~/common$ curl -fsSL https://get.docker.com -o get-docker.sh
mitina_mv@mitinamv:~/common$ ls
code get-docker.sh лабораторная.docx скрины
mitina_mv@mitinamv:~/common$ DRY_RUN=1 sudo sh get-docker.sh
[sudo] password for mitina_mv:
# Executing docker install script, commit: 1d97565eca92bb523082b7d4f530c74104e05d35
+ sh -c apt-get update -qq >/dev/null
```

Рисунок 1 – установка docker и docker-compose

### Установка php v8.1.

По сравнению с Debian для Ubuntu не нужно изобретать велосипед и искать обходные пути, чтобы поставить php v 8.1, прописываем знакомую команду и все устанавливается без проблем:

```
sudo apt -y install php8.1
```

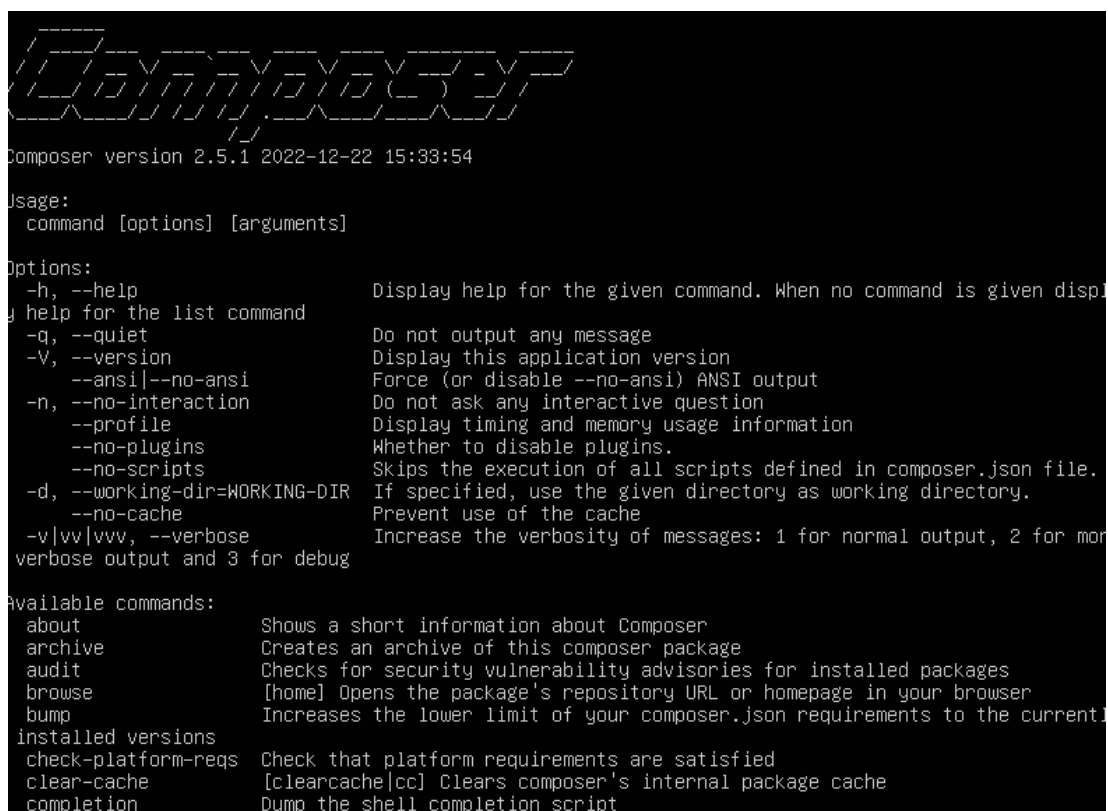
Большинство необходимых пакетов под php будут установлены автоматически, но мы догрузим некоторые пакеты, которые будут необходимы для корректной работы symfony: sqlite, xml, psql, mbstring и т.д.

Кроме того, установим postgres. Напомню, что версия php8.1 необходима для работы symfony demo проекта.

## Установка Composer

Скачиваем файл composer-setup.php с официального сайта первой командой и устанавливаем глобально второй:

```
wget -O composer-setup.php https://getcomposer.org/installer
php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```



```
Composer version 2.5.1 2022-12-22 15:33:54

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list command
  -q, --quiet               Do not output any message
  -V, --version              Display this application version
      --ansi|--no-ansi      Force (or disable --no-ansi) ANSI output
  -n, --no-interaction      Do not ask any interactive question
      --profile             Display timing and memory usage information
      --no-plugins          Whether to disable plugins.
      --no-scripts          Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
      --no-cache            Prevent use of the cache
  -v|vv|vvv, --verbose      Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  about               Shows a short information about Composer
  archive             Creates an archive of this composer package
  audit               Checks for security vulnerability advisories for installed packages
  browse              [home] Opens the package's repository URL or homepage in your browser
  bump                Increases the lower limit of your composer.json requirements to the current installed versions
  check-platform-reqs Check that platform requirements are satisfied
  clear-cache          [clearcache|cc] Clears composer's internal package cache
  completion           Dump the shell completion script
```

Рисунок 1 –результат установки composer.

## Установка symfony:

На официальном сайте находим рекомендации по установке и команду:

```
curl -sS https://get.symfony.com/cli/installer | bash
```

В выводе на рисунке 1 видим рекомендацию для работы symfony глобально – нужно перенести папку symfony в другое расположение, что и сделаем после установки командой:

```
mv /home/mitina_mv/.symfony5/bin/symfony /usr/local/bin/symfony
```

```

mitina_mv@mitinamv:~/common$ curl -sS https://get.symfony.com/cli/installer | bash
Symfony CLI installer

Environment check
[*] cURL is installed
[*] Tar is installed
[*] Git is installed
[*] Your architecture (amd64) is supported

Download
Downloading https://github.com/symfony-cli/symfony-cli/releases/latest/download/symfony-cli_linux_
amd64.tar.gz...
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Dload  % Upload   Total   Spent    Left   Speed
 0     0    0     0    0     0      0     0  --:--:-- --:--:-- --:--:--    0
 0     0    0     0    0     0      0     0  --:--:-- 0:00:01 --:--:--    0
100 5232k 100 5232k    0     0 55047    0  0:01:37 0:01:37 --:--:-- 68290
Uncompress binary...
Installing the binary into your home directory...
The binary was saved to: /home/mitina_mv/.symfony5/bin/symfony

The Symfony CLI was installed successfully!

Use it as a local file:
/home/mitina_mv/.symfony5/bin/symfony

Or add the following line to your shell configuration file:
export PATH="$HOME/.symfony5/bin:$PATH"

Or install it globally on your system:
mv /home/mitina_mv/.symfony5/bin/symfony /usr/local/bin/symfony

Then start a new shell and run 'symfony'

```

Рисунок 2 –установка symfony.

```

cloud:projects      Get a list of all active projects
cloud:tunnel:open   Open SSH tunnels to an app's relationships
cloud:user:add      Add a user to the project
cloud:variables     List variables

Show all commands with symfony help,
Get help for a specific command with symfony help COMMAND.
mitina_mv@mitinamv:~/common$ symfony check:requirements

Symfony Requirements Checker
~~~~~

> PHP is using the following php.ini file:
/etc/php/8.1/cli/php.ini

> Checking Symfony requirements:

.....W.....

[OK]
Your system is ready to run Symfony projects

Optional recommendations to improve your setup
~~~~~

* intl extension should be available
  > Install and enable the intl extension (used for validators).

Note  The command console can use a different php.ini file
~~~~~  than the one used by your web server.
      Please check that both the console and the web server
      are using the same PHP version and configuration.

```

Рисунок 3 –проверка соответствия требованиям symfony для запуска проектов командой `symfony check:requirements`.

## Настройка проекта

### Запуск сервера symfony и проекта demo

Теперь, когда все готово, скачиваем проект с официального репозитория <https://github.com/symfony/demo> - он загрузится в папку с названием demo. Git был установлен в ходе установки Ubuntu, была проведена авторизация по ключу ssh. Выполнить:

```
git clone https://github.com/symfony/demo
```

После того, как все было установлено, переходим в папку с проектом и запускаем командой: `symfony serve` или `symfony server:start`.

```
mitina_mv@mitinamv:~/common/demo$ symfony server:start

[WARNING] run "symfony server:ca:install" first if you want to run the web server with TLS support, or use "--p12" or "--no-tls" to avoid this warning

Following Web Server log file (/home/mitina_mv/.symfony5/log/602d98b792ff41e8a5d6a9fdef48065c306d9d31.log)
Following PHP log file (/home/mitina_mv/.symfony5/log/602d98b792ff41e8a5d6a9fdef48065c306d9d31/7daf403c7589f4927632ed3b6af762a992f09b78.log)
[WARNING] the current dir requires PHP 8.1.0 (composer.json from current dir: /home/mitina_mv/common/demo/composer.json), but this version is not available

[WARNING] The local web server is optimized for local development and MUST never be used in a production setup.

