**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

# ЛАБОРАТОРНАЯ РАБОТА №5

по Операционной системе Linux

Контейнеризация

Студент Жидков И.А.

Группа АС-19

Руководитель Кургасов В.В.

Доцент, к.п.н.

Липецк 2021 г.

Оглавление

Цель работы

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

Ход работы

Первоначально клонируем себе тестовый проект с помощью команды «git clone https://github.com/symfony/demo»

```
author@labserver:~$ git clone https://github.com/symfony/demo
Cloning into 'demo'...
remote: Enumerating objects: 10409, done.
remote: Counting objects: 100% (534/534), done.
remote: Compressing objects: 100% (335/335), done.
remote: Total 10409 (delta 264), reused 396 (delta 177), pack-reused 9875
Receiving objects: 100% (10409/10409), 18.88 MiB | 1.29 MiB/s, done.
Resolving deltas: 100% (6206/6206), done.
```

Рисунок 1 – скачивание тестового проекта

Перейдем в папку с проектом с помощью команды «cd demo»

```
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
info: Switch to mpm prefork for package libapache2-mod-php7.4
Module mpm_event disabled.
Enabling module mpm_prefork.
info: Executing deferred 'a2enmod php7.4' for package libapache2-mod-php7.4
Enabling module php7.4.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/sys
ache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib
md/system/apache-htcacheclean.service.
Настраивается пакет php7.4 (7.4.3-4ubuntu2.8) …
Настраивается пакет php (2:7.4+75) …
Обрабатываются триггеры для ufw (0.36-6) …
Обрабатываются триггеры для systemd (245.4-4ubuntu3.11) …
Обрабатываются триггеры для man-db (2.9.1-1) …
Обрабатываются триггеры для libc-bin (2.31-0ubuntu9.2) …
Обрабатываются триггеры для php7.4-cli (7.4.3-4ubuntu2.8) …
Обрабатываются триггеры для libapache2-mod-php7.4 (7.4.3-4ubuntu2.8) …
author@labserver:~$ cd demo
author@labserver:~/demo$ ls
assets         CONTRIBUTING.md  phpstan-baseline.neon  src           var
bin            data             phpstan.neon.dist      symfony.lock  webpack.config.js
composer.json  LICENSE          phpunit.xml.dist       templates     yarn.lock
composer.lock  migrations       public                 tests
config         package.json     README.md              translations
author@labserver:~/demo$ _
```

Рисунок 2 – содержание проекта

Далее отредактируем конфигурационные файлы.

Содержимое .env:

```
###> symfony/framework-bundle ### APP_ENV=dev
APP_SECRET=743df4115e7e1dea13b473da07c09fe6 ###< symfony/framework-bundle ###
```

```
###> doctrine/doctrine-bundle ###

DATABASE_URL="postgresql://postgres:password@127.0.0.1:15432/lab_5?serv
erVersion=13&charset=utf8"

###< doctrine/doctrine-bundle ### ###> nelmio/cors-bundle ###

CORS_ALLOW_ORIGIN=’^https?://(localhost|127\.0\.0\.1)(:[0-9]+)?$’

###< nelmio/cors-bundle ### Содержимое файла docker/.env:

###> symfony/framework-bundle ### APP_ENV=dev
APP_SECRET=743df4115e7e1dea13b473da07c09fe6 ###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###

DATABASE_URL="postgresql://postgres:password@db:5432/lab_5?serverVersio
n=13&charset=utf8"

###< doctrine/doctrine-bundle ### ###> nelmio/cors-bundle ###

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

Содержимое docker-compose.yml:

```
version: ’3.8’

services:

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:

- "80:80"

- "443:443"

db:
```

```
container_name: db image: postgres:12 restart: always environment:

POSTGRES_USER: postgres POSTGRES_PASSWORD: password POSTGRES_DB: dbtest

ports:

- "15432:5432"

volumes:

./pg-data:/var/lib/postgresql/data
```

Содержимое файла docker/nginx/Dockerfile:

```
FROM nginx:alpine

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

Содержимое Dockerfile:

```
FROM php:8.0-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 zlib1g-dev curl 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 ; wait-for-it db:5432; php-fpm

EXPOSE 9000
```

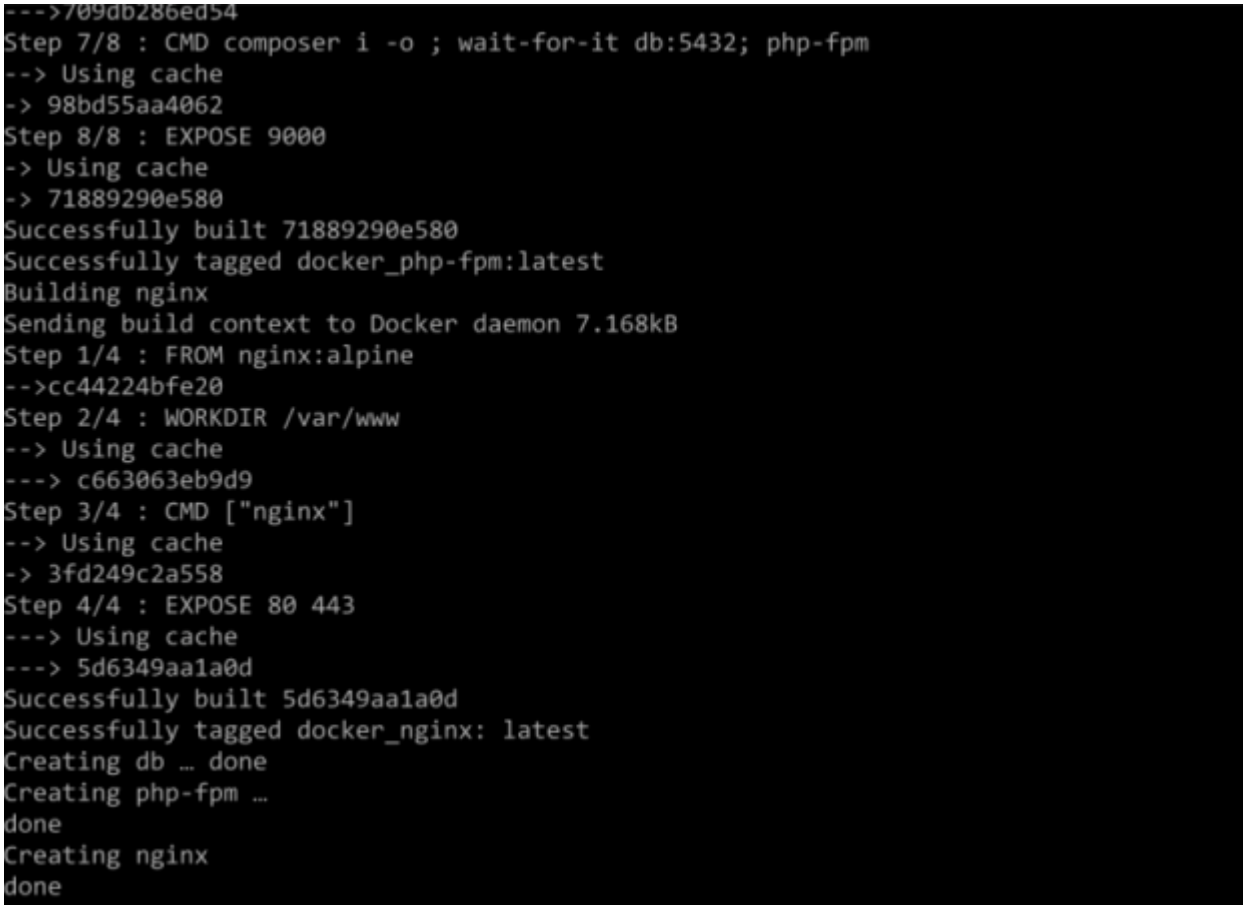


Рисунок 3 - сборка образа

Далее настроим сеть, поменяв настройки VirtualBox. NAT на Host-only adapter, в дополнительных настройках указав Allow VM. И запустим проект командой: symfony server:start -d

