



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО  
ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт  
Кафедра

компьютерных наук  
автоматизированных систем управления

**ЛАБОРАТОРНАЯ РАБОТА №5**  
по операционным системам Linux  
«Контейнеризация»

Студент

ПИ-22-1

\_\_\_\_\_  
подпись, дата

Пахомов А.А.

Руководитель

\_\_\_\_\_  
подпись, дата

Кургасов В.В.

Липецк, 2024 г.

## **Цель работы**

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

## Ход работы

### 1. Часть I

Для копирования тестового проекта клонируем его с помощью git (рис.1).

```
alex@vbox:~$ git clone https://github.com/symfony/demo
Клонирование в «demo»...
remote: Enumerating objects: 12611, done.
remote: Counting objects: 100% (164/164), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 12611 (delta 54), reused 126 (delta 49), pack-reused 12447 (from 1)
Получение объектов: 100% (12611/12611), 21.99 МБ | 2.82 МБ/с, готово.
Определение изменений: 100% (7514/7514), готово.
alex@vbox:~$ ls
demo
```

Рисунок 1 – Клонирование проекта

Установим Composer с помощью команды `sudo apt install composer` (рис. 2).

```
Composer version 2.5.5 2023-03-21 11:50:05

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is :
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
      --ansi|--no-ansi     Force (or disable --no-ansi) ANSI output
  -n, --no-interaction     Do not ask any interactive question
      --profile            Display timing and memory usage information
      --no-plugins        Whether to disable plugins.
      --no-scripts        Skips the execution of all scripts defined in composer.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working direct
      --no-cache           Prevent use of the cache
  -v|vv|vvv, --verbose    Increase the verbosity of messages: 1 for normal output

Available commands:
  about          Shows a short information about Composer
  archive        Creates an archive of this composer package
  audit          Checks for security vulnerability advisories for installed packag
  browse         [home] Opens the package's repository URL or homepage in your bro
  bump           Increases the lower limit of your composer.json requirements to t
  check-platform-reqs Check that platform requirements are satisfied
  clear-cache    [clearcache|cc] Clears composer's internal package cache
  completion    Dump the shell completion script
  config         Sets config options
  create-project Creates new project from a package into given directory
  depends        [why] Shows which packages cause the given package to be installe
  diagnose       Diagnoses the system to identify common errors
  dump-autoload [dumpautoload] Dumps the autoloader
  exec           Executes a vendored binary/script
  fund           Discover how to help fund the maintenance of your dependencies
  global         Allows running commands in the global composer dir ($COMPOSER_HOM
  help           Display help for a command
  init           Creates a basic composer.json file in current directory
  install        [i] Installs the project dependencies from the composer.lock file
  licenses       Shows information about licenses of dependencies
  list           List commands
  outdated       Shows a list of installed packages that have updates available, i
  prohibits      [why-not] Shows which packages prevent the given package from bei
  reinstall      Uninstalls and reinstalls the given package names
```

Рисунок 2 – Установка Composer

Также нам понадобится утилита curl (рис. 3).

```
alex@vbox:~$ sudo apt install curl
[sudo] пароль для alex:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие НОВЫЕ пакеты будут установлены:
  curl
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Необходимо скачать 315 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 500 kB.
Пол:1 http://deb.debian.org/debian bookworm/main amd64 curl amd64 7.88.1-10+deb12u8 [315 kB]
Получено 315 kB за 5с (67,4 kB/s)
Выбор ранее не выбранного пакета curl.
(Чтение базы данных ... на данный момент установлено 42238 файлов и каталогов.)
Подготовка к распаковке ../curl_7.88.1-10+deb12u8_amd64.deb ...
Распаковывается curl (7.88.1-10+deb12u8) ...
Настраивается пакет curl (7.88.1-10+deb12u8) ...
Обрабатываются триггеры для man-db (2.11.2-2) ...
alex@vbox:~$ curl -sS https://getcomposer.org/installer -o composer-setup.php
alex@vbox:~$ sha384sum composer-setup.php
dac665fdc30fdd8ec78b38b980061b4150413ff2e3b6f88543c636f7cd84f6db9189d43a81e5503cda447da73c7e5b6  composer-setup.php
```

Рисунок 3 – Установка curl

Запустим программу установки пакетного менеджера (рис. 4).

```
alex@vbox:~$ sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
All settings correct for using Composer
Downloading...

Composer (version 2.8.3) successfully installed to: /usr/local/bin/composer
Use it: php /usr/local/bin/composer
```