[OK] Web server listening
The Web server is using PHP CLI 8.1.2
http://127.0.0.1:8000

[Web Server ] Jan 17 14:24:48 |DEBUG | PHP   Reloading PHP versions
[Web Server ] Jan 17 14:24:48 |WARN  | PHP   the current dir requires PHP 8.1.0 (composer.json from current dir: /home/mitina_mv/common/demo/composer.json), but this version is not available
[Web Server ] Jan 17 14:24:48 |DEBUG | PHP   Using PHP version 8.1.2 (from default version in $PATH)
[Web Server ] Jan 17 14:24:48 |INFO  | PHP   listening path="/usr/bin/php8.1" php="8.1.2" port=45651
[PHP        ] [Tue Jan 17 14:24:48 2023] PHP 8.1.2-1ubuntu2.9 Development Server (http://127.0.0.1:45651) started
```

Рисунок 4 –успешный запуск проекта.

Пробрасывать порты с виртуальной машины на хостовую мы научились еще в прошлой части, поэтому я покажу конечную настройку по итогу выполнения лабораторной в части проброса портов:



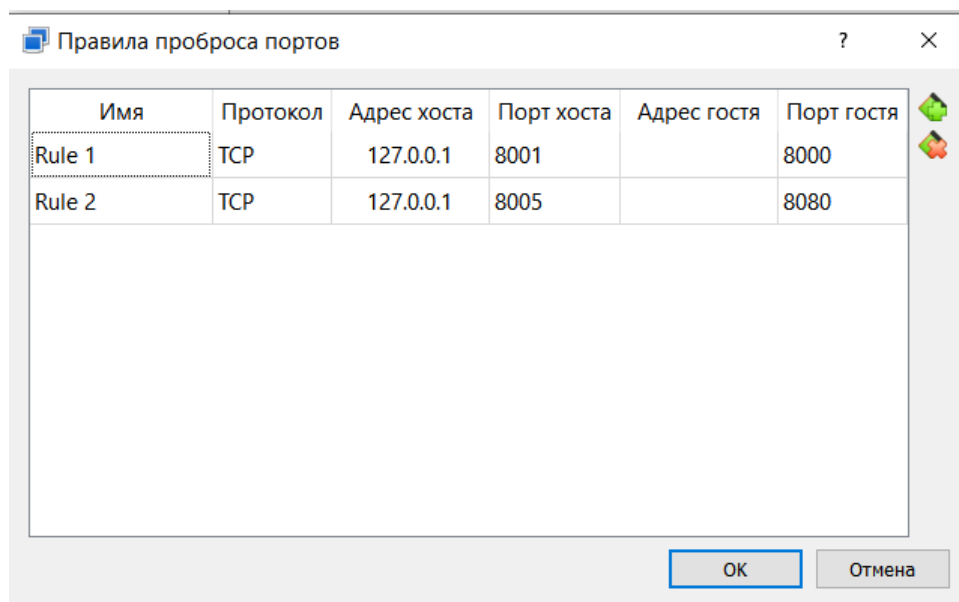


Рисунок 5 –проброс портов с ВМ на хостовую (в итоге выполнения ЛР).

Этот вопрос очень ограничивает при выполнении ЛР: нет возможности указывать домены, порт 80 занят по умолчанию Apache.

В этот раз проект запустился. Конечно, понадобилось установить средствами composer пакеты для symfony. Здесь я покажу несколько фиксов ошибок:

```
[Web Server ] Jan 17 14:32:08 |DEBUG| PHP Reloading PHP versions
[Web Server ] Jan 17 14:32:08 |WARN | PHP the current dir requires PHP 8.1.0 (composer.json from
m current dir: /home/mitina_mv/common/demo/composer.json), but this version is not available
[Web Server ] Jan 17 14:32:08 |DEBUG| PHP Using PHP version 8.1.2 (from default version in $PAT
H)
[Web Server ] Jan 17 14:32:08 |INFO | PHP listening path="/usr/bin/php8.1" php="8.1.2" port=337
13
[PHP ] [Tue Jan 17 14:32:08 2023] PHP 8.1.2-1ubuntu2.9 Development Server (http://127.0.0.1:3
3713) started
[PHP ] [Tue Jan 17 14:37:00 2023] 127.0.0.1:42868 Accepted
[PHP ] [Tue Jan 17 14:37:00 2023] PHP Fatal error: Uncaught Error: Class "App\Kernel" not fo
und in /home/mitina_mv/common/demo/public/index.php:8
[PHP ] Stack trace:
[PHP ] #0 /home/mitina_mv/common/demo/vendor/symfony/runtime/Resolver/DebugClosureResolver.ph
p(25): {closure}()
[PHP ] #1 /home/mitina_mv/common/demo/vendor/autoload_runtime.php(24): Symfony\Component\Runt
ime\Resolver\DebugClosureResolver::Symfony\Component\Runtime\Resolver\{closure}()
[PHP ] #2 /home/mitina_mv/common/demo/public/index.php(5): require_once('...')
[PHP ] #3 {main}
[PHP ] thrown in /home/mitina_mv/common/demo/public/index.php on line 8
[PHP ] [Tue Jan 17 14:37:00 2023] 127.0.0.1:42868 [200]: GET / - Uncaught Error: Class "App\K
ernel" not found in /home/mitina_mv/common/demo/public/index.php:8
[PHP ] Stack trace:
[PHP ] #0 /home/mitina_mv/common/demo/vendor/symfony/runtime/Resolver/DebugClosureResolver.ph
p(25): {closure}()
[PHP ] #1 /home/mitina_mv/common/demo/vendor/autoload_runtime.php(24): Symfony\Component\Runt
ime\Resolver\DebugClosureResolver::Symfony\Component\Runtime\Resolver\{closure}()
[PHP ] #2 /home/mitina_mv/common/demo/public/index.php(5): require_once('...')
[PHP ] #3 {main}
[PHP ] thrown in /home/mitina_mv/common/demo/public/index.php on line 8
[PHP ] [Tue Jan 17 14:37:00 2023] 127.0.0.1:42868 Closing
[Web Server ] Jan 17 14:37:00 |INFO | SERVER GET (200) / ip="10.0.2.2"
[Web Server ] Jan 17 14:37:00 |INFO | SERVER GET (200) /favicon.ico
```

Рисунок 6 –логи сервера – ответ 200, страница отдается

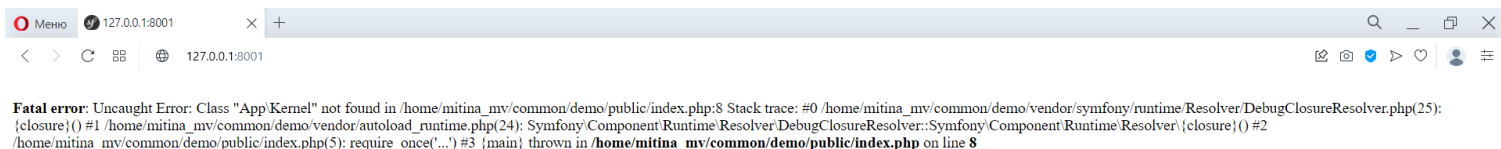


Рисунок 7 –страница symfony

Фикс этой ошибки производился через установку пакета http-kernel:

composer require symfony/http-kernel

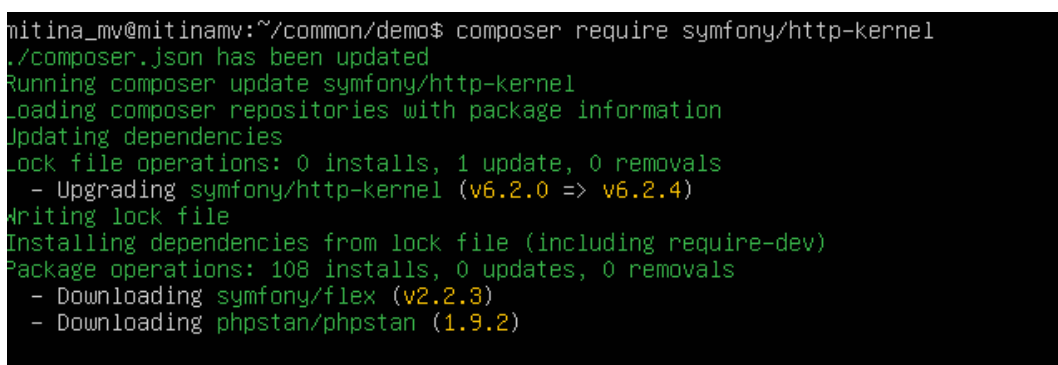


Рисунок 8 –Установка пакета.

После этого я рестартнула сервер и все заработало:

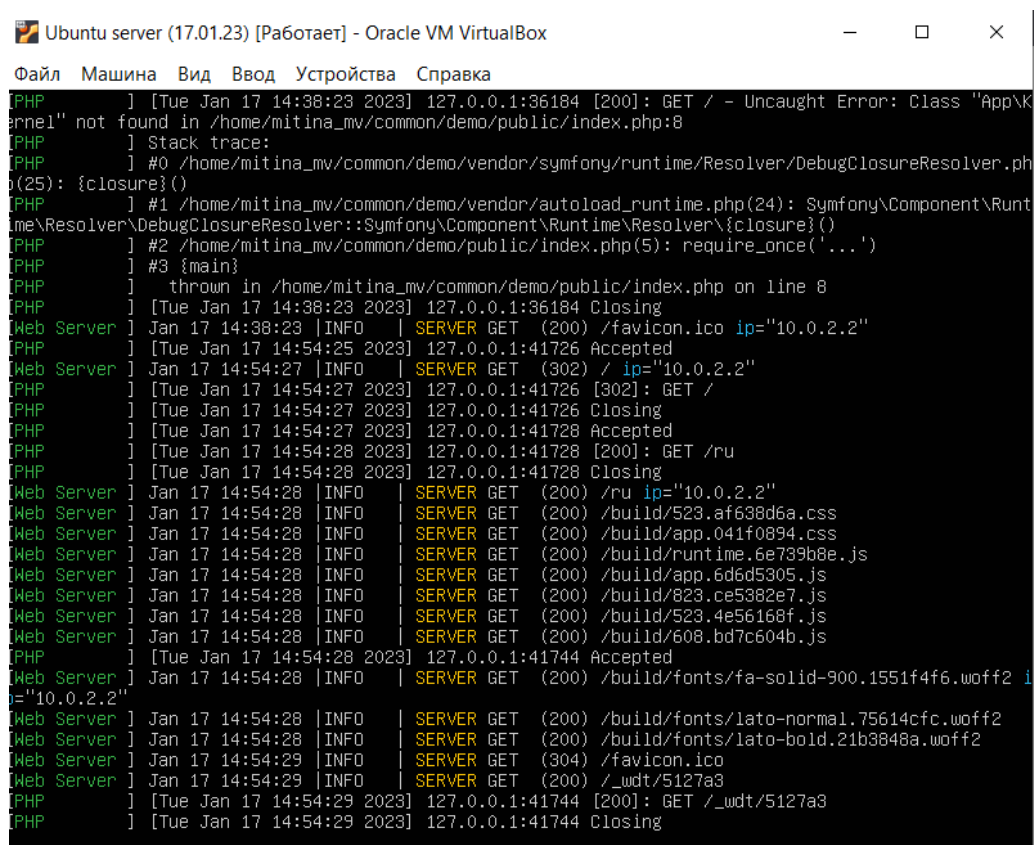


Рисунок 9 –логи сервера после установки пакета http-kernel

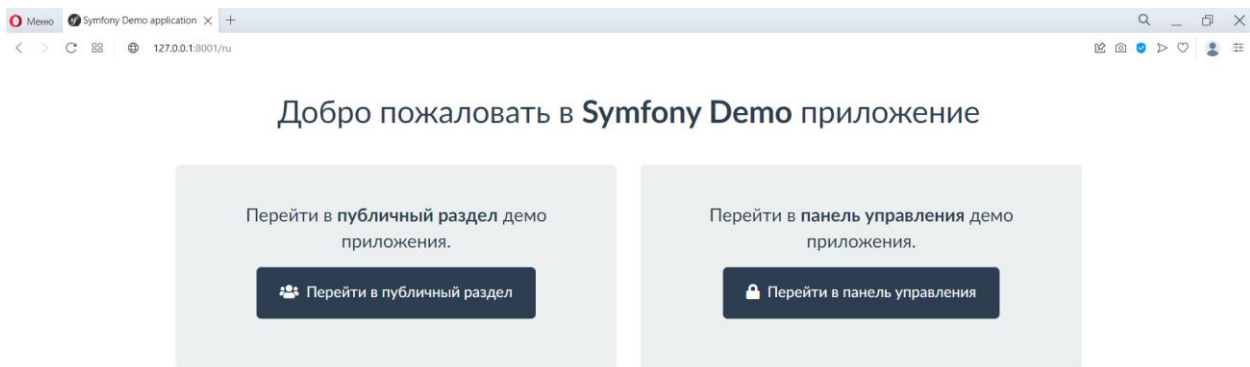


Рисунок 10 – страница проекта demo

По заданию необходимо проект с sqlite перенести на postgres, что мы и сделаем с помощью doctrine, для этого установим пакет orm-pack:

