

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**Лабораторная работа № 5**

**по OS Linux**

**Контейнеризация**

Студент

Комаричев А. В.

Группа АИ-19

Руководитель

Кургасов В. В.

Липецк 2021г.

## Цель работы

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

## Ход работы

Для начала работы необходимо установить php8.0, composer, symphony, sqlite.

```
alex@alexserver:~$ php -v
PHP 8.0.14 (cli) (built: Dec 20 2021 21:22:57) ( NTS )
Copyright (c) The PHP Group
Zend Engine v4.0.14, Copyright (c) Zend Technologies
with Zend OPcache v8.0.14, Copyright (c), by Zend Technologies
```

Рисунок 1 – php

```
alex@alexserver:~$ composer |head
  _____
 /         \       /
/          \     /
\          /     \
 \        /       \
  _____       \
Composer version 2.2.4 2022-01-08 12:30:42

Usage:
  command [options] [arguments]

    [Symfony\Component\Console\Exception\RuntimeException]
    Unable to write output.

list [--xml] [--raw] [--format FORMAT] [--] [<namespace>]
```

Рисунок 2 – composer

```
alex@alexserver:~$ symfony | head
Symfony CLI version 5.0.8 (c) 2017-2022 Symfony SAS (2022-01-08T16:44:43Z - stable)
Symfony CLI helps developers manage projects, from local code to remote infrastructure

These are common commands used in various situations:

Work on a project locally

  new                                Create a new Symfony project
  server:start                       Run a local web server
  server:stop                        Stop the local web server
```

Рисунок 3 – symfony

Клонируем демо проект командой git clone <https://github.com/symfony/demo>

После того как проект клонирован и окружение настроено, перейдем в папку проекта.

```
alex@alexserver:~/demo$ ls
assets      CONTRIBUTING.md  phpstan-baseline.neon  src          var
bin         data            phpstan.neon.dist     symfony.lock vendor
composer.json LICENSE        phpunit.xml.dist      templates    webpack.config.js
composer.lock migrations     public                tests        yarn.lock
config      package.json    README.md              translations
```

Рисунок 4 – demo проект

Запускаем проект symfony serve

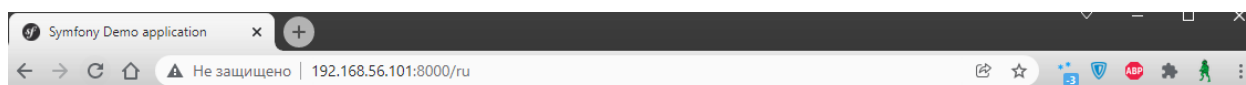
```
^Calex@alexserver:~/demo$ symfony serve -d

[WARNING] run "symfony server:ca:install" first if you want
t, or use "--no-tls" to avoid this warning

[OK] Web server listening
      The Web server is using PHP FPM 8.0.14
      http://127.0.0.1:8000

Stream the logs via symfony server:log
```

Запущенный проект можно видеть в браузере по адресу localhost. Адрес у меня в браузере отличается, так как проект запущен на гостевой машине, а браузер установлен на основной ос.