Рисунок 4 – Установка пакетного менеджера

Выполняем composer install (рис. 5).

```
Your requirements could not be resolved to an installable set of packages.

Problem 1
- Root composer.json requires PHP extension ext-pdo_sqlite * but it is missing from your platform
Problem 2
- Root composer.json requires symfony/html-sanitizer ^7 -> satisfiable by symfony/html-sanitizer[v7.0.0-BETA1, ..., 7.3.x-dev]
- symfony/html-sanitizer[v7.0.0-BETA1, ..., 7.3.x-dev] require ext-dom * -> it is missing from your platform
Problem 3
- Root composer.json requires symfony/debug-bundle ^7 -> satisfiable by symfony/debug-bundle[v7.0.0-BETA1, ..., 7.3.x-dev]
- symfony/debug-bundle[v7.0.0-BETA1, ..., 7.3.x-dev] require ext-xml * -> it is missing from your platform

Alternatively you can require one of these packages that provide the extension (or part of it):
- nphre/php-dbus DBUS bindings for PHP language

Problem 4
- Root composer.json requires symfony/framework-bundle ^7 -> satisfiable by symfony/framework-bundle[v7.0.0-BETA1, ..., 7.3.x-dev]
- symfony/framework-bundle[v7.0.0-BETA1, ..., 7.3.x-dev] require ext-xml * -> it is missing from your platform

Alternatively you can require one of these packages that provide the extension (or part of it):
- nphre/php-dbus DBUS bindings for PHP language

To enable extensions, verify that they are enabled in your .ini files:
- /etc/php/8.2/cli/php.ini
- /etc/php/8.2/cli/conf.d/10-opcache.ini
- /etc/php/8.2/cli/conf.d/10-pdo.ini
- /etc/php/8.2/cli/conf.d/20-calendar.ini
- /etc/php/8.2/cli/conf.d/20-ctype.ini
- /etc/php/8.2/cli/conf.d/20-curl.ini
- /etc/php/8.2/cli/conf.d/20-exif.ini
- /etc/php/8.2/cli/conf.d/20-ffi.ini
- /etc/php/8.2/cli/conf.d/20-fileinfo.ini
- /etc/php/8.2/cli/conf.d/20-ftp.ini
- /etc/php/8.2/cli/conf.d/20-gettext.ini
- /etc/php/8.2/cli/conf.d/20-iconv.ini
- /etc/php/8.2/cli/conf.d/20-intl.ini
- /etc/php/8.2/cli/conf.d/20-mbstring.ini
- /etc/php/8.2/cli/conf.d/20-phar.ini
- /etc/php/8.2/cli/conf.d/20-posix.ini
- /etc/php/8.2/cli/conf.d/20-readline.ini
- /etc/php/8.2/cli/conf.d/20-shmop.ini
- /etc/php/8.2/cli/conf.d/20-sockets.ini
- /etc/php/8.2/cli/conf.d/20-sysvmsg.ini
- /etc/php/8.2/cli/conf.d/20-sysvsem.ini
- /etc/php/8.2/cli/conf.d/20-sysvshm.ini
- /etc/php/8.2/cli/conf.d/20-tokenizer.ini
You can also run `php --ini` in a terminal to see which files are used by PHP in CLI mode.
Alternatively, you can run Composer with `--ignore-platform-req=ext-pdo_sqlite --ignore-platform-req=ext-dom` to ignore these required extensions.
alex@vbox:~/demo$ _
```

Рисунок 5 – Выполнение команды composer install

Теперь очистим кэш (рис. 6).

```
alex@vbox:~/demo$ composer clear-cache
Cache directory does not exist (cache-vcs-dir):
Clearing cache (cache-repo-dir): /home/alex/.cache/composer/repo
Cache directory does not exist (cache-files-dir):
Clearing cache (cache-dir): /home/alex/.cache/composer
All caches cleared.
```

Рисунок 6 – Очистка кэша

Снова повторяем composer install (рис. 7).

```
- Installing twig/extra-bundle (v3.16.0): Extracting archive
- Installing symfony/intl (v7.2.0): Extracting archive
- Installing twig/intl-extra (v3.16.0): Extracting archive
- Installing twig/markdown-extra (v3.16.0): Extracting archive
Generating autoload files
7 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Loading composer repositories with package information
Restricting packages listed in "symfony/symfony" to "7.2.*"
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
7 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
phpstan/extension-installer: Extensions installed
phpstan/phpstan-doctrine: installed
phpstan/phpstan-symfony: installed

Symfony operations: 3 recipes (ddd6d55e2d2387532b7aa6a7e75cf5c4)
- Configuring symfony/framework-bundle (>=7.2): From github.com/symfony/recipes:main
- Configuring symfony/form (>=7.2): From github.com/symfony/recipes:main
- Configuring symfony/ux-icons (>=2.17): From github.com/symfony/recipes:main
Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script importmap:install [OK]
Executing script sass:build [OK]

What's next?

Some files have been created and/or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

symfony/framework-bundle instructions:

* Run your application:
  1. Go to the project directory
  2. Create your code repository with the git init command
  3. Download the Symfony CLI at https://symfony.com/download to install a development web server

* Read the documentation at https://symfony.com/doc
```

Рисунок 7 – Установка composer

Установим Symfony на наш экземпляр ОС (рис. 8).

```
alex@vbox:~/demo$ sudo apt install symfony-cli
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие НОВЫЕ пакеты будут установлены:
  symfony-cli
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Необходимо скачать 5 817 кВ архивов.
После данной операции объём занятого дискового пространства возрастёт на 14,8 МВ.
Пол:1 https://dl.cloudsmith.io/public/symfony/stable/deb/debian bookworm/main amd64 symfony-cli amd64 5.10.5 [5 817
Получено 5 817 кВ за 12с (482 кВ/с)
Выбор ранее не выбранного пакета symfony-cli.
(Чтение базы данных ... на данный момент установлено 42553 файла и каталога.)
Подготовка к распаковке .../symfony-cli_5.10.5_amd64.deb ...
Распаковывается symfony-cli (5.10.5) ...
Настраивается пакет symfony-cli (5.10.5) ...
```

Рисунок 8 – Установка Symfony

Теперь пробросим порт на нашей виртуальной машине (рис. 9).