```
mitina_mv@mitinamy:~/common/demo$ composer require symfony/orm-pack
Info from https://repo.packagist.org: #StandWithUkraine
./composer.json has been updated
Running composer update symfony/orm-pack
Loading composer repositories with package information
Restricting packages listed in "symfony/symfony" to "6.2.*"
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking symfony/orm-pack (v2.3.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading symfony/orm-pack (v2.3.0)
  - Installing symfony/orm-pack (v2.3.0): Extracting archive
Generating autoload files
91 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
phpstan/extension-installer: Extensions installed

Run composer recipes at any time to see the status of your Symfony recipes.

Unpacking Symfony packs
  - Unpacked symfony/orm-pack
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 0 updates, 1 removal
  - Removing symfony/orm-pack (v2.3.0)
Generating autoload files
90 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
phpstan/extension-installer: Extensions installed

Run composer recipes at any time to see the status of your Symfony recipes.
```

Рисунок 11 – установка orm-pack

Теперь необходимо изменить файл `.env` в расположении `/demo/`, там нужно указать другое значение переменной окружения. Открываю его через `nano` и правлю:

```

GNU nano 6.2 .env *
In all environments, the following files are loaded if they exist,
the latter taking precedence over the former:

* .env                contains default values for the environment variables needed by the app
* .env.local          uncommitted file with local overrides
* .env.$APP_ENV       committed environment-specific defaults
* .env.$APP_ENV.local uncommitted environment-specific overrides

Real environment variables win over .env files.

DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
https://symfony.com/doc/current/configuration/secrets.html

Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=1.
https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-

##> symfony/framework-bundle ###
PP_ENV=dev
PP_SECRET=2ca64f8d83b9e89f5f19d672841d6bb8
TRUSTED_PROXIES=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
TRUSTED_HOSTS='^(localhost|example\..com)$'
##< symfony/framework-bundle ###

##> doctrine/doctrine-bundle ###
Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/c
IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml

DATABASE_URL=sqlite:///kernel.project_dir%/data/database.sqlite
DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name?serverVersion=5.7",
DATABASE_URL="postgresql://postgres:12345@127.0.0.1:5432/lab_5?serverVersion=13&charset=utf8"
##< doctrine/doctrine-bundle ###

##> symfony/mailer ###

```

Рисунок 12 – установка нового значения для DATABASE\_URL

Теперь создадим базу данных средствами psql. Переходим под пользователя

```

postgres@mitinamv:~$ createdb lab_5
postgres@mitinamv:~$ psql
psql (14.6 (Ubuntu 14.6-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# \l

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
lab_5	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8	
postgres	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8	
template0	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8	=c/postgres + postgres=CTc/postgres
template1	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8	=c/postgres + postgres=CTc/postgres

```

(4 rows)

```

Рисунок 13 – создание БД lab\_5 и просмотр баз данных в системе

Теперь воспользуемся командами для работы с базами данных. Создадим схему и подгрузим фикстуры (чтобы появились записи в БД):

```

php bin/console doctrine:schema:create
php bin/console doctrine:fixtures:load

```

```
! [CAUTION] This operation should not be executed in a production environment!  
  
Creating database schema...  
  
In ExceptionConverter.php line 77:  
  
An exception occurred in the driver: SQLSTATE[08006] [7] connection to server at "127.0.0.1", p  
ort 5432 failed: FATAL: password authentication failed for user "postgres"  
connection to server at "127.0.0.1", port 5432 failed: FATAL: password authentication failed f  
or user "postgres"  
  
In Exception.php line 28:  
  
SQLSTATE[08006] [7] connection to server at "127.0.0.1", port 5432 failed: FATAL: password aut  
hentication failed for user "postgres"  
connection to server at "127.0.0.1", port 5432 failed: FATAL: password authentication failed f  
or user "postgres"  
  
In Driver.php line 29:  
  
SQLSTATE[08006] [7] connection to server at "127.0.0.1", port 5432 failed: FATAL: password aut  
hentication failed for user "postgres"  
connection to server at "127.0.0.1", port 5432 failed: FATAL: password authentication failed f  
or user "postgres"
```

Рисунок 14 – ошибка при создании схемы

Причина ошибки в том, что в .env файле прописано, что у пользователя postgres есть пароль 12345. По умолчанию пароль для пользователя не задан. Но если в .env ничего после : не написать, то будет ошибка «пропущен пароль». Поэтому нужно установить пароль для пользователя postgres:

```
mitina_mv@mitinamv:~/common/demo$ sudo -i -u postgres psql  
psql (14.6 (Ubuntu 14.6-0ubuntu0.22.04.1))  
Type "help" for help.  
  
postgres=# alter role postgres with password '12345'  
postgres=# \password  
Enter new password for user "postgres":  
Enter it again:  
postgres=#  
\\q  
mitina_mv@mitinamv:~/common/demo$ sudo service postgresql restart  
mitina_mv@mitinamv:~/common/demo$ php bin/console doctrine:schema:create  
  
! [CAUTION] This operation should not be executed in a production environment!  
  
Creating database schema...  
  
[OK] Database schema created successfully!  
  
mitina_mv@mitinamv:~/common/demo$ php bin/console doctrine:fixtures:load  
  
Careful, database "lab_5" will be purged. Do you want to continue? (yes/no) [no]:  
> yes  
  
> purging database  
> loading App\DataFixtures\AppFixtures
```

Рисунок 15 – установка пароля и успешная инициализация БД.

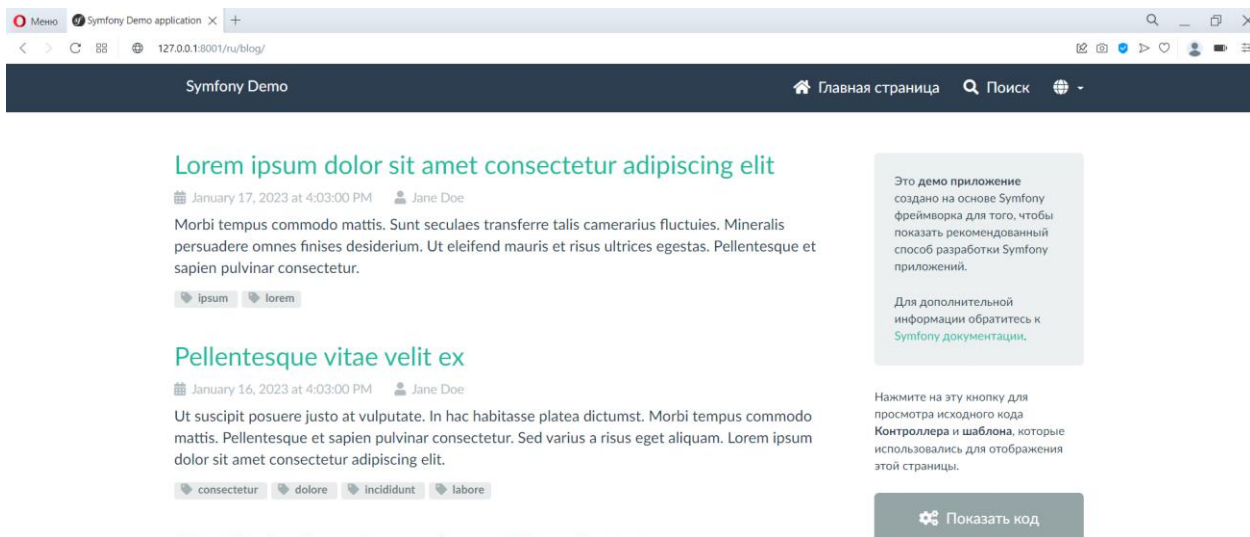


Рисунок 16 – Данные в бд подгрузились, сайт использует postgres

Ура, все работает, я молодец! А почему не работало на Debian, я не знаю ☹

## Настройка docker, сборка проекта

Как уже говорилось ранее, работа над ЛР ведется в общей папке /common/, поэтому все Dockerfile и прочее я писала в удобной VS Code с расширениями, а на виртуальной просто запустила. Код всех файлов приведен в приложении к настоящей работе.

