

# Липецкий государственный технический университет

Факультет автоматизации и информатики  
Кафедра прикладной математики

Отчет по лабораторной работе № 5  
по дисциплине «Операционная система Linux»  
Тема «Контейнеризация»

Студент

\_\_\_\_\_  
подпись, дата

Егорова М.Р.  
фамилия, инициалы

Группа ПМ-20-2

Руководитель  
учёная степень, учёное звание

\_\_\_\_\_  
подпись, дата

Кургасов В.В.  
фамилия, инициалы

Липецк 2022 г.

## Содержание

1. Цель работы	3
2. Задание кафедры	4
3. Ход выполнения работы	5
4. Контрольные вопросы	22

# 1. Цель работы

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

## 2. 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> /ссылка на github/).
2. По умолчанию проект работает с `sqlite`-базой. Нужно заменить ее на `postgres`. 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.
3. 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



```

root@masha:/home/masha# sudo apt -y install php8.1-sqlite3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Следующие НОВЫЕ пакеты будут установлены:
  php8.1-sqlite3
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 61 пакетов не о
бновлено.
Необходимо скачать 32,3 кВ архивов.
После данной операции объём занятого дискового пространства возрастёт на 139 кВ.
0% [Обработка]_

```

Рис. 3 – Устанавливаем библиотеки php

```

root@masha:/home/masha# sudo apt -y install postgresql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Будут установлены следующие дополнительные пакеты:
  libcommon-sense-perl libjson-perl libjson-xs-perl libl1vm14 libpq5 libsensors-config libsensors5
  libtypes-serialiser-perl postgresql-14 postgresql-client-14 postgresql-client-common
  postgresql-common sysstat
Предлагаемые пакеты:
  lm-sensors postgresql-doc postgresql-doc-14 isag
Следующие НОВЫЕ пакеты будут установлены:
  libcommon-sense-perl libjson-perl libjson-xs-perl libl1vm14 libpq5 libsensors-config libsensors5
  libtypes-serialiser-perl postgresql postgresql-14 postgresql-client-14 postgresql-client-common
  postgresql-common sysstat
Обновлено 0 пакетов, установлено 14 новых пакетов, для удаления отмечено 0 пакетов, и 61 пакетов не
обновлено.
Необходимо скачать 42,4 МВ архивов.
После данной операции объём занятого дискового пространства возрастёт на 161 МВ.
0% [Обработка]

```

Рис. 4 – Устанавливаем postgresql

```

> Checking Symfony requirements:
.....

[OK]
Your system is ready to run Symfony projects

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.

```

Рис. 5 – Проверяем, что symfony стоит глобально

```

root@masha:/home/masha/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 (/root/.symfony5/log/2fb8442481889aafabfad5300a6ca9649da43182.log)
Following PHP log file (/root/.symfony5/log/2fb8442481889aafabfad5300a6ca9649da43182/7daf403c7589f4927632ed3b6af762a992f09b78.log)
[WARNING] the current dir requires PHP 8.1.0 (composer.json from current dir: /home/masha/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 29 14:39:29 |DEBUG | PHP Reloading PHP versions
[Web Server ] Jan 29 14:39:29 |WARN | PHP the current dir requires PHP 8.1.0 (composer.json from current dir: /home/masha/demo/composer.json), but this version is not available
[Web Server ] Jan 29 14:39:29 |DEBUG | PHP Using PHP version 8.1.2 (from default version in $PATH)
[Web Server ] Jan 29 14:39:29 |INFO | PHP listening path="/usr/bin/php8.1" php="8.1.2" port=35099
[PHP ] [Sun Jan 29 14:39:29 2023] PHP 8.1.2-ubuntu2.10 Development Server (http://127.0.0.1:35099) started