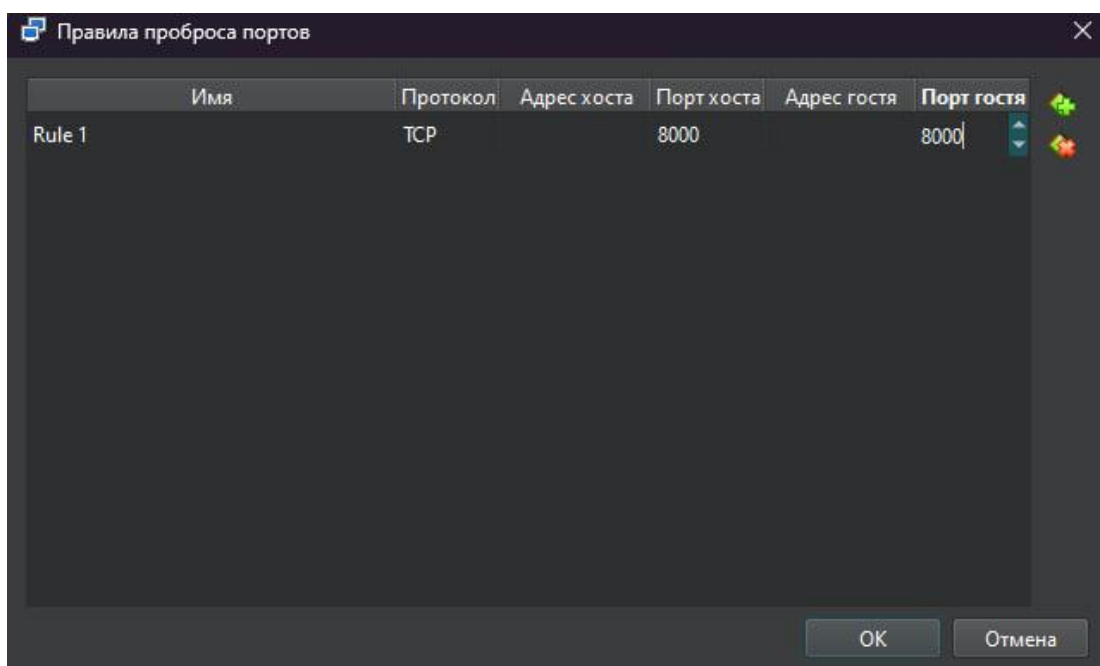


Рисунок 9 – Настройка портов

Запускаем Symfony с помощью команды `symphony --listen-ip=0.0.0.0 serve` и проверяем работу (рис. 10).

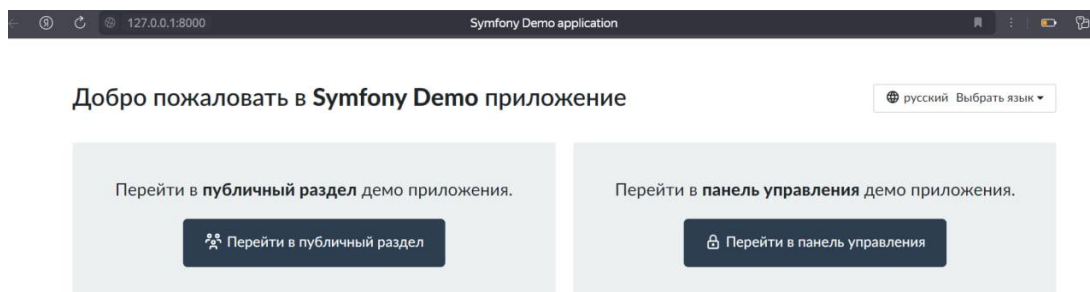


Рисунок 10 – Symfony

С официального сайта Docker возьмем код для установки Docker на ОС (рис. 11).



```

alex@vbox:~$ sudo apt-get install ca-certificates curl
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет ca-certificates самой новой версии (20230311).
Уже установлен пакет curl самой новой версии (7.88.1-10+deb12u8).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено
alex@vbox:~$ sudo install -m 0755 -d /etc/apt/keyrings
alex@vbox:~$ sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
alex@vbox:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
alex@vbox:~$ echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian bookworm InRelease" | \
> $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
> sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
alex@vbox:~$ sudo docker
sudo: docker: command not found
alex@vbox:~$ sudo apt-get update
Пол:1 http://security.debian.org/debian-security bookworm-security InRelease
Пол:2 https://download.docker.com/linux/debian bookworm InRelease [43,3 kB]
Пол:3 http://deb.debian.org/debian bookworm InRelease
Пол:4 https://dl.cloudsmith.io/public/symfony/stable/deb/debian bookworm InRelease [4 422 B]
Пол:5 http://deb.debian.org/debian bookworm-updates InRelease
Пол:6 https://download.docker.com/linux/debian bookworm/stable amd64 Packages [32,4 kB]
Получено 80,2 kB за 3с (30,8 kB/с)
Чтение списков пакетов... Готово
alex@vbox:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Будут установлены следующие дополнительные пакеты:
  docker-ce-rootless-extras iptables libip6tc2 libltdl7 libnetfilter-conntrack3 libnftnl libnftnl0 libslirp0
Предлагаемые пакеты:
  aufs-tools cgroupfs-mount | cgroup-lite firewallld
Следующие НОВЫЕ пакеты будут установлены:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin
  libnetfilter-conntrack3 libnftnl libnftnl0 libslirp0 pigz slirp4netns
Обновлено 0 пакетов, установлено 14 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено
Необходимо скачать 125 МБ архивов.
После данной операции объём занятого дискового пространства возрастёт на 448 МБ.
Хотите продолжить? [Д/Н] y
Пол:1 http://deb.debian.org/debian bookworm/main amd64 pigz amd64 2.6-1 [64,0 kB]
Пол:2 https://download.docker.com/linux/debian bookworm/stable amd64 containerd.io amd64 1.7.24-1 [29,5 MB]
Пол:3 http://deb.debian.org/debian bookworm/main amd64 libip6tc2 amd64 1.8.9-2 [19,4 kB]
Пол:4 http://deb.debian.org/debian bookworm/main amd64 libnftnl0 amd64 1.0.2-2 [15,1 kB]
Пол:5 http://deb.debian.org/debian bookworm/main amd64 libnetfilter-conntrack3 amd64 1.0.9-3 [40,7 kB]
Пол:6 http://deb.debian.org/debian bookworm/main amd64 iptables amd64 1.8.9-2 [360 kB]
Пол:7 http://deb.debian.org/debian bookworm/main amd64 libltdl7 amd64 2.4.7-7~deb12u1 [393 kB]
Пол:8 http://deb.debian.org/debian bookworm/main amd64 libslirp0 amd64 4.7.0-1 [63,0 kB]
Пол:9 http://deb.debian.org/debian bookworm/main amd64 slirp4netns amd64 1.2.0-1 [37,5 kB]
13% [2 containerd.io 1 114 kB/29,5 MB 4%]

```

Рисунок 11 – Установка Docker

В файле `composer.json` убираем `pdo_sqlite3` и устанавливаем `Postgresql` (рис. 12).

```

alex@vbox:~/demo$ sudo apt install postgresql
[sudo] пароль для alex:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Будут установлены следующие дополнительные пакеты:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5 libsodium23 libssl3
  postgresql-client-15 postgresql-client-common postgresql-common sysstat
Предлагаемые пакеты:
  lm-sensors postgresql-doc postgresql-doc-15 isag
Следующие НОВЫЕ пакеты будут установлены:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5 libsodium23 libssl3
  postgresql-client-15 postgresql-client-common postgresql-common sysstat
Обновлено 0 пакетов, установлено 15 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено
Необходимо скачать 48,9 МБ архивов.

