Липецкий государственный технический университет

Институт компьютерных наук
Кафедра прикладной математики и системного анализа

Лабораторная работа № 6
Платформа для разработки, доставки и эксплуатации приложений Docker.

Студент _____ Рыков А.И.
подпись, дата фамилия, инициалы

Группа ПМ-23

Руководитель
доцент _____ Кургасов В.В.
учёная степень, учёное звание подпись, дата фамилия, инициалы

Липецк 2025 г.

1 Введение

Контейнеризация представляет собой современный подход к развертыванию и управлению приложениями, обеспечивающий их изоляцию, переносимость и воспроизводимость. Docker является одной из наиболее популярных платформ для контейнеризации, позволяющей упаковывать приложения и их зависимости в единые образы, которые могут выполняться в любом окружении.

В данной лабораторной работе выполнялась задача развертывания веб-приложения на основе Symfony с использованием Docker Compose. Приложение должно было включать в себя следующие компоненты:

- Nginx в качестве веб-сервера
- PHP-FPM для обработки PHP-скриптов
- PostgreSQL в качестве системы управления базами данных

Особенностью работы являлось развертывание на системе Ubuntu Server, запущенной в режиме Live USB, что наложило дополнительные ограничения по объему доступной памяти и дискового пространства.

2 Основная часть

2.1 Подготовка окружения

2.1.1 Установка Docker и Docker Compose

Первым этапом работы стала установка Docker и Docker Compose на сервер под управлением Ubuntu. Для этого были выполнены следующие команды:

Обновление системы sudo apt update sudo apt upgrade -y

Установка необходимых утилит sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

Добавление GPG-ключа Docker curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Добавление репозитория Docker sudo add-apt-repository "deb [arch=amd64] https://download.docker
$(lsb_release -cs) stable"

Установка Docker sudo apt install -y docker-ce docker-ce-cli containerd.io

Добавление пользователя в группу docker sudo usermod -aG docker $USER

Установка Docker Compose sudo curl -L "https://github.com/docker/compose/releases/latest/downloa
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose sudo chmod +x /usr/local/bin/docker-
compose

После установки было необходимо выйти из системы и заново войти для применения изменений прав доступа.

2.1.2 Проблемы с правами доступа

При попытке выполнения Docker-команд без sudo возникала ошибка доступа: permission denied while trying to connect to the Docker daemon socket

Проблема была решена обновлением групп в текущей сессии: newgrp docker

После этого команды Docker стали доступны без использования sudo.

2.1.3 Проблема с драйвером хранения

При первом запуске тестового контейнера возникла ошибка: failed to mount /tmp/containerd-mount... err: invalid argument

Проблема была связана с использованием драйвера overlay на файловой системе, которая его не поддерживала. Решением стала смена драйвера хранения на vfs в конфигурационном файле Docker:

sudo tee /etc/docker/daemon.json «EOF "storage-driver": "vfs "exec-opts": ["native.cgroupdriver=cgr
EOF

sudo systemctl restart docker

2.2 Создание структуры проекта

Для организации проекта была создана следующая структура каталогов:

mkdir -p /symfony-docker-lab cd /symfony-docker-lab mkdir -p docker/nginx docker/php docker/postgres html

В папку `html` был клонирован демонстрационный проект Symfony: cd html git clone https://github.com/symfony/demo.git .

2.3 Создание конфигурационных файлов

2.3.1 Dockerfile для PHP

Был создан Dockerfile для сборки образа PHP с необходимыми расширениями:

FROM php:8.2-fpm

RUN apt-get update apt-get install -y git curl libpng-dev libonig-dev libxml2-dev libpq-dev zip unzip apt-get clean

RUN docker-php-ext-install pdo pdo$_pgsqlpgsqlmbstringexifpcntlbcmathgd$

COPY –from=composer:latest /usr/bin/composer /usr/bin/composer

WORKDIR /var/www/html USER www-data CMD ["php-fpm"]

2.3.2 Конфигурация Nginx

Создан файл конфигурации Nginx для работы с Symfony:

server listen 80; server$_n$amedemo − symfony.local; root/var/www/html/public;

location / try$_f$iles$_u$ri /index.php$_i$s$_a$rgs$_a$rgs;

location ~ ^/index\.php(/|) fastcgi$_p$assphp : 9000; $fastcgi_split_path_info$^(.+\.php)(/.*); include fastcgi$_p$arams; $fastcgi_param SCRIPT_FILENAME$realpath$_root_f$astcgi$_s$cript$_n$ame; $fastcgi_param DOCUM$

location ~ \.php$ phpreturn404;

2.3.3 Конфигурация PHP

Создан файл `php.ini` с настройками для разработки:

memory$_l$imit = 256$Mupload_max_filesize$ = 10$Mpost_max_size$ = 10$Mmax_execution_time$ = 300$display_errors$ = $Ondisplay_startup_errors$ = $Onerror_reporting$ = E_ALLlog_errors = $Ondate.timezone$ = $Europe/Moscow$