## Добро пожаловать в **Symfony Demo** приложение

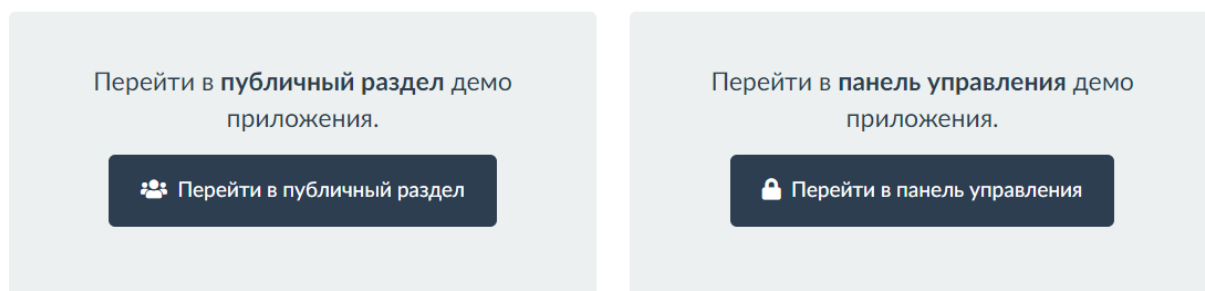


Рисунок 5 – Запуск проекта

Установим docker и docker-compose. Пакет установки Docker, доступный в официальном репозитории Ubuntu, может содержать не самую последнюю версию. Чтобы точно использовать самую актуальную версию, мы будем устанавливать Docker из официального репозитория Docker. Для этого мы добавим новый источник пакета, ключ GPG от Docker, чтобы гарантировать

загрузку рабочих файлов, а затем установим пакет. Для установки docker-compose используем команду

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

```
alex@alexserver:~$ sudo systemctl status docker
• docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-01-10 14:55:17 UTC; 1min 0s ago
   TriggeredBy: • docker.socket
   Docs: https://docs.docker.com
   Main PID: 8713 (dockerd)
   Tasks: 7
   Memory: 27.5M
   CGroup: /system.slice/docker.service
           └─8713 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Рисунок 6 – Docker установлен и работает

```
alex@alexserver:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
```

Рисунок 7 – docker-compose установлен

Установим postgresql, чтобы заменить sqlite базу (sudo apt -y install postgresql).

```
postgres=# \l
          List of databases
  Name      | Owner   | Encoding | Collate | Ctype   | Access privileges
-----+-----+-----+-----+-----+-----
 mydb       | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | =Tc/postgres +
            |          |          |             |             | postgres=Ctc/postgres+
            |          |          |             |             | myuser=Ctc/postgres
 postgres   | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | =c/postgres +
 template0  | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | postgres=Ctc/postgres
 template1  | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | =c/postgres +
            |          |          |             |             | postgres=Ctc/postgres
(4 rows)
```

Рисунок 8 – Создаем базу данных (mydb)

Редактируем файл .env. В DATABASE\_URL указываем postgresql базу, указываем адрес, порт, название базы данных, имя пользователя и его пароль.

```

GNU nano 4.8                               .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.
#
# Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure)