```

Рисунок 12 – Установка postgresql

Для добавления новых зависимостей изменим переменную `DATABASE_URL` (рис. 13).

```

GNU nano 7.2
# 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 >=1.2).
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-configuration
###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=
###< 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="mysql://app:lcHangeMe!@127.0.0.1:3306/app?serverVersion=8&charset=utf8mb4"
DATABASE_URL="postgresql://postgres:postgres@postgres:5432/app?serverVersion=17.1&charset=utf8"
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###
# MAILER_DSN=null://null
###< symfony/mailer ###

```

Рисунок 13 – Переменная DATABASE\_URL

Теперь создаем файл Dockerfile в нашем проекте (рис. 14).

```

GNU nano 7.2
FROM wyved/nginx-php-fpm:php82
WORKDIR /var/www/html/demo
COPY composer.json ./
COPY . .
RUN \
    wget https://get.symfony.com/cli/installer -O - | bash \
    && mv /root/.symfony5/bin/symfony /usr/local/bin/symfony \
    && symfony composer install
EXPOSE 8000
CMD ["symfony", "--listen-ip=0.0.0.0", "serve"]

```

Рисунок 14 – Dockerfile

В той же папке создаем файл docker-compose.yml:



```

GNU nano 7.2
version: "3"
services:
  app:
    container_name: docker-node-mongo
    restart: always
    build: .
    ports:
      - "81:8000"
    links:
      - postgres
  postgres:
    container_name: postgres
    image: postgres
    ports:
      - "5432:5432"
    environment:
      POSTGRES_PASSWORD: postgres
    volumes:
      - pgdata:/var/lib/postgresql/data
volumes:
  pgdata:

```

Рисунок 15 – Файл docker-compose.yml

Меняем порт и запускаем Docker с помощью команды `docker compose up` (рис. 16).

```

alex@vbox:~/demo$ sudo docker compose up --build
WARN[0000] /home/alex/demo/docker-compose.yml: the attribute `version` is obsolet
[+] Running 15/15
  postgres Pulled
  bc0965b23a04 Pull complete
  002e1a8eb6f9 Pull complete
  a24f300391ed Pull complete
  627f580b7ad7 Pull complete
  cfb3c2203f88 Pull complete
  9e592465b243 Pull complete
  8d4265d09d9c Pull complete
  e3a8293e92fd Pull complete
  2cb801c39436 Pull complete
  c5fdb20d8658 Pull complete
  67c5fe618f0c Pull complete
  c9cdd1fe82e4 Pull complete
  8f152c4aceed Pull complete
  2cd360f3b7db Pull complete
[+] Building 28.3s (3/9)
=> [app internal] load build definition from Dockerfile
=> => transferring dockerfile: 351B
=> [app internal] load metadata for docker.io/wyveo/nginx-php-fpm:php82
=> [app internal] load .dockerignore
=> => transferring context: 2B
=> [app 1/5] FROM docker.io/wyveo/nginx-php-fpm:php82@sha256:37ef8b959f555ac4e
=> => resolve docker.io/wyveo/nginx-php-fpm:php82@sha256:37ef8b959f555ac4eb5fd
=> => sha256:37ef8b959f555ac4eb5fd34200c60fb978f361c1cfd8454fa0531ece2a8c2117
=> => sha256:570c8f81c445b2fc3954fbd0a964d7a281f1c03f931686d8b7631879bf4a1ec4
=> => sha256:7dbc1adf280e1aa588c033eaa746aa6db327ee16be705740f81741f5e6945c86
=> => sha256:9403cbe0152d96680300bf6bc4784541d4631278badb5e3dc111444beffda751
=> => sha256:a134f3675ea26449f84e5708ab882d95d1a5e39e8633ad2b175fbae17fbdebe7
=> => sha256:3e4c46da5536c182e618d6ab999b05abc5f0f52040a65ca21b59bb5ebac6642
=> => sha256:2d2e58a4006fd71101daf52b7e71606822649ea632d796da42069d3d5c9ee637
=> => sha256:a223309ce8ca45189acbd9aee232324b6b14ecfcf6aad85fc11a0f2e870f61c
=> [app internal] load build context
=> => transferring context: 141.88MB
[+] Running 0/1al load build context
  Service app Building

```

Рисунок 16 – Запуск Docker

Теперь заходим в запущенное приложение и входим в admin (рис. 17).