```

Рис. 6 – Запускаем проект

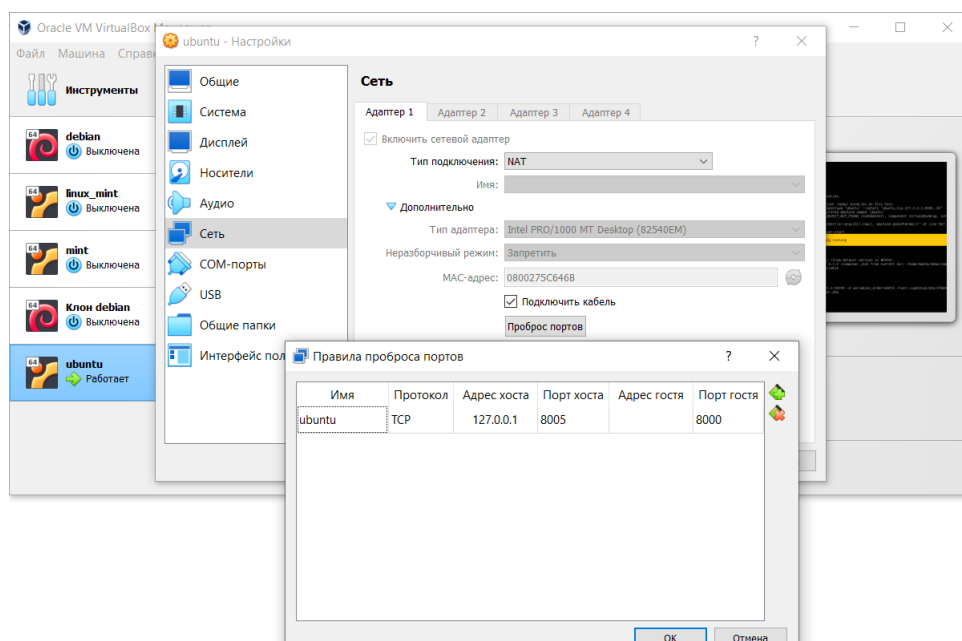


Рис. 7 – Пробрасываем порты, чтобы иметь доступ к интернету

```

root@masha:/home/masha/demo# composer require symfony/intl
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? y
Info from https://repo.packagist.org: #StandWithUkraine
Using version ^6.2 for symfony/intl
./composer.json has been updated
Running composer update symfony/intl
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading symfony/intl (v6.2.0 => v6.2.5)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 110 installs, 0 updates, 0 removals
  - Downloading symfony/flex (v2.2.3)
  - Downloading symfony/runtime (v6.2.0)

```

Рис. 8 – Устанавливаем пакеты symfony

```

root@masha:/home/masha/demo# composer require symfony/http-kernel
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? y
./composer.json has been updated
Running composer update symfony/http-kernel
Loading composer repositories with package information
Updating dependencies
Lock file operations: 0 installs, 1 update, 0 removals
  - Upgrading symfony/http-kernel (v6.2.0 => v6.2.5)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 0 installs, 1 update, 0 removals
  - Downloading symfony/http-kernel (v6.2.5)
  - Upgrading symfony/http-kernel (v6.2.0 => v6.2.5): Extracting archive
Generating autoload files

```

Рис. 9 – Устанавливаем пакеты symfony

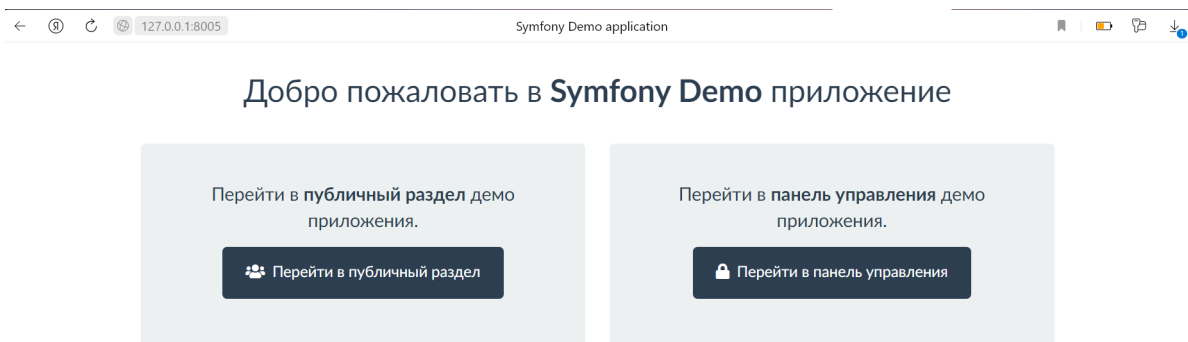


Рис. 10 – Открытый проект в браузере



```
root@masha:/home/masha/demo# touch Dockerfile
root@masha:/home/masha/demo# nano Dockerfile
```