Структура, которая была сделана для проекта:

```
/common/docker
  /logs
    /access.log
    /error.log
  /nginx
    /conf.d
      / default.conf
    /sites
      /default.conf
    /Dockerfile
    /nginx.conf
  /php-fpm
    /Dockerfile
  /postgres
    /Dockerfile
  /docker-compose.yml
  /.env
  /lab_5_backup.sql
```

Для того, чтобы пробросить в наш контейнер базу данных, сделаю ее backup средствами pg\_dump из-под пользователя postgres:

```
mitina_mv@mitinamv:~/common/docker$ sudo -i -u postgres
[sudo] password for mitina_mv:
postgres@mitinamv:~$ pg_dump postgresql://postgres:12345@127.0.0.1:5432/lab_5 > lab_5_backup.sql
postgres@mitinamv:~$ ls
14 lab_5_backup.sql
postgres@mitinamv:~$ pwd
/var/lib/postgresql
```

Рисунок 17 – создание бекапа базы данных

```
# In all environments, the following files are loaded if they exist,
# the latter taking precedence over the former:
#
# * .env contains default values for the environment variables needed by the app
# * .env.local uncommitted file with local overrides
# * .env.$APP_ENV committed environment-specific defaults
# * .env.$APP_ENV.local uncommitted environment-specific overrides
#
# Real environment variables win over .env files.
#
# DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
# https://symfony.com/doc/current/configuration/secrets.html
#
# Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=1.0)
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-configuration

###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=2ca64f8d83b9e89f5f19d672841d6bbb8
#TRUSTED_PROXIES=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
#TRUSTED_HOSTS='^(localhost|example\..com)$'
###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
#DATABASE_URL=sqlite://%kernel.project_dir%/data/database.sqlite
# DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name?serverVersion=5.7",
DATABASE_URL="postgresql://postgres:12345@db:5432/lab_5?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###
```

Рисунок 18 – в .env меняем ip хоста на название контейнера

Теперь все готово и можно запускать сборку командой `docker compose build`:

```
docker-compose.yml nginx php-fpm
mitina_mv@mitinamv:~/common/docker$ docker compose build
WARN[0000] The "APP_ENV" variable is not set. Defaulting to a blank string.
WARN[0000] The "APP_SECRET" variable is not set. Defaulting to a blank string.
WARN[0000] The "DATABASE_URL" variable is not set. Defaulting to a blank string.
[+] Building 9.9s (6/13)
=> [docker-nginx internal] load build definition from Dockerfile 0.3s
=> => transferring dockerfile: 102B 0.1s
=> [docker-nginx internal] load .dockerignore 0.3s
=> => transferring context: 2B 0.1s
=> [docker-php-fpm internal] load build definition from Dockerfile 0.3s
=> => transferring dockerfile: 653B 0.0s
=> [docker-php-fpm internal] load .dockerignore 0.3s
=> => transferring context: 2B 0.0s
=> [docker-nginx internal] load metadata for docker.io/library/nginx:alpine 5.1s
=> [docker-php-fpm internal] load metadata for docker.io/library/php:8.1-fpm 5.1s
=> [docker-nginx 1/2] FROM docker.io/library/nginx:alpine@sha256:659610aadb34b7967dea7686926fddc08d588a71 4.3s
=> => resolve docker.io/library/nginx:alpine@sha256:659610aadb34b7967dea7686926fddc08d588a71 0.1s
=> => sha256:c1b9fe3c0c015486cf1e4a0ecabe78d05864475e279638e9713eb55f013f907 1.78kB / 1.78kB 0.0s
=> => sha256:c433c51bbd66153269da1c592105c9c19bf353e9d7c3d1225ae2bbbeb888c 16.35kB / 16.35kB 0.0s
=> => sha256:d52adec6f48bc3fe2c544a2003a277d91d194b4589bb88d47f4cfa72eb16015d 624B / 624B 3.7s
=> => sha256:659610aadb34b7967dea7686926fddc08d588a71c5121edb094ce0e4cdcbc45e 1.65kB / 1.65kB 0.0s
=> => sha256:8921db27df2831fa6eaa85321205a2470c669b855f3ec95d5a3c2b46de0442c9 0B / 3.37MB 4.1s
=> => sha256:83e90619bc2e4993eafde3a1f5caf5172010f30ba87bbc5af3d06ed5ed93a9e9 0B / 1.80MB 4.1s
=> => sha256:10eb2ce358fad29dd5edbd0d9faa50ff455c915138fdb94ffe9dd88dbe855fbc 0B / 957B 4.1s
=> [docker-php-fpm stage-0 1/4] FROM docker.io/library/php:8.1-fpm@sha256:bdac671e3ca663c0e9dc7008f28b3ee8dc2cc5b2d 4.5s
=> => resolve docker.io/library/php:8.1-fpm@sha256:bdac671e3ca663c0e9dc7008f28b3ee8dc2cc5b2d 0.1s
=> => sha256:92b09bb98bb9ad33d5ea72942b98924117ac190f51e6257754263be91fd5050 2.41kB / 2.41kB 0.0s
=> => sha256:b9e4e67f8b13639b015c0bd0406aae414476416f603a741f0968944106703 11.71kB / 11.71kB 0.0s
=> => sha256:bdac671e3ca663c0e9dc7008f28b3ee8dc2cc5b2d11fcde83d2142ab0369f45 1.86kB / 1.86kB 0.0s
=> [docker-php-fpm] FROM docker.io/library/composer:latest 4.5s
=> => resolve docker.io/library/composer:latest 4.5s
```

Рисунок 19 – запуск сборки.

```

mitina_mv@mitinamv:~/common/docker$ docker compose up -d
[+] Running 14/14
  ⚡ db Pulled
  ⚡ 8740c948ff4d Already exists
  ⚡ c8dbd2beab50 Pull complete
  ⚡ 05d9dc9d0fbd Pull complete
  ⚡ ddd89d5ec714 Pull complete
  ⚡ f98bb9f03867 Pull complete
  ⚡ 0554611e703f Pull complete
  ⚡ 64e0a8694477 Pull complete
  ⚡ 8b868a753f47 Pull complete
  ⚡ 18bfb5f850c Pull complete
  ⚡ 5744def07d7d Pull complete
  ⚡ fe9fb494287c Pull complete
  ⚡ 818b3a4590df Pull complete
  ⚡ a7343ebe57c4 Pull complete
[+] Running 4/4
  ⚡ Network docker_default Created
  ⚡ Container db Started
  ⚡ Container php-fpm Started
  ⚡ Container nginx Started

```

Рисунок 20 – запуск контейнеров

```

Attaching to db, nginx, php-fpm
nginx | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform config
uration
nginx | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.
conf
nginx | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the pa
ckaged version
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx | nginx: [alert] could not open error log file: open() "/var/log/nginx/error.log" failed (2
: No such file or directory)
nginx | 2023/01/18 16:13:04 [emerg] 1#1: invalid parameter "9000" in /etc/nginx/conf.d/default.co
nf:2
nginx exited with code 1

```

Рисунок 21 – Завершение контейнера nginx

```

nginx exited with code 1
db exited with code 1
db | The files belonging to this database system will be owned by user "postgres".
db | This user must also own the server process.
db |
db | The database cluster will be initialized with locale "en_US.utf8".
db | The default database encoding has accordingly been set to "UTF8".
db | The default text search configuration will be set to "english".
db |
db | Data page checksums are disabled.
db |
db | initdb: error: could not access directory "/var/lib/postgresql/data": Permission denied
db exited with code 1

```

Рисунок 22 – Завершение контейнера db

В то же время в браузере после моего запуска страница была недоступна, потому что nginx валился и не перезапускался. Действительно, в docker-compose.yml указано, что контейнер с базой данных перезапускаться всегда, а для nginx такого не приписано (чтобы если были какие-то ошибки, сайт просто был недоступен).