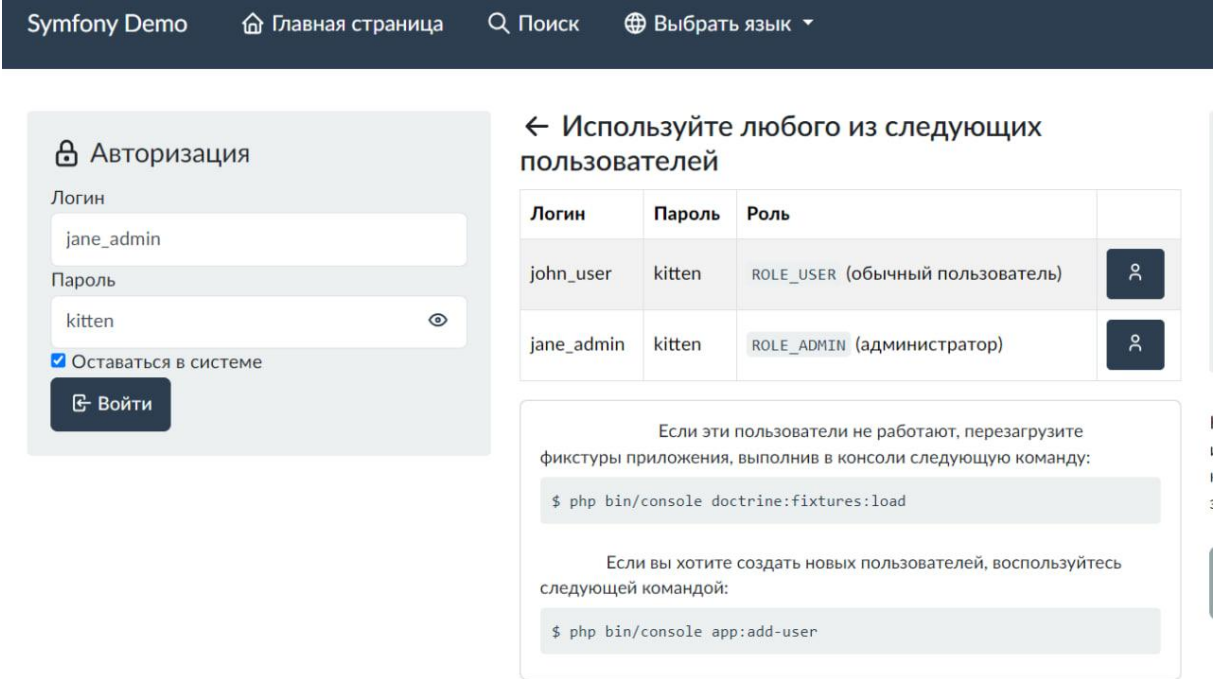
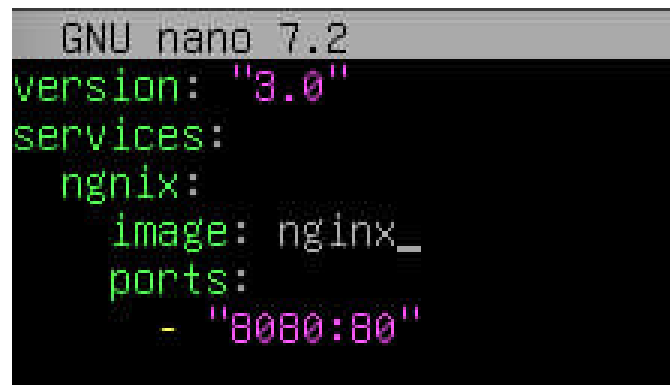


Рисунок 17 – Проверка работы запущенного контейнера

## Часть II

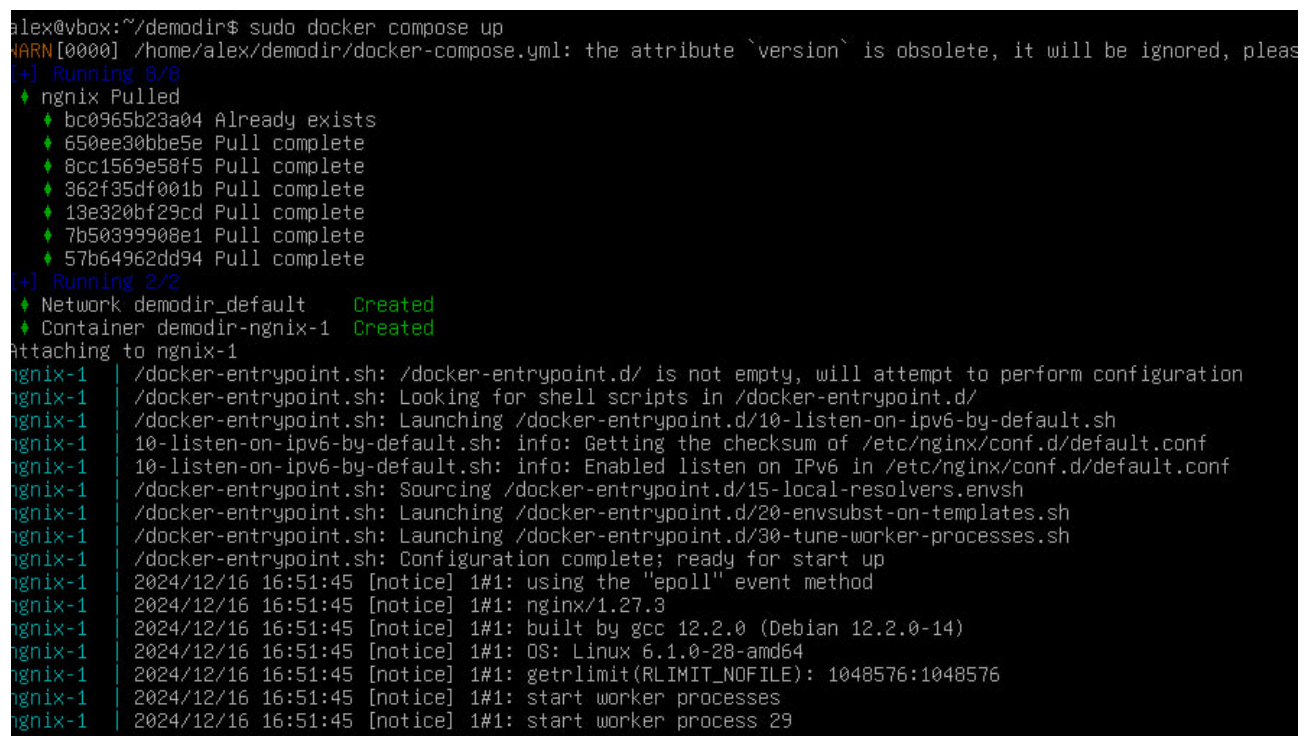
Создаем новую директорию и в ней создаем файл `docker-compose.yml` (рис. 18).



```
GNU nano 7.2
version: "3.0"
services:
  nginx:
    image: nginx_
    ports:
      - "8080:80"
```