###> 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
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.y
#
DATABASE_URL=postgresql://127.0.0.1:5432/mydb?user=myuser&password=1234
#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://db_user:db_password@127.0.0.1:5432/db_name?serverVersion=13&charset=
###< doctrine/doctrine-bundle ###
MAILER_URL=null://localhost
###> symfony/mailer ###

```

Рисунок 9 – Изменение файла .env

Заполняем базу данных данными из фикстур. Команды:

php bin/console doctrine:schema:create,

php bin/console doctrine:fixtures:load.

Для этого необходимо также установить драйвер pdo-pgsql (sudo apt install php-pgsql).

Для проверки работоспособности новой базы данных, добавим туда новую запись.

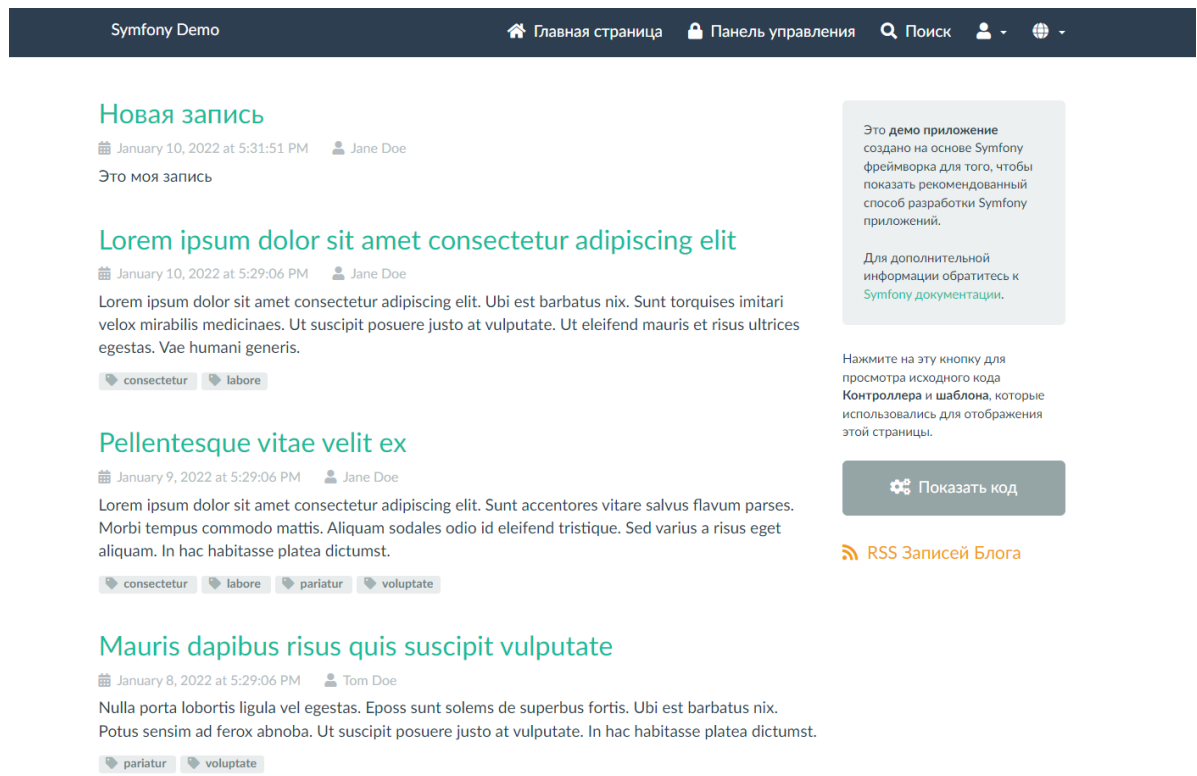


Рисунок 10 – Добавление новой записи

Проект работает, перейдем к настройке контейнеров.

Создадим папку `docker` внутри проекта. В этой папке создадим файл `docker-compose.yaml` в этом файле будет конфигурация всех контейнеров. Создадим папку `nginx` и `Dockerfile` в ней с конфигурацией контейнера `nginx`, папку `php-fpm` и `Dockerfile` в ней с конфигурацией контейнера `php-fpm`, папку `postgres` и `Dockerfile` в ней с конфигурацией контейнера `postgres`.

```
GNU nano 4.8 Dockerfile
FROM nginx
COPY ./config.conf /etc/nginx/conf.d/default.conf
WORKDIR /var/www
CMD ["nginx"]
EXPOSE 80 443
```

Рисунок 11 – Dockerfile nginx

Для конфигурации `nginx` также создадим файл `config.conf`

```

GNU nano 4.8                                     config.conf
server {
    listen 80;
    root /var/www/symfony/public;
    server_name_;
    error_log /var/log/nginx/symfony_error.log;
    access_log /var/log/nginx/symfony_access.log;
    location / {
        try_files $uri /$uri /index.php?$query_string;
    }
    location ~ ^/index\.php(/|$){
        fastcgi_pass php-fpm:9000;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param HTTPS off;
    }
}

```

Рисунок 12 – config.conf

```

GNU nano 4.8                                     Dockerfile
FROM php:8.0-fpm
RUN apt-get update && apt-get install -y \
    libfreetype6-dev \
    libjpeg62-turbo-dev \
    libpng-dev \
    && docker-php-ext-configure gd --with-freetype --with-jpeg \
    && docker-php-ext-install -j$(nproc) gd

```

Рисунок 13 – Dockerfile php-fpm