Рис. 11 – Создаем файл Dockerfile

```
GNU nano 6.2
FROM nginx-php-fpm
WORKDIR /var>>/www/html/demo
COPY composer.install
COPY ..
EXPOSE 8005
CMD ["php","bin/console","server:start"]
```

Рис. 12 – Заполняем его

```
root@masha:/home/masha/demo# touch docker-compose.yml
root@masha:/home/masha/demo# nano docker-compose.yml
```

Рис. 13 – Создаем файл docker-compose.yml

```
GNU nano 6.2
version: "3"
services:
  app:
    container_name: docker-node-mongo
    restart: always
    build:
    ports:
      - "8003:8005"
    links:
      - postgres
  postgres:
    container_name: postgres
    image: postgres
    ports:
      - "5432:5432"
```

Рис. 14 – Заполняем его

```
postgres=# create database coolbd;
CREATE DATABASE
```

Рис. 15 – Создаем новую базу 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:29122002@127.0.0.1:5432/coolbd?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###

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

Рис. 16 – Заменяем DATABASE\_URL в .env на строку подключения к postgres и вводим туда необходимые данные

```
postgres=# \l
```

List of databases					
Name	Owner	Encoding	Collate	Ctype	Access privileges
coolbd	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 +
template1	postgres	UTF8	ru_RU.UTF-8	ru_RU.UTF-8	=c/postgres +
postgres=Ctc/postgres					

(4 rows)

Рис. 17 – Проверяем наличие созданной нами БД

```
postgres@masha:~$ composer require symfony/orm-pack
Using version ^2.3 for symfony/orm-pack
./composer.json has been updated
Running composer update symfony/orm-pack
Loading composer repositories with package information
```

Рис. 18 – Скачиваем пакет для работы doctrine

```
masha@masha:~/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!
```

Рис. 19 – Заполняем созданную БД данными

```
masha@masha:~/demo$ php bin/console doctrine:fixtures:load

Careful, database "coolbd" will be purged. Do you want to continue? (yes/no) [no]:
> y

> purging database
> loading App\DataFixtures\AppFixtures
```

