МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ



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

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №5

по курсу “ОС Linux”

Студент ПИ-21-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Красиков И. А.

(подпись, дата)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кургасов В.В.

(подпись, дата)

Липецк 2023

[**Цель работы** 3](#_Toc1)

[**Задание**  3](#_Toc2)

[**I часть** 5](#_Toc3)

[**Установка необходимых инструментов** 5](#_Toc4)

[**Выполнение работы** 9](#_Toc5)

[**II часть** 18](#_Toc6)

[**Шаг №1. Установка Nginx** 18](#_Toc7)

[**Шаг №2. Передача в контейнер html-файлов** 19](#_Toc8)

[**Шаг №3. Web-разработка** 20](#_Toc9)

[**Шаг №4. Запуск Wordpress.** 25](#_Toc10)

[**Вопросы для самопроверки** 28](#_Toc11)

# **Цель работы**

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

# **Задание**

**I часть**

С помощью Docker Compose на своем компьютере поднять сборку nginx+phpfpm+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.

**II часть**

Шаг №1. Установка Nginx Для начала необходимо установить один лишь Nginx. Что требует создания compose-файла включая директиву ports, иначе порт будет доступен только внутри контейнера и nginx через браузер уже будет недоступен.

Шаг №2. Передача в контейнер html-файлов. В этом нам поможет volumes, которая говорит, что происходит монтирование локальной папки в контейнер по указанному адресу. При монтировании папка по указанному адресу внутри контейнера заменяется папкой с локального компьютера. Необходимо создать папку html на одном уровне с docker-compose.yml и добавить в нее файл index.html с произвольным текстом «Ваш текст», после чего пересоздадим контейнер (docker-compose up -d).

Шаг 3. Web-разработка. Создать папку proxy и в ней сборку dockercompose.yml для обращения по домену и пробросу такого домена на основной контейнер. И сборку nginx, php, mysql и phpmyadmin с использованием proxy сети.

Шаг 4. Имеется работающий Web-сервер. Создайте образ с одним из движков (WordPress, Joomla). Папка для хранения внешних данных с курсами должна быть Вами определена.

# **I часть**

## **Установка необходимых инструментов**

**Установка Docker и docker compose:**

Установка пакетов позволяющих apt использовать пакеты через HTTPS

sudo apt install apt-transport-https ca-certificates curl software-properties-common

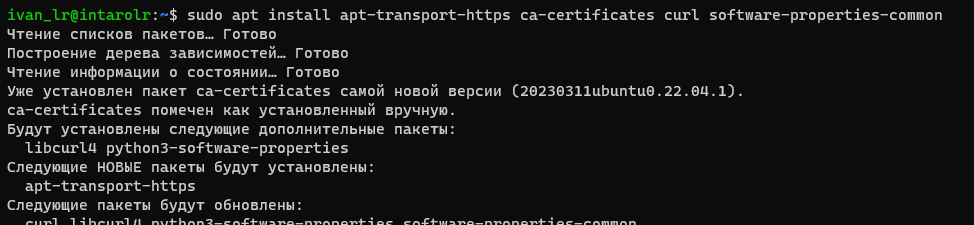


Рис 1 – Установка пакетов для доступа к пакетам через HTTPS

Добавляем ключ GPG для официального репозитория Docker в нашу систему

sudo apt install apt-transport-https ca-certificates curl software-properties-common