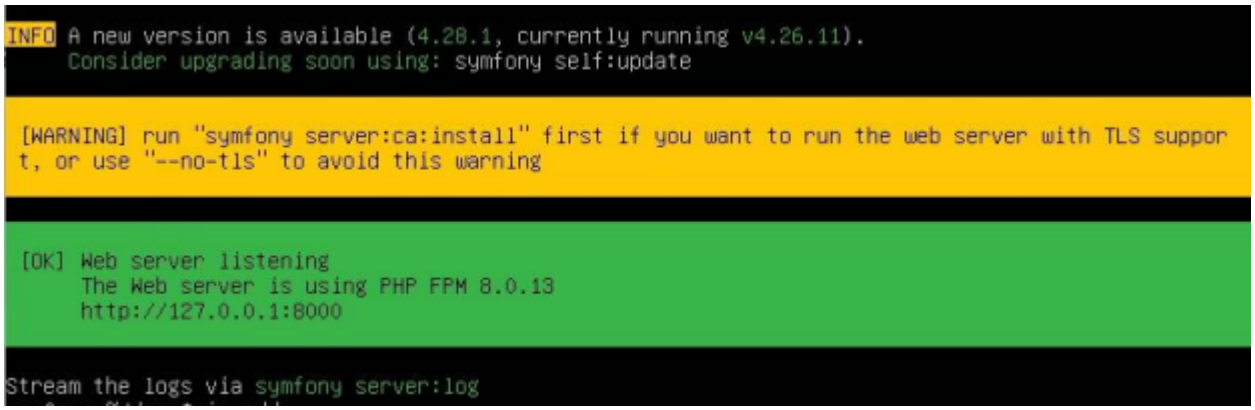


Рисунок 4 - сообщение об удачном запуске

Далее узнаем свой ip адрес при помощи команды ip addr

```
author@testserver:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 10
0
    link/ether 08:00:27:c6:e9:a4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic enp0s3
       valid_lft 395sec preferred_lft 395sec
    inet6 fe80::a00:27ff:fec6:e9a4/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:26:7e:5d:25 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
author@testserver:~$
```

Рисунок 5 - команда ip addr

Перейдем по соответствующему адресу в браузере на хост-машине.

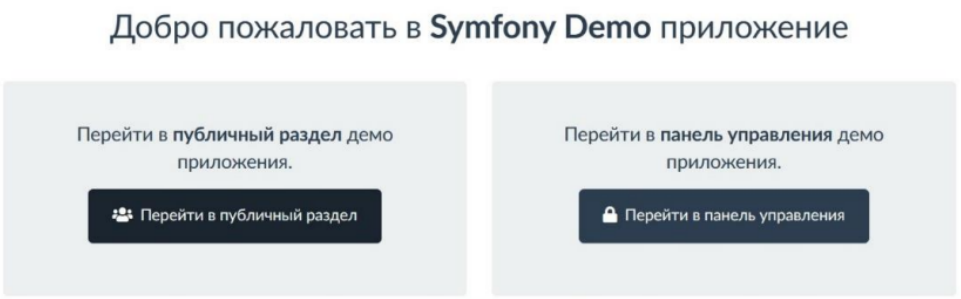


Рисунок 6 - главная страница сайта

Вывод

В ходе выполнения лабораторной работы были изучены современные методы разработки ПО в динамических и распределенных средах на примере контейнеров Docker.

Контрольные вопросы

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.

A. Меньшие накладные расходы на инфраструктуру

2. Назовите основные компоненты Docker.