Рисунок 18 – Файл `docker-compose.yml`

Запускаем `docker` (рис. 19) и заходим на сайт для проверки контейнера (рис. 20).



```
alex@vbox:~/demodir$ sudo docker compose up
WARN[0000] /home/alex/demodir/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please
[+] Running 8/8
  * nginx Pulled
  * bc0965b23a04 Already exists
  * 650ee30bbe5e Pull complete
  * 8cc1569e58f5 Pull complete
  * 362f95df001b Pull complete
  * 13e320bf29cd Pull complete
  * 7b50399908e1 Pull complete
  * 57b64962dd94 Pull complete
[+] Running 2/2
  * Network demodir_default Created
  * Container demodir-nginx-1 Created
Attaching to nginx-1
nginx-1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx-1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx-1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx-1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginx-1 | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx-1 | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: using the "epoll" event method
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: nginx/1.27.3
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: OS: Linux 6.1.0-28-amd64
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: start worker processes
nginx-1 | 2024/12/16 16:51:45 [notice] 1#1: start worker process 29
```

Рисунок 19 – Запуск контейнера

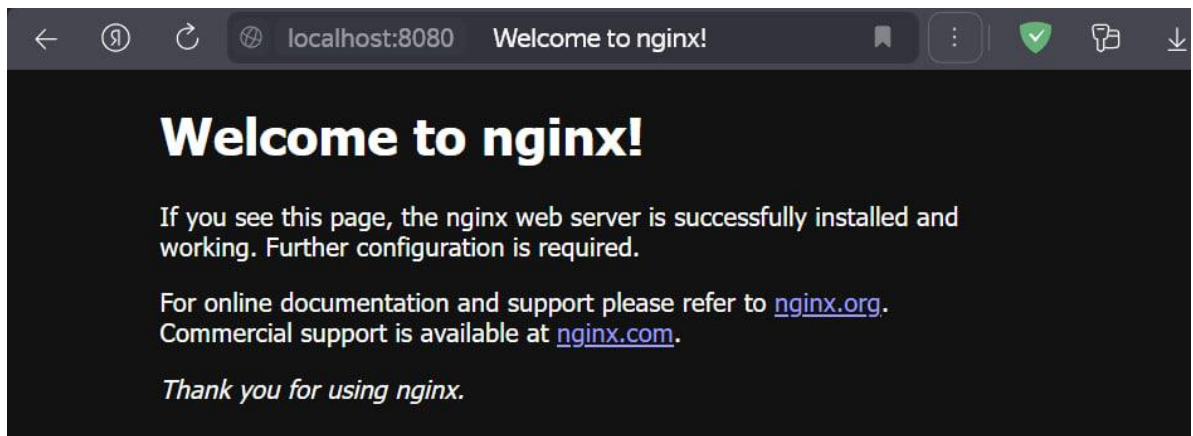


Рисунок 20 – nginx

Создаем в той же папке директорию html. В ней создаем файл index.html с сообщением и редактируем ранее созданный docker-compose.yml (рис. 21).

```
GNU nano 7.2
version: "3.0"
services:
  nginx:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - type: bind
        source: ./html
        target: /usr/share/nginx/html_
```

Рисунок 21 – Редактирование docker-compose.yml

Заново запускаем контейнер (рис. 22).

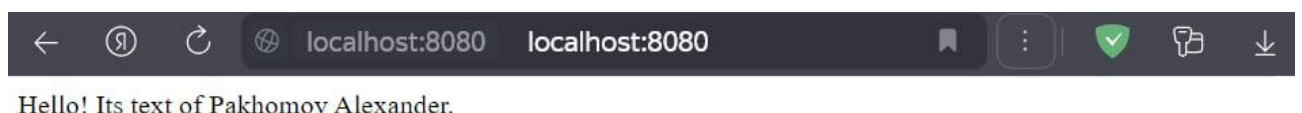


Рисунок 22 – Проверка работы контейнера

Создаем все необходимые директории и файлы в соответствии с заданием. Настраиваем сетевой мост в виртуальной машине и в папке проху создаем файл docker-compose.yml (рис. 23).

```

version: '3.0'
services:
  proxy:
    image: jwilder/nginx-proxy
    ports:
      - "80:80"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock:ro
    networks:
      - proxy
networks:
  proxy:
    driver: bridge

```

Рисунок 23 – Файл docker-compose.yml

Запускаем docker и наблюдаем результаты (рис. 24).

```

alex@vbox:~$ sudo docker network ls
[sudo] пароль для alex:
NETWORK ID          NAME                DRIVER              SCOPE
f3c9f0653242        bridge              bridge              local
6584a31bc980        demo_default        bridge              local
2094664f627b        demodir_default     bridge              local
05cacd289418        host                host                local
5dbb24c61d96        none                null                local
82aa2610128b        proxy_proxy         bridge              local

```

Рисунок 24 – docker network

В главной папке изменяем docker-compose.yml (рис. 25).

```

GNU nano 7.2                                                                 docker
nginx:
  image: nginx
  environment:
    VIRTUAL_HOST: site.local
  depends_on:
    - php
  ports:
    - "8081:80"
  volumes:
    - ./docker/nginx/conf.d/default.nginx:/etc/nginx/conf.d/default.conf
    - ./html:/var/www/html/
  networks:
    - frontend
    - backend
php:
  build:
    context: ./docker/php
  volumes:
    - ./docker/php/php.ini:/usr/local/etc/php/php.ini
    - ./html:/var/www/html/
  networks:
    - backend
mysql:
  image: mysql:5.7
  volumes:
    - ./docker/mysql/data:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: root
  networks:
    - backend
phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  environment:
    VIRTUAL_HOST: phpmyadmin.local
    PMA_HOST: mysql
    PMA_USER: root
    PMA_PASSWORD: root
  ports:
    - "8082:80"
  networks:
    - frontend
networks:
  frontend:
    external:
      name: proxy_proxy
  backend:

```

Рисунок 25 – Изменение файла docker-compose.yml

Добавляем файл конфигурации default.nginx для nginx (рис. 26).



```
server {
    listen: 80;
    server_name_in_redirect off;
    access_log /var/log/nginx/host.access.log main;
    root /var/www/html;

    location / {
        try_files $uri /index.php$is_args$args;
    }
    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

Рисунок 26 – Файл конфигурации

В папке php создаем Dockerfile для php (рис. 27).