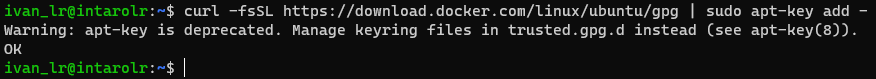


Рис 2 – Добавление ключа GPG

Добавляем репозиторий Docker в источник APT

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

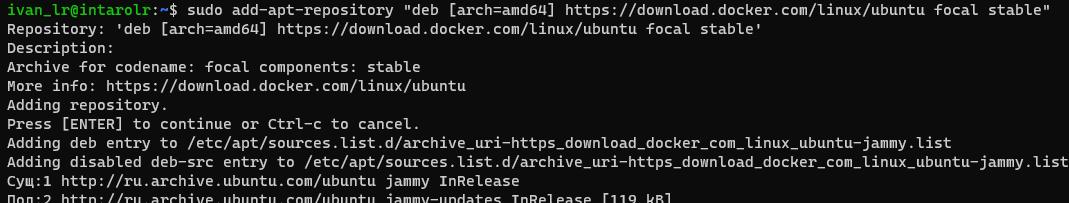


Рис 3 – Добавление репозитория Docker в источник APT

Установка Docker

sudo apt-get install docker-ce

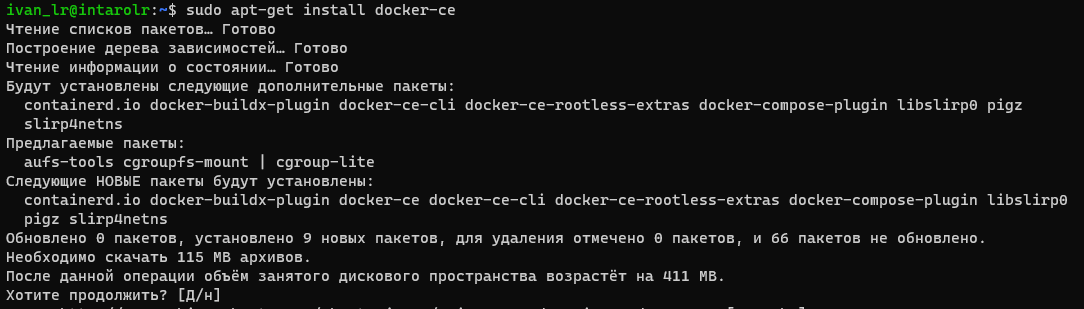


Рис 3 – Установка Docker

Настройка команды docker без sudo

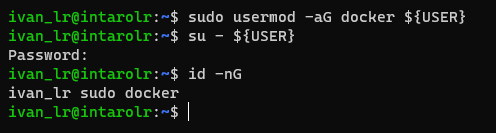


Рис 4 – Docker без sudo

Установка Docker-compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

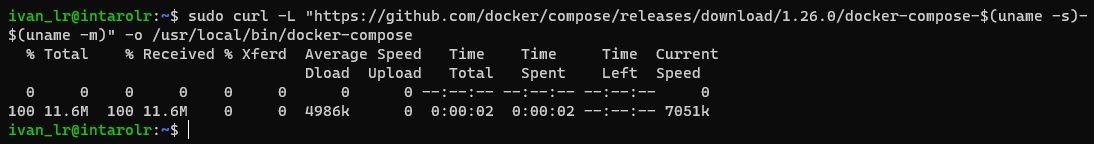


Рис 5 – Установка Docker-compose 1 этап

sudo chmod +x /usr/local/bin/docker-compose



Рис 6 – Установка Docker-compose 2 этап

Проверка успешности установки

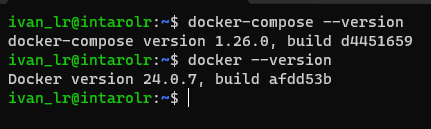


Рис 6 – Успешность установки

**Установка PHP**

sudo apt install php8.1

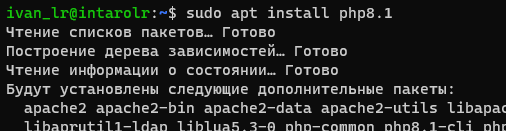


Рис 7 – Установка php

sudo apt install php8.1-mbstring php-sqlite3 php8.1-pgsql php8.1-xml

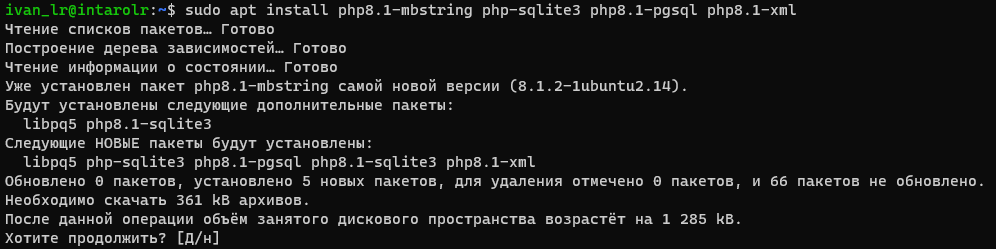


Рис 8 – Установка дополнительных пакетов для php

**Установка Postgresql**

sudo apt install postgresql postgresql-contrib

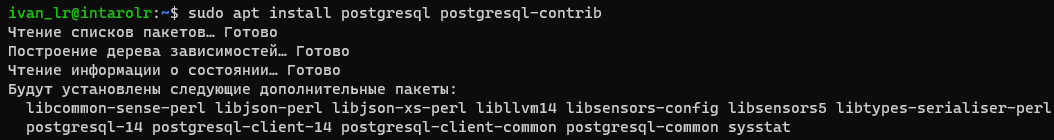


Рис 9 – Установка Postgresql

**Установка Composer**

sudo apt install php-cli unzip

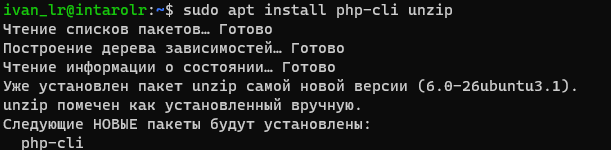


Рис 10 – Установка Composer 1 этап

curl -sS https://getcomposer.org/installer -o composer-setup.php

php -r "if (hash\_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"

sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer

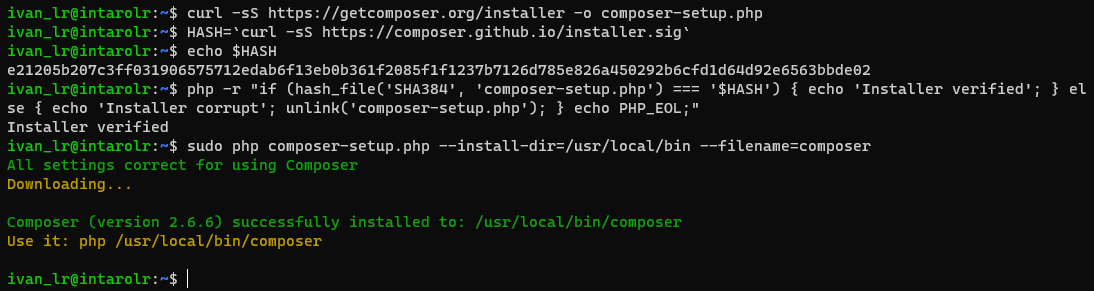


Рис 11 – Установка Composer 2 этап



Рис 12 – Проверка установки

**Установка Symfony**

sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony

sudo chmod a+x /usr/local/bin/symfony

## **Выполнение работы**

**1)** Клонируем тестовый проект

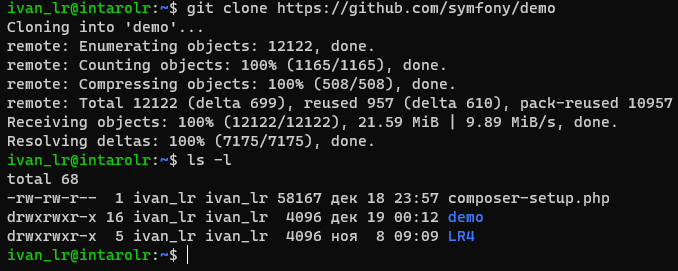


Рис 13 – Клонирование тестового проекта

**2)** Переходим в папку с проектом

С помощью команды cd demo

**3)** Запустим тестовый проект

Но перед тем как запустить проект нам нужно настроить проброс портов в VirtualBox, чтобы мы могли увидеть тестовый сайт