Рис. 20 – Заполняем созданную БД данными

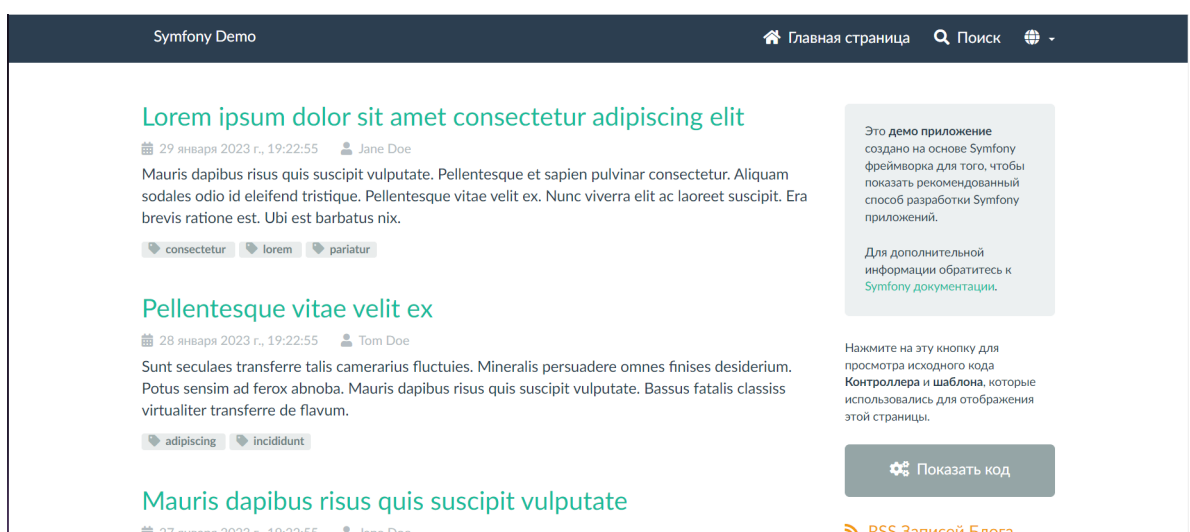


Рис. 21 – Проверяем работу созданной БД на сайте

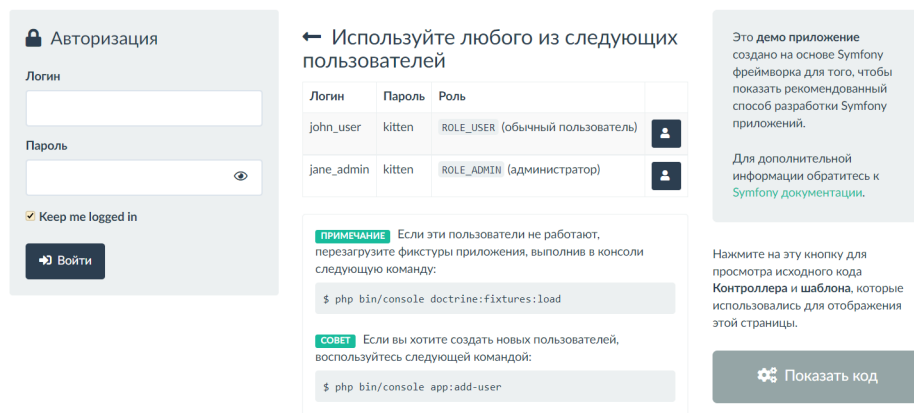


Рис. 22 – Замена на postgres прошла успешно

```
root@masha:/home/masha/demo# sudo -i -u postgres
postgres@masha:~$ pg_dump postgres://postgres:29122002@127.0.0.1:5432/coolbd > coolbd.sql
postgres@masha:~$ ls
14 composer.json composer.lock coolbd.sql vendor
postgres@masha:~$ pwd
/var/lib/postgresql
```

Рис. 23 – Создаем дамп нашей БД

```

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 >=1.0)
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-configuration

###> symfony/framework-bundle ###
APP_ENV=dev
APP_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/configuration.html
# 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:29122002@db:5432/coolbd?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###

```

Рис. 24 – Меняем IP адрес в .env на название контейнера "db"

```

Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
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: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 5b842e528e99d4d4c1686467debf2bd4b88ecd86
runc version: v1.1.4-0-g5fd4c4d
init version: de40ad0
Security Options:
  apparmor
  seccomp
   Profile: default
  cgroupns
Kernel Version: 5.15.0-58-generic
Operating System: Ubuntu 22.04.1 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 1.93GiB
Name: masha
ID: YXSJ:YH6Y:NWQR:M4IA:ZZ75:ISD2:QX4A:OVBJ:677S:K6EF:H0OL:2AKD
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

```

Рис. 25 – Выводим информацию о docker

```

GNU nano 6.2 .env *
###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=5b842e528e99d4d4c1686467debf2bd4b88ecd86_
###< symfony/framework-bundle ###
###> doctrine/doctrine-bundle ###
DATABASE_URL="postgresql://postgres:29122002@127.0.0.1:15432/coolbd?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###
###> nelmio/cors-bundle ###
CORS_ALLOW_ORIGIN='^https?:/(localhost|127\.0\.0\.1)(:|0-9|+)?$'
###< nelmio/cors-bundle ###

```

Рис. 26 – Разобьем проект на две папки docker и src. Содержимое src/ .env

```

GNU nano 6.2                               .env *
###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=5b842e528e99d4d4c1686467debf2bd4b88ecd86_
###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###
DATABASE_URL="postgresql://postgres:29122002@db:5432/coolbd?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###

###> nelmio/cors-bundle ###
CORS_ALLOW_ORIGIN='^https?://(localhost|127\.\.0\.\.1)(:[0-9]+)?$'
###< nelmio/cors-bundle ###

```

Рис. 27 – Содержимое docker/.env

```

GNU nano 6.2                               docker-compose.yml *
version: '3.8'
services:
  db:
    container_name: db
    image: postgres:12
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: 29122002
      POSTGRES_DB: coolbd
      PGDATA: /var/lib/postgresql/data
    ports:
      - 15432:5432
    volumes:
      - ./pg-data:/var/lib/postgresql/data
      - ./coolbd.sql:/docker-entrypoint-initdb.d/coolbd.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:
      - ../../src:/var/www
  nginx:
    container_name: nginx
    build:
      context: ./nginx
    volumes:

```

Рис. 28 – Содержимое docker-compose.yml

```

volumes:
  - ../../src:/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

```

Рис. 29 – Содержимое docker-compose.yml

```

FROM nginx:alpine

WORKDIR /var/www

CMD ["nginx"]

EXPOSE 80 443

