



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**  
**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт (факультет)  
Кафедра

Институт компьютерных наук  
Автоматизированные системы управления

**ЛАБОРАТОРНАЯ РАБОТА №6**

По дисциплине: «Операционные системы»

На тему: «Контейнеризация»

Студент

АИ-23

Группа

подпись, дата

Жданов М.С.

фамилия, инициалы

Руководитель

кандидат наук

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

фамилия, инициалы

Липецк 2025

### **Задание:**

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

## Ход работы:

Для начала я установил docker на свою виртуальную машину. Для этого первым делом нужно обновить систему командами:

```
sudo apt update  
sudo apt upgrade -y
```

Далее нужно было поставить зависимости:

```
sudo apt install ca-certificates curl gnupg lsb-release -y
```

Следующим шагом шло добавление ключей и репозитория Docker:

```
sudo mkdir -p /etc/apt/keyrings  
  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o  
/etc/apt/keyrings/docker.gpg  
  
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \  
https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >  
/dev/null
```

После чего можно было приступить к установке Docker и Compose:

```
sudo apt update  
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y
```

После этого проведём проверку установки командами:

```
docker --version  
  
docker compose version
```

Статус установки Docker можно видеть на рисунке 1:

```
Настраивается пакет pigz (2.8-1) ...
Настраивается пакет docker-ce-rootless-extras (5:29.1.3-1~ubuntu.24.04~noble) ...
Настраивается пакет slirp4netns (1.2.1-1build2) ...
Настраивается пакет docker-ce (5:29.1.3-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Обрабатываются триггеры для man-db (2.12.0-4build2) ...
Обрабатываются триггеры для libc-bin (2.39-0ubuntu8.6) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
tjtt@localhost:~$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
tjtt@localhost:~$ sudo systemctl start docker
tjtt@localhost:~$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-12-25 21:30:34 UTC; 1min 9s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 1740 (dockerd)
       Tasks: 9
      Memory: 27.3M (peak: 27.8M)
         CPU: 81ms
      CGroup: /system.slice/docker.service
              └─1740 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.275175425Z" level=info msg="Restoring containers: start."
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.287588018Z" level=info msg="Deleting nftables IPv4 rules" en
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.297566384Z" level=info msg="Deleting nftables IPv6 rules" en
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.362564939Z" level=info msg="Loading containers: done."
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.364495727Z" level=info msg="Docker daemon" commit=fbf3ed2 co
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.364548007Z" level=info msg="Initializing buildkit"
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.372101059Z" level=info msg="Completed buildkit initialization
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.375430860Z" level=info msg="Daemon has completed initializati
дек 25 21:30:34 localhost dockerd[1740]: time="2025-12-25T21:30:34.375443983Z" level=info msg="API listen on /run/docker.sock"
дек 25 21:30:34 localhost systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

Рисунок 1 – Итог установки Docker

Проверка установки путём запуска «hello world» - рисунок 2:

```
tjtt@localhost:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
198f93fd5094: Pull complete
95ce02e4a4f1: Download complete
Digest: sha256:d4aabb6242e0cace87e2ec17a2ed3d779d18fbfd03042ea58f2995626396a274
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