Посмотрим с помощью команды `docker ps`, какие у нас контейнеры сейчас работают:

```
mitina_mv@mitinamv:~/common/docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
aff680833e19	docker-php-fpm	"docker-php-entrypoi..."	2 minutes ago	Up 2 minutes
9000/tcp	php-fpm			
a4f11c7e3a67	postgres:12	"docker-entrypoint.s..."	2 minutes ago	Restarting (1) 1 second ago
	db			

Рисунок 23 – вывод команды `docker ps`

Вижу, что контейнера `nginx` нет, а контейнер `db` живет уже 2 минуты, но в статусе прописано, что он постоянно пытается перезагрузиться, возникает ошибка и он падает, но докер снова его поднимает. Для начала починим ошибку `db`. Для фикса этого нашла совет использовать том и прописать в `docker-compose` файле следующее:

```
volumes:
  pgdata:
    driver: local
```

Пересобрали `docker-compose` и запустили, ошибка с `db` пропала, но `nginx` все еще мертвый:

```
Container nginx Created 0.0s
Attaching to db, nginx, php-fpm
db
db | PostgreSQL Database directory appears to contain a database; Skipping initialization
db
db | 2023-01-18 17:39:17.865 UTC [1] LOG: starting PostgreSQL 14.6 (Debian 14.6-1.pgdg110+1)
on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
db | 2023-01-18 17:39:17.866 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db | 2023-01-18 17:39:17.866 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db | 2023-01-18 17:39:17.877 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PG
SQL.5432"
db | 2023-01-18 17:39:17.893 UTC [25] LOG: database system was shut down at 2023-01-18 17:39:
04 UTC
db | 2023-01-18 17:39:17.908 UTC [1] LOG: database system is ready to accept connections
php-fpm | Installing dependencies from lock file (including require-dev)
php-fpm | Verifying lock file contents can be installed on current platform.
nginx | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform config
uration
nginx | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.
conf
nginx | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the pa
ckaged version
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx | nginx: [alert] could not open error log file: open() "/var/log/nginx/error.log" failed (2
: No such file or directory)
nginx | 2023/01/18 17:39:19 [emerg] 1#1: unknown "url" variable
php-fpm | Nothing to install, update or remove
php-fpm | Generating optimized autoload files
nginx exited with code 1
```

Рисунок 24 – запуск `docker compose`

Если сравнить рисунок 22 и рисунок 25, увидим, что nginx теперь ругается на другое. Как я исправила ошибку с рисунка 22 я уже не помню, но дело оказалось не в 9000. На рисунке 25 видим, что он не знает переменную. Действительно, в конфиге я описалась – меняем переменную url на uri и делаем docker compose restart. В этот раз изменены были не dockerfile, поэтому снова билдить ничего не надо, но нужно перезагрузить nginx. И тадам:

```
on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
db      | 2023-01-18 17:46:37.246 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db      | 2023-01-18 17:46:37.246 UTC [1] LOG:  listening on IPv6 address ":::", port 5432
db      | 2023-01-18 17:46:37.259 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PG
SQL.5432"
db      | 2023-01-18 17:46:37.281 UTC [25] LOG:  database system was shut down at 2023-01-18 17:46:
34 UTC
db      | 2023-01-18 17:46:37.290 UTC [1] LOG:  database system is ready to accept connections
php-fpm | Installing dependencies from lock file (including require-dev)
php-fpm | Verifying lock file contents can be installed on current platform.
nginx   | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform config
uration
nginx   | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx   | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx   | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.
conf
nginx   | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the pa
ckaged version
nginx   | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx   | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx   | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx   | nginx: [alert] could not open error log file: open() "/var/log/nginx/error.log" failed (2
: No such file or directory)
php-fpm | Nothing to install, update or remove
php-fpm | Generating optimized autoload files
php-fpm | 90 packages you are using are looking for funding.
php-fpm | Use the `composer fund` command to find out more!
php-fpm | phpstan/extension-installer: Extensions installed
php-fpm | Run composer recipes at any time to see the status of your Symfony recipes.
php-fpm | Executing script cache:clear [OK]
php-fpm | Executing script assets:install public [OK]
php-fpm |
php-fpm | [18-Jan-2023 17:47:49] NOTICE: fpm is running, pid 27
php-fpm | [18-Jan-2023 17:47:49] NOTICE: ready to handle connections
```

Рисунок 25 – успешный запуск контейнера

Да, там есть ошибка открытия файла, но она не критичная, я еще не стала исправлять потому что в браузере увидела следующее:



Рисунок 26 – ошибка 404 на сайте

Пыталась пофиксить это два дня, но что-то не получилось. Скорее всего, что-то с папками не так, nginx их не видит или не имеет прав из доступа. В то же время nginx без проблем писал ошибки в логи, поэтому я не знаю, как это исправить. Если знаете, пишите в telegram - @mitina\_mv.

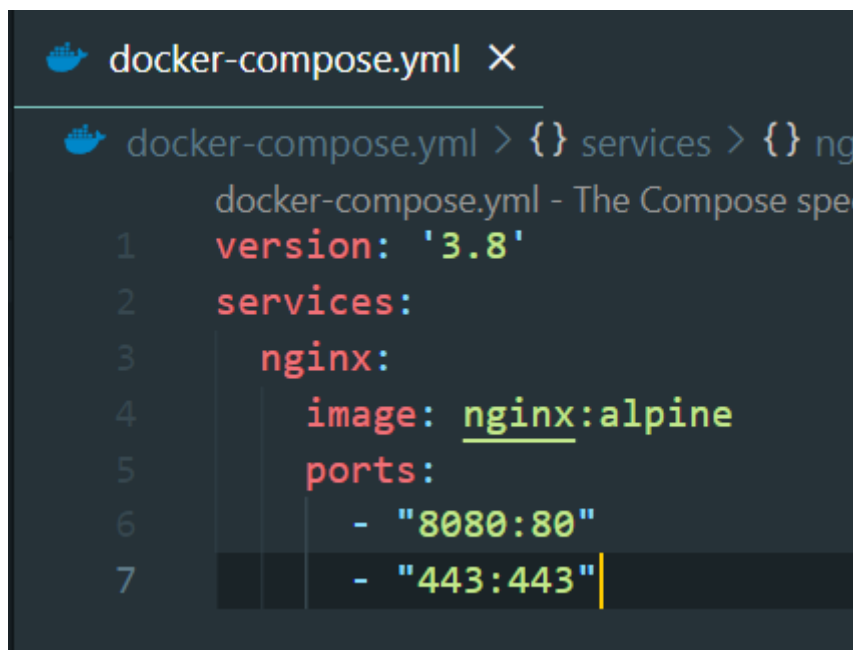
Ошибка из логов:

```
2023/01/18    19:18:06    [crit]    32#32:    *1    stat()
"/var/www/public/index.php" failed (13: Permission denied), client:
10.0.2.2, server: localhost, request: "GET / HTTP/1.1", host:
"127.0.0.1:8005"
```

По ошибке делаю вывод, что все пути найдены, все файлы имеются, но контейнер не имеет доступа. Так как главной задачей было запустить мультиконтейнерное приложение через docker, что и удалось реализовать – считаю задачу выполненной и перехожу к следующему пункту.

## Задание 2

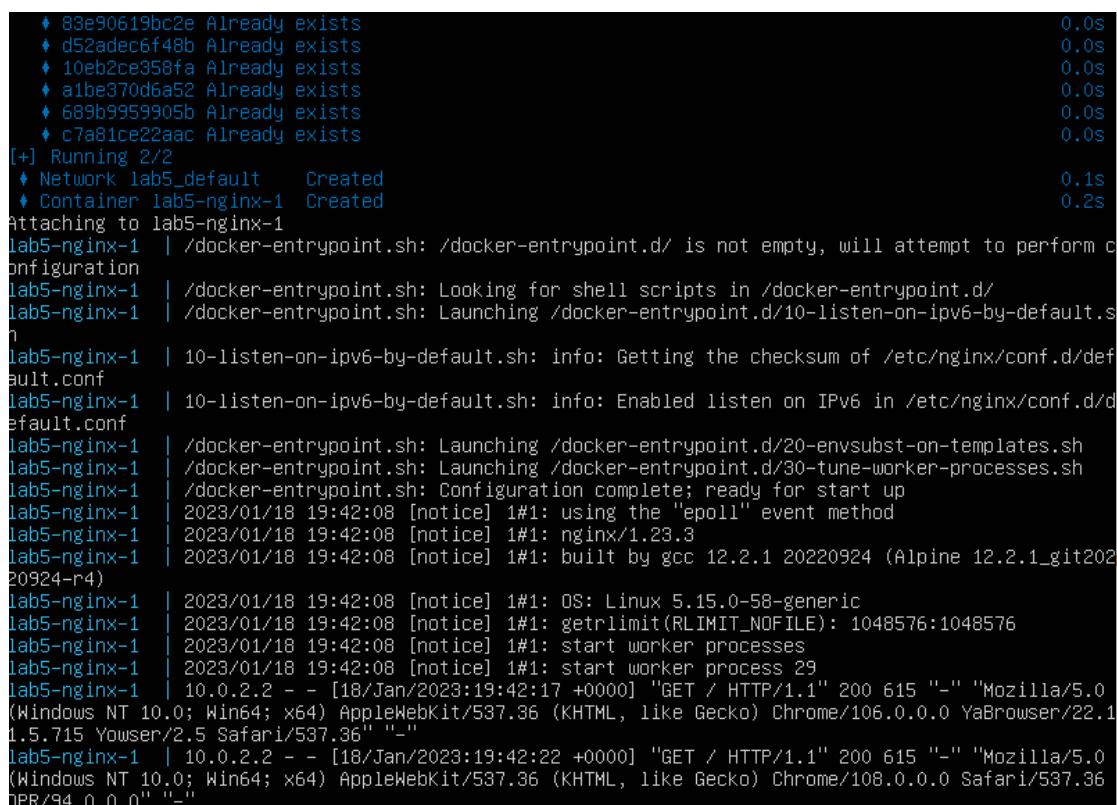
**Шаг 1.** Создаем docker-compose файл следующего содержания:



```
docker-compose.yml X

docker-compose.yml > {} services > {} ng
docker-compose.yml - The Compose spec
1  version: '3.8'
2  services:
3    nginx:
4      image: nginx:alpine
5      ports:
6        - "8080:80"
7        - "443:443"
```

Рисунок 27 – docker-compose.yml