```

Рис. 30 – Содержимое docker/nginx/Dockerfile

```

GNU nano 6.2 Dockerfile *
FROM php: 8.1-fpm

COPY wait-for-it.sh /usr/bin/wait-for-it

RUN chmod +x /usr/bin/wait-for-it

RUN apt-get update && \
  apt-get install -y --no-install-recommends libssl-dev curl git unzip netcat limvm12-dev libpq-dev \
  pecl install apcu && \
  docker-php-ext-configure pgsql -with-pgsql=/usr/local/pgsql && \
  docker-php-ext-install -g$(nproc) zip opcache intl pdo_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 ; wait-for-it db:5432; php-fpm

EXPOSE 9000_

```

Рис. 31 – Содержимое docker/php-fpm/Dockerfile



```

GNU nano 6.2                                     docker-co
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_coolbd
  db:
    image: mysql:5.7
    restart: always
    volumes:
      - "/opt/mysql:/var/lib/mysql"
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: wp_coolbd
      MYSQL_USER: root
      MYSQL_PASSWORD: 12345

```

Рис. 32 – Содержимое dream/docker-compose.yml

```
root@masha:/home/masha/dream# docker compose up
[+] Running 0/34
 * wordpress Pulling
   * 8740c948ffd4 Downloading 10.46MB/31.4MB
   * 1873be858264 Download complete
   * 7ce6a163d8c1 Downloading 9.73MB/91.63MB
   * 008a172010ba Download complete
   * d15353ae3d77 Downloading 192.8kB/19.24MB
   * 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
 * db Pulling
   * e048d0a38742 Waiting
   * c7847c8a41cb Waiting
   * c400748dd907 Waiting
   * c59c21f62b1f Waiting
   * 426e724ab5eb Waiting
   * cc537924e52d Waiting
   * caa4edfb8fd8 Waiting
   * cdce197f17dc Waiting
   * f3e7bdd9aebc Waiting
   * 08de2e85fded Waiting
   * 006f2ea3576a Waiting
```

Рис. 33 – Строим и запускаем докер

```

dream-db-1 | 2023-01-31 07:14:25+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
dream-db-1 | 2023-01-31 07:14:25+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
dream-db-1 | 2023-01-31 07:14:25+00:00 [ERROR] [Entrypoint]: MYSQL_USER="root", MYSQL_USER and MYSQL_PASSWORD are for configuring a regular user and cannot be used for the root user
dream-db-1 | Remove MYSQL_USER="root" and use one of the following to control the root user password:
dream-db-1 | - MYSQL_ROOT_PASSWORD
dream-db-1 | - MYSQL_ALLOW_EMPTY_PASSWORD
dream-db-1 | - MYSQL_RANDOM_ROOT_PASSWORD
dream-db-1 exited with code 1
dream-db-1 | 2023-01-31 07:15:25+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
dream-db-1 | 2023-01-31 07:15:25+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
dream-db-1 | 2023-01-31 07:15:25+00:00 [ERROR] [Entrypoint]: MYSQL_USER="root", MYSQL_USER and MYSQL_PASSWORD are for configuring a regular user and cannot be used for the root user
dream-db-1 | Remove MYSQL_USER="root" and use one of the following to control the root user password:
dream-db-1 | - MYSQL_ROOT_PASSWORD
dream-db-1 | - MYSQL_ALLOW_EMPTY_PASSWORD
dream-db-1 | - MYSQL_RANDOM_ROOT_PASSWORD
dream-db-1 exited with code 1
dream-db-1 | 2023-01-31 07:16:26+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
dream-db-1 | 2023-01-31 07:16:26+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
dream-db-1 | 2023-01-31 07:16:26+00:00 [ERROR] [Entrypoint]: MYSQL_USER="root", MYSQL_USER and MYSQL_PASSWORD are for configuring a regular user and cannot be used for the root user
dream-db-1 | Remove MYSQL_USER="root" and use one of the following to control the root user password:
dream-db-1 | - MYSQL_ROOT_PASSWORD
dream-db-1 | - MYSQL_ALLOW_EMPTY_PASSWORD
dream-db-1 | - MYSQL_RANDOM_ROOT_PASSWORD
dream-db-1 exited with code 1