B. Контейнеры

3. Какие технологии используются для работы с контейнерами?

C. 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?

Виртуальная машина – программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой целевой и исполняющая программы для гостевой платформы на платформе-хозяине (хосте) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы.

Виртуальные машины запускают на физических машинах, используя гипервизор.

В отличие от виртуальной машины, обеспечивающей аппаратную виртуализацию, контейнер обеспечивает виртуализацию на уровне операционной системы с помощью абстрагирования пользовательского пространства.

В целом контейнеры выглядят как виртуальные машины. Например, у них есть изолированное пространство для запуска приложений, они

позволяют выполнять команды с правами суперпользователя, имеют частный сетевой интерфейс и IP-адрес, пользовательские маршруты и правила межсетевого экрана и т. д. Одна большая разница между контейнерами и виртуальными машинами в том, что контейнеры разделяют ядро хоста с другими контейнерами.

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

— docker ps — показывает список запущенных контейнеров;

— docker pull — скачать определённый образ или набор образов (репозиторий);

— docker build — эта команда собирает образ Docker из Dockerfile и «контекста»;

— docker run — запускает контейнер, на основе указанного образа;

— docker logs — эта команда используется для просмотра логов указанного контейнера;

— docker volume ls — показывает список томов, которые являются предпочитаемым механизмом для сохранения данных, генерируемых и используемых контейнерами Docker;

— docker rm — удаляет один и более контейнеров;

— docker rmi — удаляет один и более образов;

— docker stop — останавливает один и более контейнеров;

— docker exec –it ... - выполняет команду в определенном контейнере

7. Каким образом осуществляется поиск образов контейнеров?

Сначала проверяется локальный репозиторий на наличия нужного контейнера, если он не найден локально, то поиск производится в репозитории Docker Hub.

8. Каким образом осуществляется запуск контейнера?

Для запуска контейнера его необходимо изначально создать из образа,

поэтому изначально контейнер собирается с помощью команды docker build,а уже затем запускается с помощью команды docker run.

9. Что значит управлять состоянием контейнеров?

Это означает, что в любой момент времени есть возможность запустить, остановить или выполнить команды внутри контейнера. Как изолировать контейнер?

Контейнеры уже по сути своей являются изолированными единицами, поэтому достаточно без ошибок сконфигурировать файлы Dockerfile и/или docker-compose.yml.

10. Опишите последовательность создания новых образов, назначение Dockerfile?

Производится выбор основы для нового образа на Docker Hub, далее производится конфигурация Dockerfile, где описываются все необходимые пакеты, файлы, команды и т.п.

Dockerfile — это текстовый файл с инструкциями, необходимыми для создания образа контейнера. Эти инструкции включают идентификацию существующего образа, используемого в качестве основы, команды, выполняемые в процессе создания образа, и команду, которая будет выполняться при развертывании новых экземпляров этого образа контейнера.

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

Да, если использовать Kubernetes

12. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes.

Kubernetes — открытое программное обеспечение для автоматизации развёртывания, масштабирования контейнеризированных приложений и управления ими. Поддерживает основные технологии контейнеризации, включая Docker, rkt, также возможна поддержка технологий аппаратной

виртуализации.

— Nodes: Нода это машина в кластере Kubernetes.

— Pods: Pod это группа контейнеров с общими разделами, запускаемых как единое целое.

— Replication Controllers: replication controller гарантирует, что определенное количество «реплик» pod’ы будут запущены в любой момент времени. Services: Сервис в Kubernetes – это абстракция, которая определяет логический объединённый набор pod и политику доступа к ним.

— Volumes: Volume(раздел) это директория, возможно, с данными в ней, которая доступна в контейнере.

— Labels: Label’ы это пары ключ/значение которые прикрепляются к объектам, например pod’ам. Label’ы могут быть использованы для создания и выбора наборов объектов.

— Kubectl Command Line Interface: kubectl интерфейс командной строки для управления Kubernetes.