```

  83e90619bc2e Already exists          0.0s
  d52adec6f48b Already exists          0.0s
  10eb2ce358fa Already exists          0.0s
  a1be370d6a52 Already exists          0.0s
  689b9959905b Already exists          0.0s
  c7a81ce22aac Already exists          0.0s
[+] Running 2/2
  Network lab5_default Created          0.1s
  Container lab5-nginx-1 Created        0.2s
Attaching to lab5-nginx-1
lab5-nginx-1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform c
onfiguration
lab5-nginx-1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
lab5-nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
lab5-nginx-1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
lab5-nginx-1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
lab5-nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
lab5-nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
lab5-nginx-1 | /docker-entrypoint.sh: Configuration complete; ready for start up
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: using the "epoll" event method
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: nginx/1.23.3
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git202
20924-r4)
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: OS: Linux 5.15.0-58-generic
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: start worker processes
lab5-nginx-1 | 2023/01/18 19:42:08 [notice] 1#1: start worker process 29
lab5-nginx-1 | 10.0.2.2 - - [18/Jan/2023:19:42:17 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 YaBrowser/22.1
1.5.715 Yowser/2.5 Safari/537.36" "-"
lab5-nginx-1 | 10.0.2.2 - - [18/Jan/2023:19:42:22 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
OPR/94.0.0.0" "-"
```

Рисунок 28 – docker compose up --build



Рисунок 29 – приветственная страница nginx

**Шаг 2.** Добавляем в docker-compose в контейнер nginx volume для указания папок:

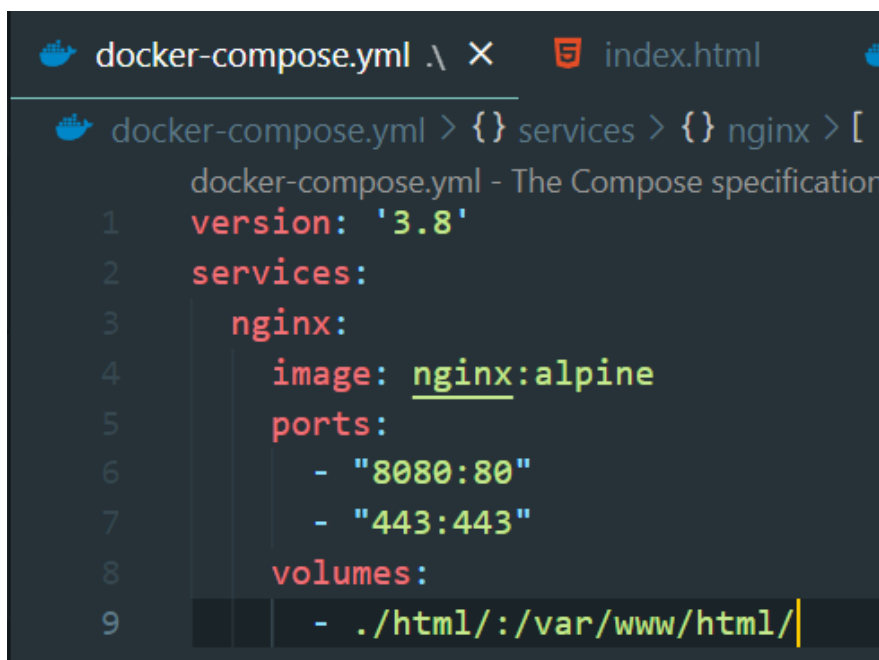


Рисунок 30 – docker-compose.yml - шаг 2

К сожалению, в браузере вижу все еще приветствие nginx. Добавляю nginx.conf, там указываю root на /var/www/html и теперь вижу в браузере:

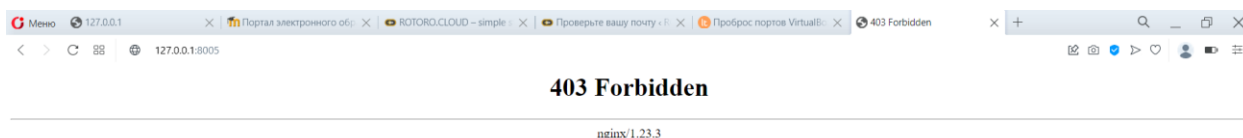


Рисунок 31 – Ошибка 403 в браузере

Опять этот nginx... Ну что ж за беда?

Смотрим `docker inspect <id контейнера>`

```
"Mounts": [
  {
    "Type": "bind",
    "Source": "/home/mitina_mv/common/lab5/docker/nginx/conf.d/default.nginx",
    "Destination": "/etc/nginx/conf.d/default.conf",
    "Mode": "rw",
    "RW": true,
    "Propagation": "rprivate"
  },
  {
    "Type": "bind",
    "Source": "/home/mitina_mv/common/lab5/html",
    "Destination": "/var/www/html",
    "Mode": "",
    "RW": false,
    "Propagation": "rprivate"
  }
],
```

Рисунок 32 – inspect контейнера в части Mounts

Я подумала, что, возможно, нет прав, или папка не монтируется, поэтому изменила 9 строчку на:

`./html:/var/www/html:rm` – даем права на чтение-запись в папку

Соответственно, режим изменился, но ошибка сохранилась. Победить не получилось.

**Шаг 3.** По указаниям я создала сеть `proxy_proxy` и все `docker-compose` файлы. Посмотрим на доступные сети с помощью команды `docker network ls`:

```
mitina_mv@mitinamv:~/common/lab5/proxy$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b6db267a5c06        bridge              bridge              local
bcfa0a5534e3        docker_default      bridge              local
be92bdf8b528        host                host                local
6c2ea8002cf0        lab5_default        bridge              local
e8086a94dd13        none                null                local
7de08efeea1f        proxy_proxy         bridge              local
```

Рисунок 33 – сети docker

Здесь вступили ограничения проброса портов с ВМ на хостовую машину – нельзя указать домен в настройках. Поэтому сеть `proxy` осталась невостребованной. Я удалила эту часть из основного `docker-compose` и запустила. Но `nginx` опять показал ошибку 403 ☹

#### Шаг 4. Создам docker-compose.yml для приложения с mysql и wordpress.

Код представлен в приложении. Запускаем:

```
mitina_mv@mitinamv:~/common/lab5$ docker compose up -d --build
[+] Running 4/34
* db Pulling
  * d26998a7c52d Pulling fs layer
  * 4a9d8a3567e3 Download complete
  * bfee1f0f349e Downloading 245.1kB/930.2kB
  * 71ff8dfb9b12 Waiting
  * bf56cbebc916 Waiting
  * 9a7bdfff0f5b Waiting
  * b7677d20b791 Waiting
  * b4bcfc1167c2 Waiting
  * 03f1c0b71b70 Waiting
  * 61aa2e026ac3 Waiting
  * 3b784fa66c50 Waiting
* wordpress Pulling
  * 8740c948ffd4 Already exists
  * 1873be858264 Already exists
  * 7ce6a163d8c1 Already exists
  * 008a172010ba Already exists
  * d15353ae3d77 Waiting
  * 223eb1888c0f Waiting
  * 83374c2a967a Waiting
  * 8adb6fee4c96 Waiting
  * 04e8dd13d367 Waiting
  * 08c657b572e7 Waiting
  * 4e2a69062e74 Waiting
  * f340310b8889 Waiting
  * c2839f599e98 Waiting
  * 07cf3f6c92fa Waiting
  * a61869359229 Waiting
  * a84ad5ffd8f9 Waiting
  * c7fdd14fc94d Waiting
  * f7d19cc4ffaa Waiting
  * 1b81deb9db45 Waiting
  * db8f5037e095 Waiting
  * 34b5166230af Waiting
```

Рисунок 34 – сборка из docker-compose.yml

И внезапно (после неудач с nginx неожиданно даже для меня) оно заработало как нужно:

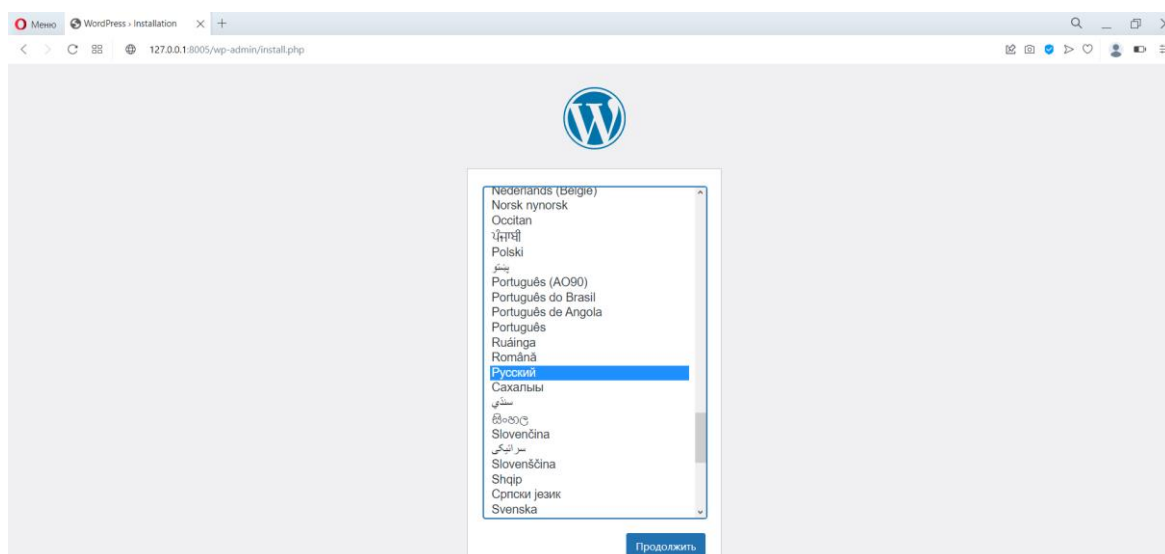


Рисунок 35 – Первая страница wordpress

Так как в сборке не было nginx, и оно заработало, могу предположить, что ошибка была все-таки не в пробросе папок или портов, а в настройках конфигов сервера. Так как я не имею навыков в настройке nginx, починить их так и не получилось. Покажу еще несколько скринов из настройки:

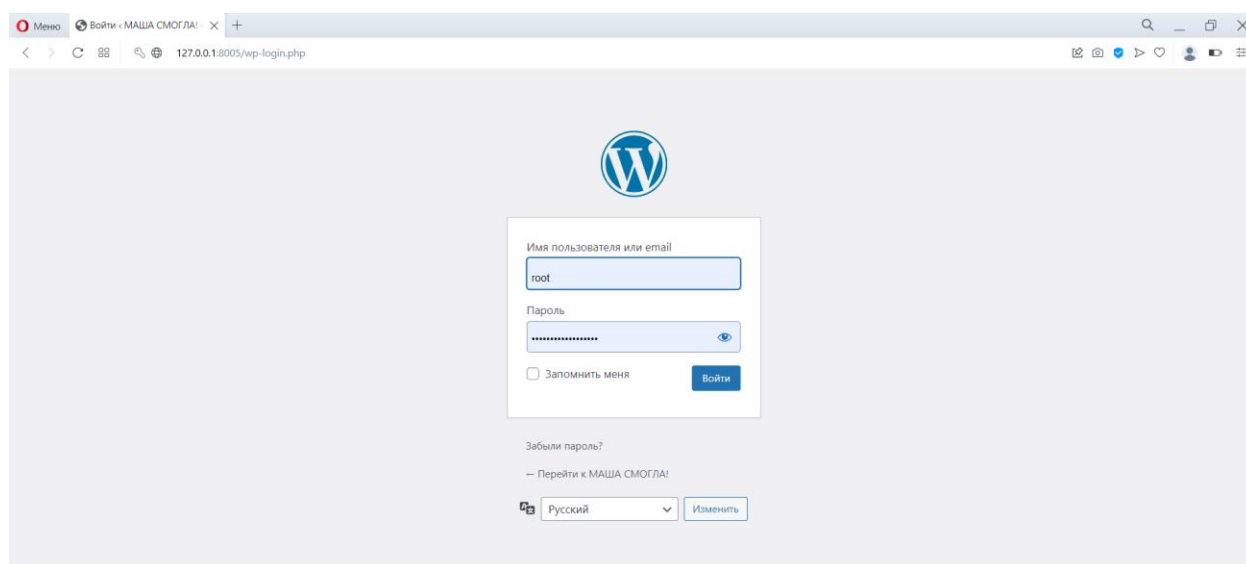


Рисунок 36 – Заходим в личный кабинет админа сайта МАША СМОГЛА

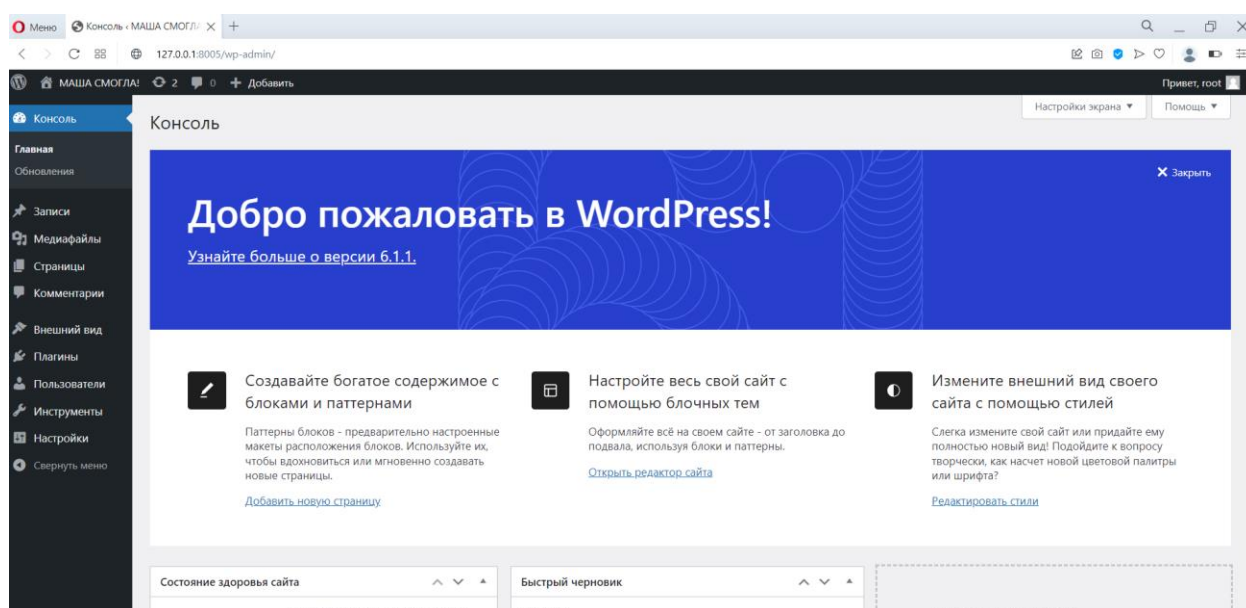


Рисунок 37 – Админка сайта МАША СМОГЛА



## **Вывод**

В ходе выполнения ЛР на Ubuntu удалось выполнить большую часть задач. В настройке nginx не все получилось, но в то же время ошибки в рамках docker были решены и все контейнеры работали. Я так и не поняла, почему на Debian не удалось выполнить лабораторную, так как она не сильно отличается от Ubuntu, возможно, в процессе установки Ubuntu были поставлены автоматически какие-то важные пакеты и это сыграло свою роль. В процессе решения заданий я приводила описание возникающих ошибок и показала методы их исправления.

## Контрольные вопросы

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.

А. Меньшие накладные расходы на инфраструктуру

С. Невозможность запуска GNU/Linux- и Windows-приложений на одном хосте – контейнеризация предполагает, что виртуальная среда запускается на основе ядра ОС хостовой машины (что снижает потребление ресурсов), а виртуализация – это когда с помощью гипервизора мы можем установить любую ОС на любую ОС хостовой машины.

2. Назовите основные компоненты Docker.

В. Контейнеры

Д. Реестры – так называются публичные и приватные хранилища докер образов, например, Docker Hub.

3. Какие технологии используются для работы с контейнерами?

А. Пространства имен (Linux Namespaces) – изоляция в виртуализации обеспечивается через ограничения со стороны процессора, а в контейнеризации это делается через пространство имен: при создании контейнера докер создает набор пространства имен для данного контейнера. PID рабочего процесса внутри контейнера начинаются с 1, в то же время на хостовой машине тоже есть PID 1. В реальности все процессы контролируются хостовой машиной, поэтому внутренние PID процессов в контейнере сопоставляются с PID хоста при помощи pid-namespace.

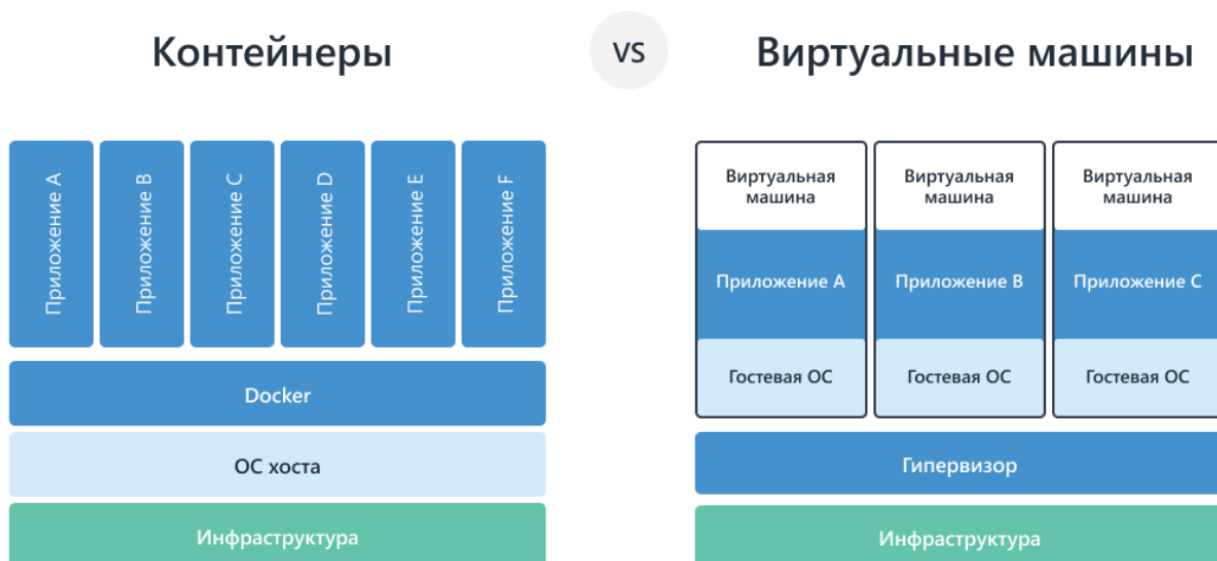
С. Контрольные группы (cgroups) – позволяют разделить доступные ресурсы железа и создавать ограничения при необходимости.

4. Найдите соответствие между компонентом и его описанием:

- образы – доступные только для чтения шаблоны приложений;
- контейнеры – изолированные при помощи технологий операционной системы пользовательские окружения, в которых выполняются приложения;
- реестры (репозитории) – сетевые хранилища образов.

5. В чем отличие контейнеров от виртуализации?

В первую очередь отличия в организации работы. Виртуализация позволяет создать полноценную операционную систему, с собственным ядром и другими изолированными ресурсами. Действительно, даже при настройке VM в Virtual Box мы указывали размер диска и другие характеристики. Это все потому, что VM больше указанного от хоста не возьмет. Контейнеризация же позволяет нам создавать изолированные системы не на аппаратном уровне, а на базе операционной системы хостовой машины. Это позволяет Docker распределять ресурсы между контейнерами, кроме того, мы можем даже управлять уровнем изоляции контейнеров. Так как используется ядро хоста, нам нужны только те средства, которые нужны контейнеру, поэтому развёртывание и запуск осуществляются быстрее. При этом такой подход накладывает ограничения на запуск приложений – у нас нет возможностей запустить на хосте с ОС Linux приложения под винду и наоборот.



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

- `run <название образа>`— запуск контейнера из образа
  - `-it` — позволяет войти в контейнер, чтобы работать изнутри.  
Например, в контейнер с Ubuntu мы может работать на `bash`.
  - `--name` — позволяет задать имя контейнеру
  - `-v` или `--mount` — указывает для контейнера папки монтирования
  - `-p` — позволяет пробросить порты из контейнера на хост
  - `-d` — запуск контейнера в фоновом режиме
  - `-e` — устанавливает переменную окружения
  - `--link` — позволяет связать контейнер с другим
- `pull <название образа>`— скачивание образа из реестра
- `stop <id/name>` - остановка контейнера
- `ps` — просмотр списка работающих контейнеров
  - `-a` — весь список контейнеров (запущенных и нет)
- `rm <id/name>` - удаляет остановленный или жестко завершает контейнер
- `images` — просмотр списка образов на хосте
- `rmi` — удаляет образ (перед удалением обязательно остановить все контейнеры этого образа)
- `exec <контейнер> <команда>` - позволяет выполнить команду на работающем контейнере
- `build` — сборка образа на основе файлов
- `push` — позволяет отправить образ в реестр (нужны имя и адрес)
- `inspect` — просмотр полной информации о контейнере
- `network ls` — просмотр сетей на хосте
- `compose up` — запуск мультиконтейнерного приложения
  - `-d` — запуск в фоновом режиме
  - `--scale <name>=<num>` - создает кол-во реплик контейнера

- `compose build` – сборка мультиконтейнерного приложения на основе файла `docker-compose.yml`

## 7. Каким образом осуществляется поиск образов контейнеров?

Сначала докер ищет указанный образ на хосте, и если его не находит, то идет на доступные реестры и скачивает их оттуда. Реестр может быть частным, или запрашиваемый образ может быть закрытым – для доступа к таким образам нужно авторизоваться – команда `docker login`.

## 8. Каким образом осуществляется запуск контейнера?

Контейнер запускается на основе образа. Образ может быть скачен из реестра или создан на хосте. Созданные на хосте образы обычно включает в себя инструкцию `FROM`, указывающую на основе какого образа осуществляется создание текущего. В образе указаны все необходимые зависимости и другие инструкции. При запуске контейнера из образа на хосте в первый раз его сначала нужно собрать командой `docker build`. Сборка осуществляется по слоям, каждый слой имеет свой размер. Если осуществляется пересборка образа, то идентичные инструкции будут взяты из кеша.

## 9. Что значит управлять состоянием контейнеров?

Контейнер – это экземпляр образа. По мере выполнения цели контейнера может изменяться его поведение – т.е. состояние контейнера. Посмотреть состояния можно, например, через `docker ps`, в графе статус будет указан текущее состояние контейнера. Состояния у контейнеров бывают следующие: создан, запущен, перезапущен, вышедший, приостановлен, мертвый. Большую часть из них я показала в ходе выполнения ЛР.

## 10. Как изолировать контейнер?

Контейнеры изолированы по умолчанию, но есть возможность отключить все механизмы изоляции Docker – в таком случае запуск

приложения в контейнере не будет отличаться от запуска на хосте. Две основы изоляции – это Linux namespace и cgroup.

#### 11. Опишите последовательность создания новых образов, назначение Dockerfile?

Dockerfile – это текстовый файл, содержащий набор инструкций по сборке докер-образа.

Первым шагом в создании докер образа является выбор родительского образа, на основе которого будет создан наш. Все слои данных нового образа будут наложены поверх родительского. Инструкция FROM позволяет указать родительский образ. Далее можем прописать, например, RUN – инструкция выполнит указанные команды. Здесь можно прописать установку необходимых пакетов через apt-get install или создать папки – и многое другое. Инструкция COPY позволит скопировать что-то куда-то. CMD – установит команду по умолчанию при старте образа. И многое другое.

После того, как докер файл будет создан, начинается его сборка. Сборка состоит из слоев, где последовательно выполняются все инструкции dockerfile. Каждая инструкция – это новый слой образа.

#### 12. Возможно ли работать с контейнерами Docker без одноименного движка?

Да. Например, облачные сервисы: Fly.io, Stackpath, Deno.land, Vercel.app.

#### 13. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes?

Система оркестрации контейнеров (не обязательно Kubernetes) нужны для управления контейнерами. Она позволяет их создавать (планировать время создания тоже), масштабировать, распределять между несколькими хостами, выделять ресурсы, следить за статусами контейнеров, перезапускать и многое другое.

Kubernetes – одна из систем оркестрации, но очень хорошая. Kubernetes не зависит от языка программирования, платформы и операционной системы, он предлагает широкий спектр вариантов развертывания. Кроме того, он предоставляет множество разных удобных функций, которых нет, например, в Docker Swarm – бесплатной встроенной в докер системе оркестрации.

Основные объекты Kubernetes:

- Кластеры: пул для вычислений, хранения и сетевых ресурсов.
- Ноды: хост-машины, работающие в кластере.
- Пространства имен: логические разделы кластера.
- Поды: единицы развертывания.
- Метки и селекторы: пары «ключ-значение» для идентификации и обнаружения сервисов.
- Сервисы: коллекция подов, принадлежащих одному и тому же приложению.
- Набор реплик: обеспечивает доступность и масштабируемость.
- Развертывание: управляет жизненным циклом приложения.

## Приложения

### /docker/docker-compose.yml