```

Рис. 34 – Успешный запуск

```

root@masha:/home/masha# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
df0cf583bc19   wordpress:latest "docker-entrypoint.s..." 16 minutes ago Up 16 minutes
0.0.0.0:8080->80/tcp, :::8080->80/tcp   dream-wordpress-1
cfa2478566d9   mysql:5.7      "docker-entrypoint.s..." 16 minutes ago Restarting (1) 7 seconds
ago                                         dream-db-1

```

Рис. 35 – Смотрим, что докер работает

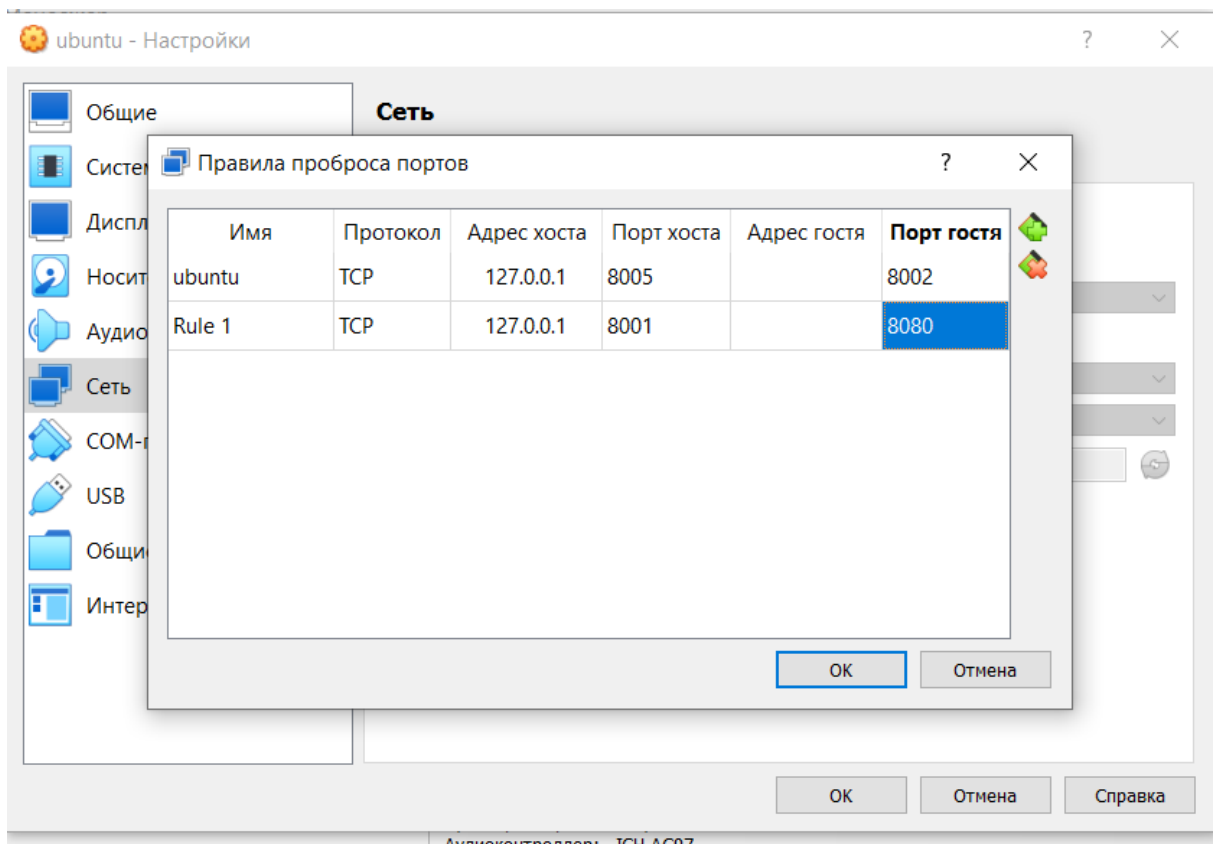


Рис. 36 – Пробрасываем порт для выхода в интернет

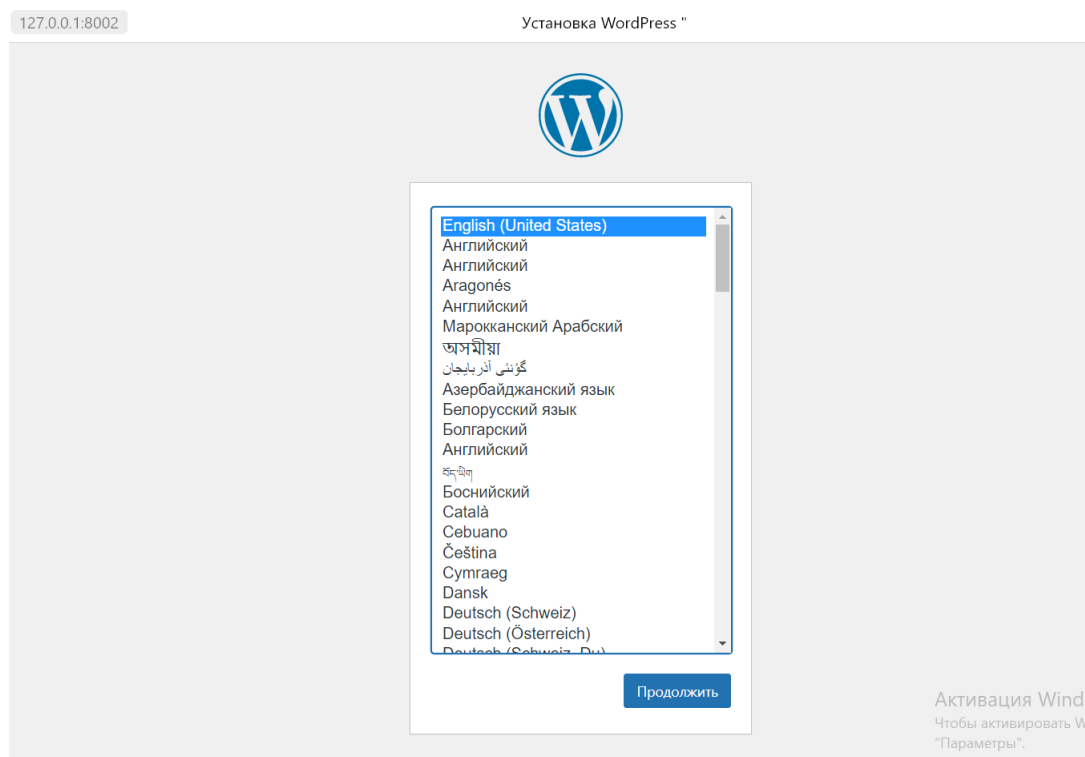



Рис. 37 – Успешная сборка. Регистрируемся на сайте

Добро пожаловать в знаменитую пятиминутную установку WordPress! Просто заполните поля — и вперёд к использованию самой мощной и гибкой персональной платформы для публикаций в мире!

## Требуется информация

Пожалуйста, укажите следующую информацию. Не переживайте, потом вы всегда сможете изменить эти настройки.

**Название сайта**

**Имя пользователя**  

Имя пользователя может содержать только латинские буквы, пробелы, подчёркивания, дефисы, точки и символ @.

**Пароль**  [Show](#)

Strong