tjtt@localhost:~$ _
```

Рисунок 2 – Проверка Docker

Далее для теста я создал простой uml файл и html файл и запустил, чтобы удостовериться, что всё работает – рисунок 3:

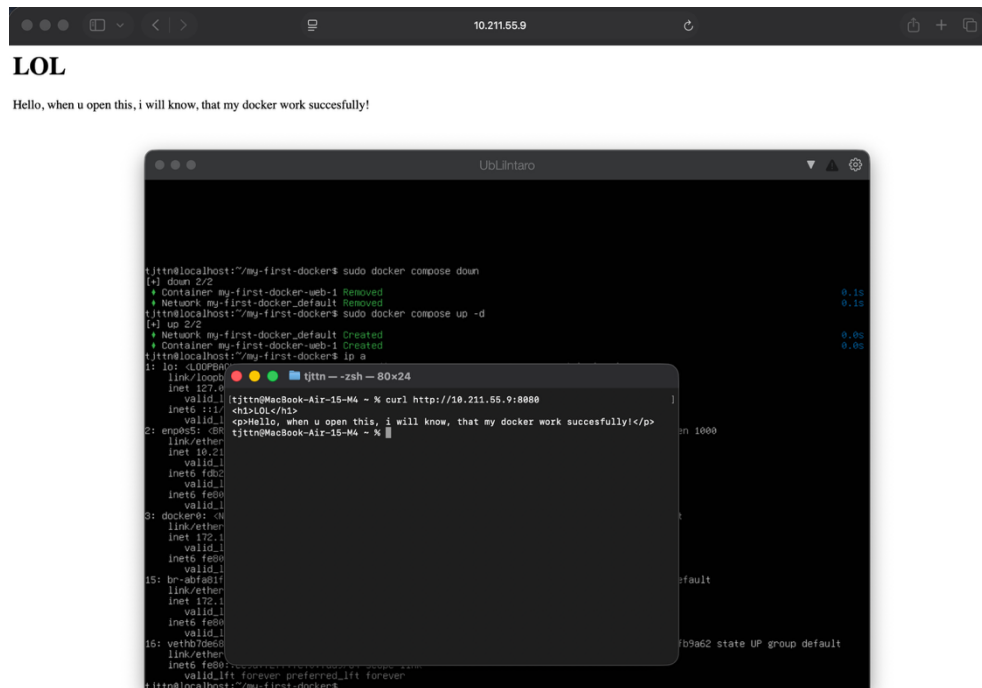


Рисунок 3 – Тестовый запуск

Установка symphony:

Для установки symphony я клонировал репозиторий с github на виртуальную машину – рисунок 4:

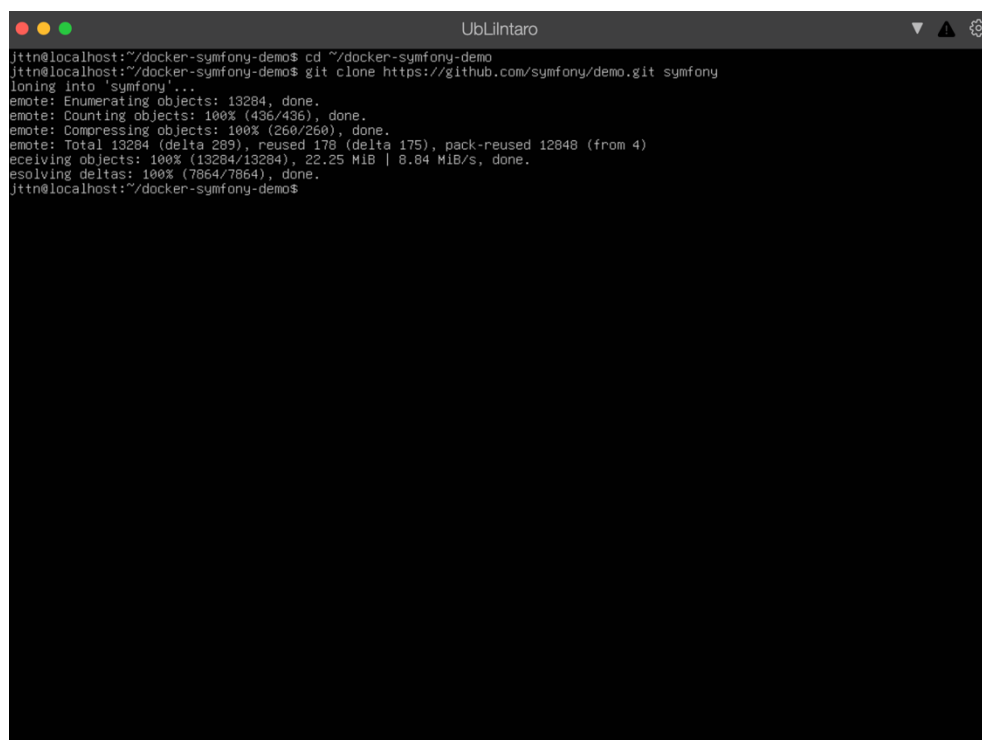


Рисунок 4 – Клонирование репозитория

Далее нужно создать правильную структуру проекта – рисунок 5:

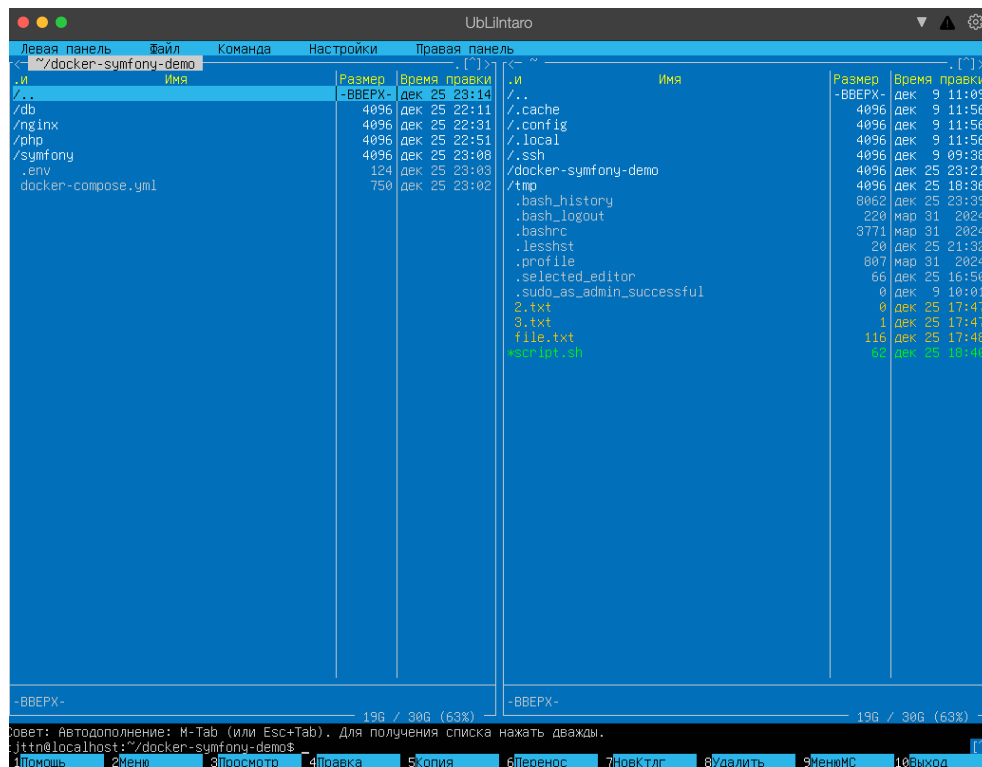


Рисунок 5 – Структура докера

А также нужно написать Dockerfile, который находится в php – для работы с Composer и расширениями:

```
FROM php:8.2-fpm
```

```
RUN apt-get update && apt-get install -y git unzip libpq-dev libicu-dev libzip-dev \
```

```
&& docker-php-ext-install pdo pdo_pgsql intl zip opcache
```

```
COPY --from=composer:2 /usr/bin/composer /usr/bin/composer
```

```
WORKDIR /var/www/symfony
```

В docker-compose.yml содержатся три сервиса:

- Php – symphony и composer;
- Nginx – прокси, монтирует symphony;
- Postgres - бд;

Текст .env файла для PostgreSQL и окружения Symfony выглядит так:

```
POSTGRES_USER=demo

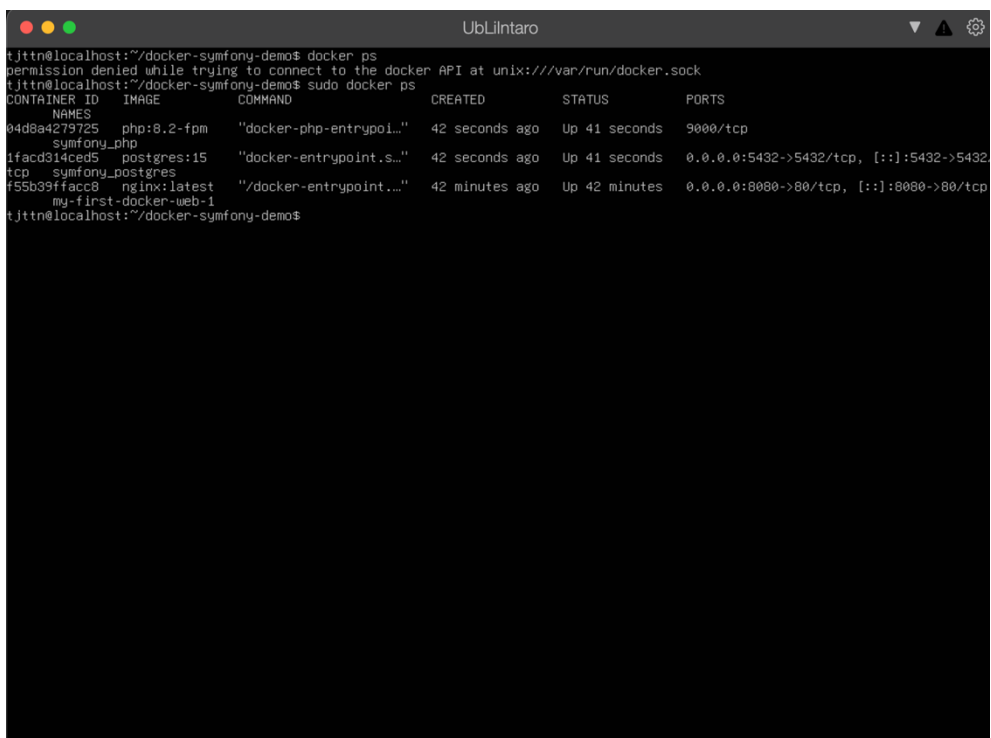
POSTGRES_PASSWORD=demo

POSTGRES_DB=demo

POSTGRES_DATA_PATH=./db

APP_ENV=dev
```

Чтобы удостовериться что всё корректно установлено и написано выполним команду с рисунка 6:



```
tjtt@localhost:~/docker-symfony-demo$ docker ps
permission denied while trying to connect to the docker API at unix:///var/run/docker.sock
tjtt@localhost:~/docker-symfony-demo$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
04d8a4279725   php:8.2-fpm                        "docker-php-entrypoi..." 42 seconds ago Up 41 seconds 9000/tcp
1facd314ced5   postgres:15                        "docker-entrypoint.s..." 42 seconds ago Up 41 seconds 0.0.0.0:5432->5432/tcp
f55b39ffacc8   nginx:latest                       "/docker-entrypoint...." 42 minutes ago Up 42 minutes 0.0.0.0:8000->80/tcp, [::]:8000->80/tcp
my-first-docker-web-1
```

Рисунок 6 – Финальная проверка

Далее последовала сборка контейнеров:

```
sudo docker compose build

sudo docker compose up -d
```

Установка зависимостей Symfony внутри контейнера php выглядит следующим образом:

```
sudo docker exec -it symfony_php bash  
  
composer install  
  
php bin/console doctrine:schema:create  
  
php bin/console doctrine:fixtures:load
```

А результатом будет – рисунок 7:

```
root@0b70648c99a5:/var/www/symfony# 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!  
  
root@0b70648c99a5:/var/www/symfony#
```

Рисунок 7 – Результат установки зависимостей

После чего можно проверять сайт на работоспособность – рисунок 8:

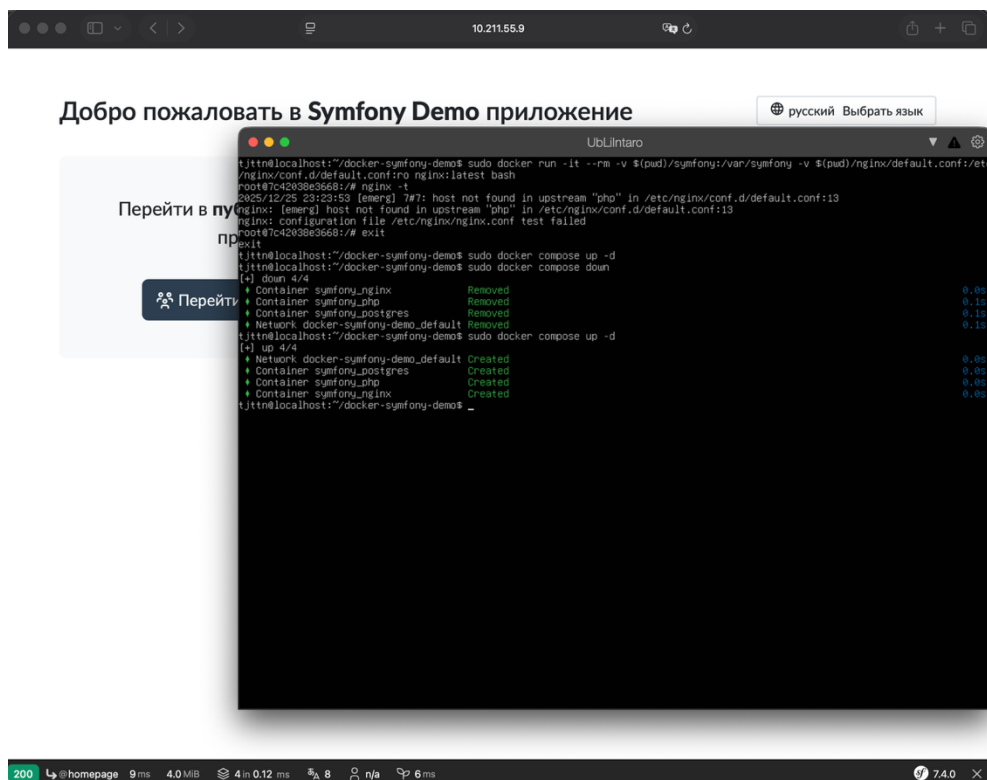


Рисунок 8 – Итог работы



**Вывод:**

В ходе работы я изучил современные методы разработки ПО в динамических и распределенных средах на примере контейнеров Docker.