```

GNU nano 4.8                                     Dockerfile
FROM postgres:9.4
RUN localedef -i de_DE -c -f UTF-8 -A /usr/share/locale/locale.alias de_DE.UTF-8
ENV LANG de_DE.utf8

```

Рисунок 14 – Dockerfile postgres



```
GNU nano 4.8                                docker-compose.yaml                                Modified
version: '3'
services:
  postgres:
    build: postgres
    image: postgres
    restart: always
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: 1234
      POSTGRES_DB: mydb
    ports:
      - '54320:5432'
    volumes:
      - ./pg-data:/var/lib/postgresql/data/
      - ./db_backup.sql:/docker-entrypoint-initdb.d/db_backup.sql
  php:
    build: php-fpm
    ports:
      - "9002:9000"
    links:
      - postgres
    volumes:
      - ../:/var/www/symfony:cached
      - ../logs/symfony:/var/www/symfony/var/logs:cached
  nginx:
    build: nginx
    ports:
      - "8080:80"
    links:
      - php
    volumes:
      - ../:/var/www/symfony
      - ../logs/nginx:/var/log/nginx
```

Рисунок 15 – Файл docker-compose.yaml

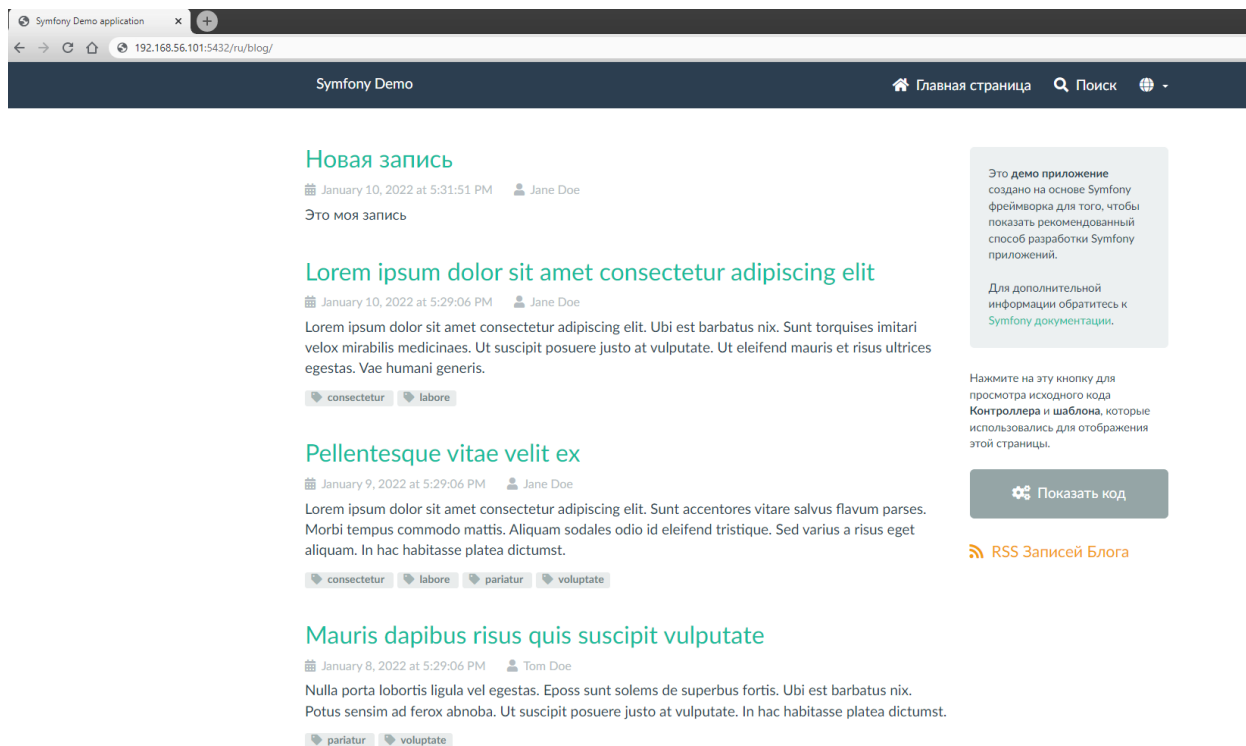
Далее редактируем файл .env. Вместо 127.0.0.1 указываем имя контейнера.