**Важно:** Этот пароль понадобится вам для входа. Сохраните его в надёжном месте.

**Ваш e-mail**

Внимательно проверьте адрес электронной почты, перед тем как продолжить.

**Видимость для поисковых систем** ☐ Попросить поисковые системы не индексировать сайт

Будет ли учитываться этот запрос — зависит от поисковых систем.

[Установить WordPress](#)

Рис. 38 – Создаем аккаунт

dream
 2 0
 [Добавить](#)

Привет, masha
 [Настройки экрана](#)
[Помощь](#)

Консоль
 

Главная
 Обновления
 Записи
 Медиафайлы
 Страницы
 Комментарии
 Внешний вид
 Плагины
 Пользователи
 Инструменты
 Настройки
 Свернуть меню

## Добро пожаловать в WordPress!

[Узнайте больше о версии 6.1.1.](#)

**Создавайте богатое содержимое с блоками и паттернами**

Паттерны блоков - предварительно настроенные макеты расположения блоков. Используйте их, чтобы вдохновиться или мгновенно создавать новые страницы.

[Добавить новую страницу](#)

**Настройте весь свой сайт с помощью блочных тем**

Оформляйте всё на своем сайте - от заголовка до подвала, используя блоки и паттерны.

[Открыть редактор сайта](#)

**Измените внешний вид своего сайта с помощью стилей**

Слегка измените свой сайт или придайте ему полностью новый вид! Подойдите к вопросу творчески, как насчет новой цветовой палитры или шрифта?