```
FROM php:fpm

RUN apt-get update && apt-get install -y libzip-dev zip
RUN docker-php-ext-configure zip
RUN docker-php-ext-install zip
RUN docker-php-ext-install mysqli

COPY --from=composer:latest /usr/bin/composer /usr/bin/composer_
WORKDIR /var/www/html
```

Рисунок 27 – Dockerfile для php

Создаем index.php в директории html.

```
<?php

$link=mysqli_connect('mysql', 'root', 'root'); if
(! $link) {
    die('Ошибка соединения: ' .mysqli_error());
}
echo 'Успешно соединились';
mysqli_close($link);_
```

Рисунок 28 – Файл index.php

Запускаем docker (рис. 29) и проверяем результаты (рис.30).

```
Run 'docker compose COMMAND --help' for more information on a command.
alex@vbox:~/demodir$ sudo docker compose up
WARN[0000] /home/alex/demodir/docker-compose.yml: the attribute `version` is obsolete, it will be i
WARN[0000] networks.frontend: external.name is deprecated. Please set name and external: true
[+] Running 1/31
  ⬇ mysql [+++++ ] Pulling
    ⬇ 20e4dcae4c69 Waiting
    ⬇ 1c56c3d4ce74 Waiting
    ⬇ e9f03a1c24ce Waiting
    ⬇ 68c3898c2015 Waiting
    ⬇ 6b95a940e7b6 Waiting
    ⬇ 90986bb8de6e Waiting
    ⬇ ae71319cb779 Waiting
    ⬇ ffc89e9dfd88 Waiting
    ⬇ 43d05e938198 Waiting
    ⬇ 064b2d298fba Waiting
    ⬇ df9a4d85569b Waiting
  ⬇ phpmyadmin [+++++ ] Pulling
    ⬇ faef57eae888 Pulling fs layer
    ⬆ 989a1d6c052e Download complete
    ⬇ 0705c9c2f22d Pulling fs layer
    ⬇ 621478e043ce Waiting
    ⬇ 98246dcca987 Waiting
    ⬇ bfe8c155cb6 Waiting
    ⬇ 7a7c2e908867 Waiting
    ⬇ d176994b625c Waiting
    ⬇ 2d8ace6a2716 Waiting
    ⬇ c70df516383c Waiting
    ⬇ 15e1b44fe4c7 Waiting
    ⬇ 65e50d44e95a Waiting
    ⬇ 77f68910bc0a Waiting
    ⬇ 605dd3a6e332 Waiting
    ⬇ 99ce27188f07 Waiting
    ⬇ 74d64e32c5d5 Waiting
    ⬇ ef5fc9928b9f Waiting
    ⬇ 163f3256e112 Waiting
```

Рисунок 29 – Запуск контейнера

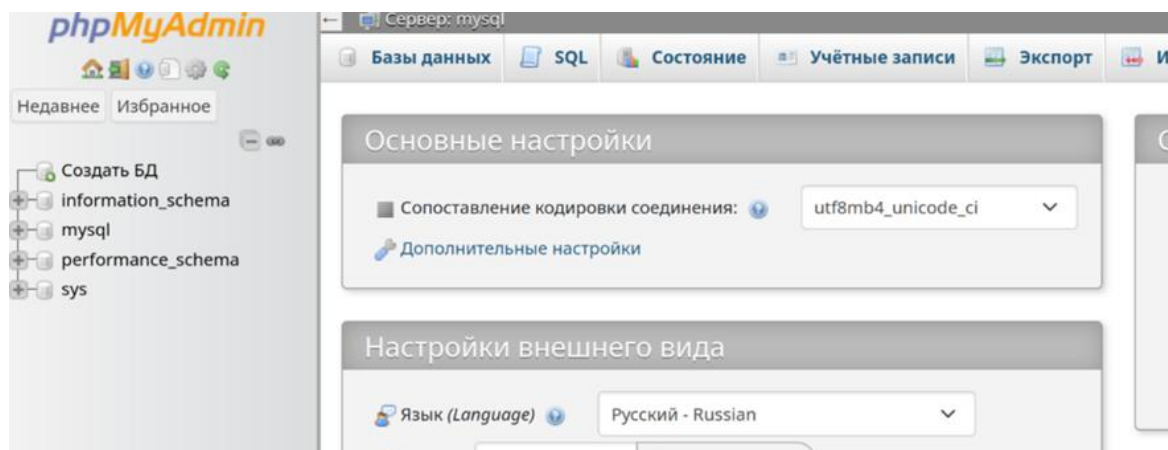


Рисунок 30 – Проверка результатов

Для создания образа с движком WordPress добавляем сервис в корневом docker-compose.yml (рис. 31).