```
GNU nano 4.8                                .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.
#
# 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 ###
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/c
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yam
#
DATABASE_URL=postgresql://postgres:5432/mydb?user=myuser&password=1234
#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://db_user:db_password@127.0.0.1:5432/db_name?serverVersion=13&charset=utf8"
###< doctrine/doctrine-bundle ###
MAILER_URL=null://localhost
###> symfony/mailer ###
```

Рисунок 16 – Файл .env

## Запускаем проект командой docker-compose up.



## Рисунок 17 – Проект, запущенный из контейнера

## Контрольные вопросы

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.

Основное отличие контейнеров и виртуальных машин заключается в том, что виртуальные машины виртуализируют весь компьютер вплоть до аппаратных уровней, а контейнеры — только программные уровни выше уровня операционной системы. Следовательно, меньшие накладные расходы на инфраструктуру.

2. Основные компоненты Docker

Контейнеры.

3. Какие технологии используются для работы с контейнерами?

Контрольные группы.

4. Соответствие между компонентом и его описанием

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

5. В чем отличие контейнеров от виртуализации?

Виртуальная машина.

Требуется гипервизор, для каждой ВМ используется собственная гостевая ОС. Позволяет создавать неоднородные вычислительные среды на одном компьютере. Из-за собственной ОС ВМ может занимать несколько ГБ, а запуск ОС и всех приложений занимает какое-то время.

Контейнер.

Даже несколько контейнеров используют ядро одной хостовой ОС. Позволяет

создавать на одном компьютере только однородные вычислительные среды. Намного легче ВМ, размер измеряется в Мб. Способен запускаться почти мгновенно.

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

Команда	Описание
docker login (logout)	Вход (выход) в реестр
docker search nginx	Поиск образа
docker pull (push) nginx	Выгрузка из реестра образа (загрузка в реестр)
docker create [options] image [command] [arg...]	Создание контейнера
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]	Запуск контейнера
docker rename	Переименование контейнера
docker rm	Удаление контейнера
docker update	Обновление контейнера
docker stop (start) container	Остановка (запуск) контейнера
docker restart nginx	Перезагрузка контейнера
docker kill -s HUP nginx	Отправка сигнала контейнеру
docker ps	Работающие контейнеры
docker images	Список образов
docker build ...	Создание образов
docker rmi	Удаление образа

7. Каким образом осуществляется поиск образов контейнеров?

Docker проверяет локальный репозиторий на наличие искомого образа, если образ не был найден, производится поиск в репозитории Docker hub.

8. Каким образом осуществляется запуск контейнера?

Docker выполняет инициализацию и запускает контейнер. Контейнер собирается из образа.

9. Что значит управлять состоянием контейнеров?

В любой момент времени можно запустить, остановить или выполнить команду внутри контейнера.

10. Как изолировать контейнер?

Контейнеры являются изолированными.

11. Опишите последовательность создания новых образов, назначение Dockerfile?

Для создания нового образа необходимо выбрать основу на Docker Hub, затем произвести конфигурацию Dockerfile. Dockerfile – это текстовый файл с инструкциями для создания образа контейнера. Инструкции включают идентификацию существующего образа, команды, выполняемые в процессе создания образа и команды, выполняемые при развертывании новых экземпляров этого контейнера.

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

Да возможно, в среде виртуализации Kubernetes.

13. Опишите назначение системы оркестрации контейнеров Kubernetes.

Перечислите основные объекты Kubernetes?

Kubernetes – это открытое программное обеспечение для автоматизации развертывания, масштабирования контейнеризированных приложений и управления ими. Поддерживает основные технологии контейнеризации.

Основные объекты:

- 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.