[Редактировать стили](#)

Рис. 39 – Регистрация пройдена успешно

## 4. Контрольные вопросы

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.
  - Меньшие накладные расходы на инфраструктуру.
2. Назовите основные компоненты Docker.
  - Контейнеры.
3. Какие технологии используются для работы с контейнерами?
  - Контрольные группы (cgroups)
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?
  - Главное отличие – способ работы. При виртуализации создается полностью отдельная операционная система. При контейнеризации используется ядро операционной системы той машины, на которой открывается контейнер.
  - Ещё одно значимое отличие – размер и скорость работы. Размер виртуальной машины может составлять несколько гигабайт. Также для загрузки операционной системы и запуска приложений, которые в них размещены, требуется много времени. Контейнеры более лёгкие — их размер измеряется в мегабайтах. По сравнению с виртуальными машинами, контейнеры могут запускаться намного быстрее.

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

- Контейнеры (`docker container my_command`):
  - `create` – Создать контейнер из изображения.
  - `start` – Запустить существующий контейнер.
  - `run` – Создайте новый контейнер и запустите его.
  - `ls` – Список работающих контейнеров.
  - `inspect` – Смотрите много информации о контейнере.
  - `logs` – Печать журналов.
  - `stop` – Изящно прекратить запуск контейнера.
  - `kill` – внезапно остановить основной процесс в контейнере.
  - `rm` – Удалить остановленный контейнер.
- Изображения (`docker image my_command`):
  - `build` – Построить образ.
  - `push` – Нажмите на изображение в удаленном реестре.
  - `ls` – Список изображений.
  - `history` – Смотрите промежуточную информацию изображения.
  - `inspect` – Смотрите много информации об изображении, в том числе слоев.
  - `rm` – Удалить изображение.

7. Каким образом осуществляется поиск образов контейнеров?

- Изначально Docker проверяет локальный репозиторий на наличие нужного образа. Если образ не найден, Docker проверяет удаленный репозиторий.

8. Каким образом осуществляется запуск контейнера?

- Docker выполняет инициализацию и запуск ранее созданного по образу контейнера по его имени.

9. Что значит управлять состоянием контейнеров?

- Это значит иметь возможность взаимодействовать с контролирующим его процессом.
10. Как изолировать контейнер?
    - Сконфигурировать необходимые для этого файлы «docker-compose.yml» и «Dockerfile».
  11. Опишите последовательность создания новых образов, назначение Dockerfile?
    - Для создания нового образа выбирается основа образа (любой подходящий пакет из репозитория Docker Hub), добавляются необходимые слои, выполняются нужные операции и разворачивается рабочее окружение внутри контейнера с необходимыми зависимостями. После чего происходит сборка образа. Dockerfile – это простой текстовый файл с инструкциями по созданию образа Docker. Он содержит все команды, которые пользователь может вызвать в командной строке для создания образа.
  12. Возможно ли работать с контейнерами Docker без одноименного движка?
    - Да, возможно при использовании среды другой виртуализации.
  13. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes?
    - Назначение Kubernetes состоит в выстраивании эффективной системы распределения контейнеров по узлам кластера в зависимости от текущей нагрузки и имеющихся потребностей при работе сервисов. Kubernetes способен обслуживать сразу большое количество хостов, запускать на них многочисленные контейнеры Docker или Rocket, отслеживать их состояние, кон-



тролировать совместную работу и репликацию, проводить масштабирование и балансировку нагрузки.

- Основные объекты:

- Kubectl Command Line Interface (kubectl.md): kubectl интерфейс командной строки для управления Kubernetes.
- Volumes (volumes.md): Volume(раздел) это директория, возможно, с данными в ней, которая доступна в контейнере.
- Labels (labels.md): Label'ы это пары ключ/значение которые прикрепляются к объектам, например pod'ам. Label'ы могут быть использованы для создания и выбора наборов объектов
- Replication Controllers (replication-controller.md): replication controller гарантирует, что определенное количество «реплик» pod'ы будут запущены в любой момент времени.
- Services (services.md): Сервис в Kubernetes это абстракция которая определяет логический объединённый набор pod и политику доступа к ним.
- Pods (pods.md): Pod это группа контейнеров с общими разделами, запускаемых как единое целое.
- Nodes (node.md): Нода это машина в кластере Kubernetes.