```

wordpress:
  image: wordpress:latest
  environment:
    WORDPRESS_DB_HOST: mysql:3306
    WORDPRESS_DB_USER: root
    WORDPRESS_DB_PASSWORD: root
    WORDPRESS_DB_NAME: wordpress
  depends_on:
    - mysql
  volumes:
    - ./html/courses_data:/var/www/html/wp-content/uploads/courses_data
  ports:
    - "8083:80"
  networks:
    - frontend
    - backend
networks:

```

Рисунок 31 – Добавление WordPress

На рисунке 32 показано, что произошло успешное добавление этого сервиса.

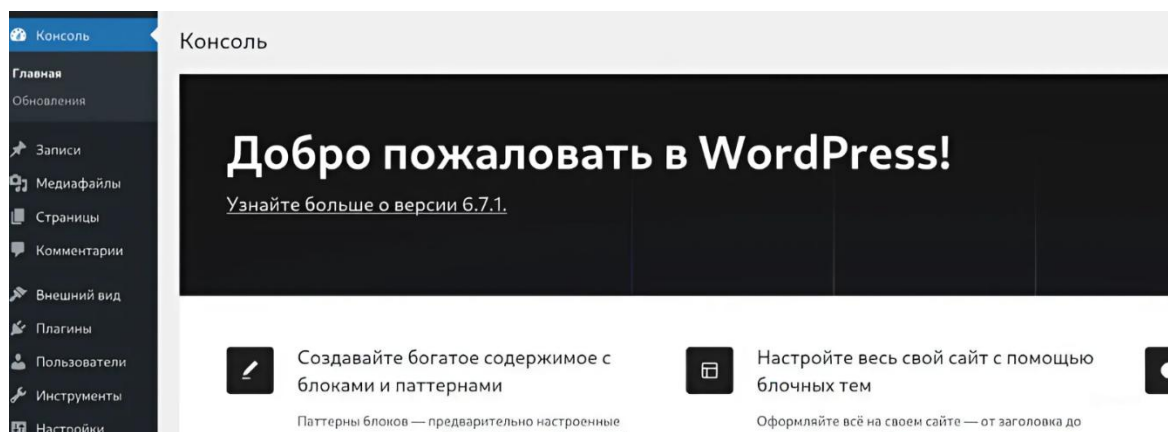


Рисунок 32 – Проверка работы контейнера

## **Контрольные вопросы**

**1. Назовите отличия использования контейнеров по сравнению с виртуализацией.**

Меньшие накладные расходы на инфраструктуру и невозможность запуска GNU/Linux- и Windows-приложений на одном хосте.

**2. Назовите основные компоненты Docker.**

Контейнеры и реестры.

**3. Какие технологии используются для работы с контейнерами?**

Пространства имен (Linux Namespaces) и контрольные группы (cgroups)

**4. Найдите 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 (репозитории) – сетевые хранилища образов.

**5. В чем отличие контейнеров от виртуализации?**

Контейнеры экономят ресурсы, быстрее работают и удобны для разработчиков. Виртуализация предоставляет полную изоляцию на уровне операционных систем, но требует больше ресурсов.

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

`docker run` - Запускает контейнер на основе указанного образа.

`docker build` - Создает образ на основе Dockerfile.

`docker pull` - Загружает образ из удаленного реестра (например, Docker Hub).

`docker push` - Загружает локальный образ в удаленный реестр.

`docker ps` - Показывает список работающих контейнеров.

`docker ps -a` - Показывает все контейнеры, включая остановленные.

`docker stop` - Останавливает работающий контейнер.

`docker start` - Запускает ранее остановленный контейнер.

`docker rm` - Удаляет контейнер.

`docker rmi` - Удаляет образ.

`docker logs` - Показывает логи контейнера.

`docker exec` - Выполняет команду внутри работающего контейнера.

`docker images` - Показывает список локально сохраненных образов.

`docker inspect` - Показывает детальную информацию о контейнере или образе.

## **7. Каким образом осуществляется поиск образов контейнеров?**

Поиск образов контейнеров осуществляется через репозитории контейнеров, например, Docker Hub, GitHub Container Registry или приватные регистры. Для этого используется команда `docker search <имя образа>`.

## **8. Каким образом осуществляется запуск контейнера?**

Запуск контейнера выполняется с использованием команды `docker run`, которая принимает параметры образа, флаги конфигурации и аргументы.

## **9. Что значит управлять состоянием контейнеров?**

Управление состоянием контейнеров подразумевает выполнение следующих операций:

- Запуск остановленных контейнеров.
- Временное прекращение работы контейнера.
- Перезапуск контейнера для применения изменений или устранения сбоев.
- Завершение и очистка контейнера, если он больше не нужен.
- Отслеживание статуса, загрузки ресурсов и логов.

## **10. Как изолировать контейнер?**

Контейнеры в Docker изолированы по умолчанию. Для обеспечения изоляции используются:

- Namespaces: изоляция процессов, сетевых интерфейсов и файловой системы.
- Control Groups (cgroups): управление ресурсами (ЦПУ, память, диски).
- Сетевые мосты: ограничение взаимодействия между контейнерами.

- Флаги запуска, с помощью которых можно дополнительно ограничивать права контейнера.

### **11. Опишите последовательность создания новых образов, назначение Dockerfile?**

Dockerfile — это текстовый файл с инструкциями, описывающими процесс создания образа. Основные директивы:

- FROM: базовый образ.
- RUN: выполнение команды при сборке.
- COPY / ADD: копирование файлов в образ.
- CMD / ENTRYPOINT: команды для выполнения при запуске контейнера.
- EXPOSE: указание открытых портов.

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

Нет, работать с контейнерами Docker без движка Docker невозможно, так как именно этот компонент отвечает за выполнение всех операций с контейнерами. Однако существуют альтернативные платформы для работы с контейнерами, которые не требуют установки Docker Engine и обеспечивают совместимость с образами Docker.

### **13. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes.**

Kubernetes — это система оркестрации контейнеров, предназначенная для:

- Автоматизации развертывания, управления и масштабирования контейнеризованных приложений.
- Обеспечения высокой доступности и отказоустойчивости.
- Управления сетевыми взаимодействиями между контейнерами.

Основные объекты Kubernetes:

- Pod – минимальная единица развертывания, может содержать один или несколько контейнеров.
- Node – рабочий узел, где запускаются Pods.



- Deployment управляет созданием и обновлением Pods.
- Service обеспечивает доступ к Pod через стабильный IP или DNS-имя.
- Namespace – логическая группировка ресурсов.
- ConfigMap необходимо для хранения конфигурационных данных.
- Secret необходимо для хранения конфиденциальных данных (пароли, ключи).