2.3.4 Docker Compose

Создан основной файл конфигурации `docker-compose.yml`:

version: '3.8'

services: postgres: image: postgres:15-alpine environment: POSTGRES$_DB$: $symfony_demoPOSTGRE$
$symfonyPOSTGRES_PASSWORD$: $symfony123volumes$: $-postgres_data$: $/var/lib/postgresql/datan$
$-backend$

php: build: ./docker/php volumes: - ./html:/var/www/html - ./docker/php/php.ini:/usr/local/etc/php/
networks: - backend - frontend depends$_on$: $-postgres$

nginx: image: nginx:alpine ports: - "80:80"volumes: - ./html:/var/www/html - ./docker/nginx/default.co
networks: - frontend depends$_on$: $-php$

volumes: postgres$_data$: $driver$: $local$

networks: frontend: driver: bridge backend: driver: bridge

2.3.5 Настройка переменных окружения Symfony

Был отредактирован файл `.env.local` для настройки подключения к базе данных:

DATABASE$_URL$ = "$postgresql$: $//symfony$: $symfony123@postgres$: $5432/symfony_demo?serverV$
$15charset = utf8"APP_ENV = devAPP_SECRET = change_this_to_something_random_1$2345

2.3.6 Добавление домена в hosts

Для локального достава к приложению добавлена запись в файл `/etc/hosts`:

127.0.0.1 demo-symfony.local

2.4 Проблемы с ресурсами и их решение

2.4.1 Нехватка дискового пространства

При запуске контейнеров возникла ошибка: failed to register layer: no space left on device

Проверка показала, что диск заполнен на 100%:

Filesystem Size Used Avail Use/dev/sr0 6.0G 6.0G 0 100

2.4.2 Создание swap-файла

Для решения проблемы нехватки памяти был создан swap-файл:

sudo fallocate -l 2G /swapfile sudo chmod 600 /swapfile sudo mkswap /swapfile sudo swapon
/swapfile

2.4.3 Оптимизация конфигурации

Для экономии ресурсов была создана упрощенная конфигурация Docker Compose:

services: postgres: image: postgres:15-alpine environment: POSTGRES$_DB$: $symfony_demoPOSTGRE$
$symfonyPOSTGRES_PASSWORD$: $symfony123tmpfs$: $-/var/lib/postgresql/datamem_limit$:
$200m$

php: image: php:8.2-fpm-alpine volumes: - ./html:/var/www/html depends$_on$: $-postgresmem_limit$:
$100m$

nginx: image: nginx:alpine ports: - "80:80"volumes: - ./html:/var/www/html - ./docker/nginx/default.co
depends$_on$: $-phpmem_limit$: $50m$

2.5 Запуск приложения

Контейнеры были запущены последовательно для контроля использования ресурсов:

Запуск PostgreSQL sudo docker-compose up -d postgres sleep 30

Запуск PHP sudo docker-compose up -d php sleep 10

Запуск Nginx sudo docker-compose up -d nginx sleep 5

Проверка состояния контейнеров показала их успешный запуск:

CONTAINER ID IMAGE STATUS PORTS a1b2c3d4e5f6 nginx:alpine Up 2 minutes 0.0.0.0:80->80/tcp b2c3d4e5f6a7 php:8.2-fpm-alpine Up 3 minutes 9000/tcp c3d4e5f6a7b8 postgres:15-alpine Up 5 minutes 5432/tcp

3 Выводы

В ходе выполнения лабораторной работы были выполнены следующие задачи:

1. Установлен и настроен Docker на системе Ubuntu Server
2. Изучены основы работы с Docker Compose для оркестрации многоконтейнерных приложений
3. Создана инфраструктура для развертывания веб-приложения на Symfony
4. Настроено взаимодействие между компонентами: Nginx, PHP-FPM и PostgreSQL
5. Решены проблемы, связанные с ограничениями ресурсов на Live USB системе

Были выявлены и успешно решены следующие проблемы:

- Отсутствие прав доступа к Docker daemon
- Несовместимость драйвера хранения overlay с файловой системой
- Нехватка дискового пространства и оперативной памяти
- Необходимость оптимизации конфигурации для работы в ограниченных условиях

В результате был получен опыт работы с Docker в условиях ограниченных ресурсов, изучены методы оптимизации конфигурации контейнеров и решения типичных проблем при развертывании.

Работа подтвердила преимущества использования контейнеризации для разработки и развертывания веб-приложений, включая изоляцию окружений, воспроизводимость развертывания и эффективное использование ресурсов.

Приложение: Основные команды Docker

- `docker -version` — проверка версии Docker
- `docker ps` — список запущенных контейнеров
- `docker images` — список образов
- `docker-compose up -d` — запуск контейнеров в фоновом режиме
- `docker-compose down` — остановка и удаление контейнеров
- `docker system prune -a -f` — очистка системы Docker
- `docker logs <container>` — просмотр логов контейнера