```
version: '3.8'
services:
  db:
    container_name: db
    build:
      context: ./postgres
    restart: always
    env_file:
      - .env
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: 12345
      POSTGRES_DB: lab_5
      PGDATA: /var/lib/postgresql/data
    ports:
      - 15432:5432
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./lab_5_backup.sql:/docker-entrypoint-initdb.d/lab_5_backup.sql
  php-fpm:
    container_name: php-fpm
    build:
      context: ./php-fpm
    depends_on:
      - db
    environment:
      - APP_ENV=${APP_ENV}
      - APP_SECRET=${APP_SECRET}
      - DATABASE_URL=${DATABASE_URL}
    volumes:
      - ../../demo:/var/www
  nginx:
    container_name: nginx
    build:
      context: ./nginx
    volumes:
      - ../../demo:/var/www
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/sites:/etc/nginx/sites-available
      - ./nginx/conf.d:/etc/nginx/conf.d
```



```
    - ./logs:/var/log
  depends_on:
    - php-fpm
  ports:
    - 8080:80
    - 443:443
volumes:
  pgdata:
    driver: local
```

### **/docker/nginx/Dockerfile**

```
FROM nginx:alpine
WORKDIR /var/www
CMD ["nginx"]
EXPOSE 80 443
```

### **/docker/nginx/nginx.conf**

```
user nginx;
worker_processes 4;
daemon off;

error_log /var/log/error.log warn;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    access_log /var/log/access.log;
    sendfile on;
    keepalive_timeout 65;
    gzip on;
    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-available/*.conf;
}
```

## **/docker/nginx/conf.d/default.conf**

```
upstream php-upstream {  
    server php-fpm:9000;  
}
```

## **/docker/nginx/sites/default.conf**

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server ipv6only=on;  
  
    server_name localhost;  
    root /var/www/public;  
    index index.php index.html index.htm;  
  
    location / {  
        try_files $uri $uri/ /index.php$is_args$args;  
    }  
  
    location ~ \.php {  
        try_files $uri /index.php =404;  
        fastcgi_pass php-upstream;  
        fastcgi_index index.php;  
        fastcgi_buffers 16 16k;  
        fastcgi_buffer_size 32k;  
        fastcgi_param SCRIPT_FILENAME  
$document_root$fastcgi_script_name;  
        fastcgi_read_timeout 600;  
        include fastcgi_params;  
    }  
  
    location ~ /\.ht {  
        deny all;  
    }  
  
    location /.well-known/acme-challenge/ {  
        root /var/www/letsencrypt/;  
        log_not_found off;  
    }  
}
```

## **/docker/postgres/Dockerfile**

```
FROM postgres:14
RUN mkdir -p "$PGDATA" && chmod 700 "$PGDATA"
```

## **/docker/php-fpm/Dockerfile**

```
FROM php:8.1-fpm

RUN apt-get update -y && \
    apt-get install -y --no-install-recommends libssl-dev zlib1g-dev
&& \
    apt install -y git unzip netcat libxml2-dev libpq-dev libzip-dev
&& \
    pecl install apcu && \
    docker-php-ext-configure pgsql -with-pgsql=/usr/local/pgsql && \
    docker-php-ext-install -j$(nproc) zip opcache intl pdo_pgsql pgsql
&& \
    docker-php-ext-enable apcu pdo_pgsql sodium && \
    apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var tmp/*
COPY --from=composer /usr/bin/composer /usr/bin/composer
WORKDIR /var/www
CMD composer i -o ; php-fpm
EXPOSE 9000
```

## **/lab5/docker-compose.yml (для части 2 с wordpress)**

```
version: "3.8"
services:
  wordpress:
    image: wordpress:latest
    restart: always
    links:
      - db:mysql
    ports:
      - "8080:80"
    working_dir: /var/www/html
    volumes:
      - "/opt/wp-content:/var/www/html/wp-content"
    environment :
      WORDPRESS_DB_HOST: "db:3306"
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: 12345
      WORDPRESS_DB_NAME: wp_lab5
  db:
    image: mysql:5.7
    restart: always
    volumes:
      - "/opt/mysql:/var/lib/mysql"
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: wp_lab5
      MYSQL_USER: root
      MYSQL_PASSWORD: 12345
```