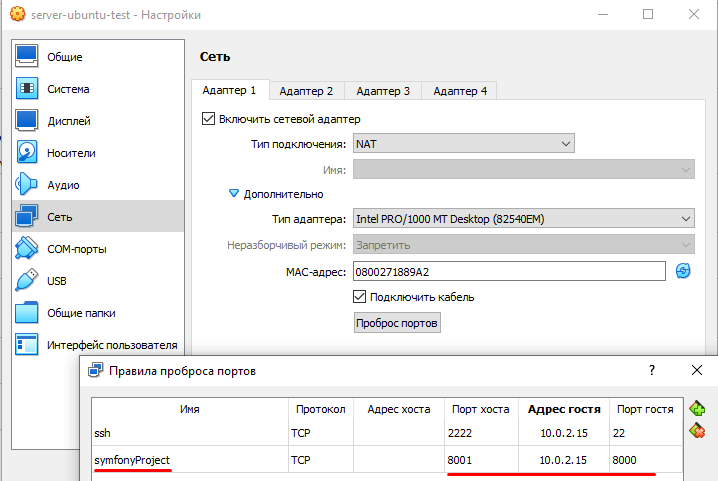


Рис 14 – Проброс портов

С помощью команды symfony serve:start запускаем наш сервер

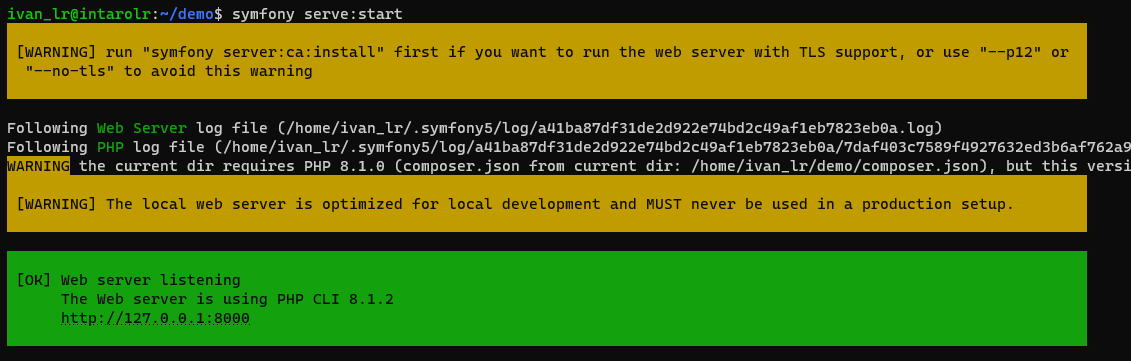


Рис 15 – Запуск сервера

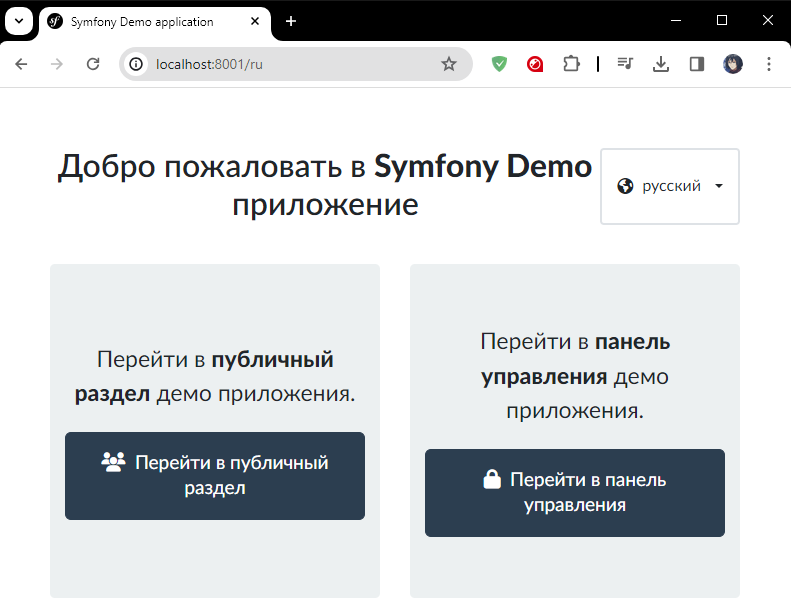


Рис 16 – Наш запущенный сайт

**5)** Docker и Docker-compose установили

**5.1)** Настройка Postgres

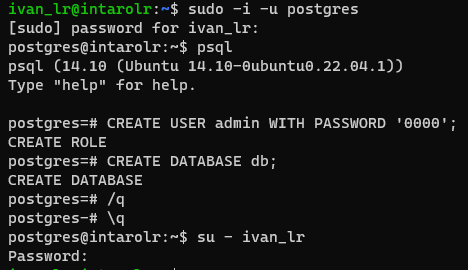


Рис 17 – Создание БД



Рис 18 – Замена DATABASE\_URL

**6)** Создадим файл docker-compose.yml и файл Dockerfile для php-fpm

Вместе с созданием файлов для docker создадим папки для конфига nginx и для dockerfile php-fpm, а также папку с нашим демо-проектом

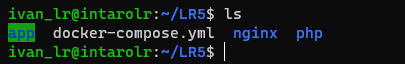


Рис 19 – Структура проекта

В папке nginx – хранится конфигурация сервера

В папке php – хранится Dockerfile

В папке app – хранится наш сайт

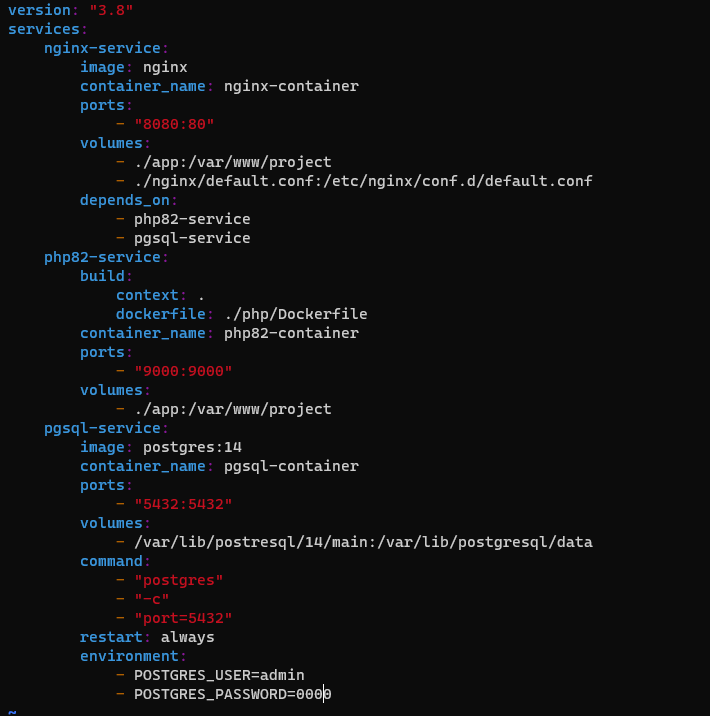


Рисунок 20 – файл docker-compose.yml

Рисунок 21 – Dockerfile для php-fpm

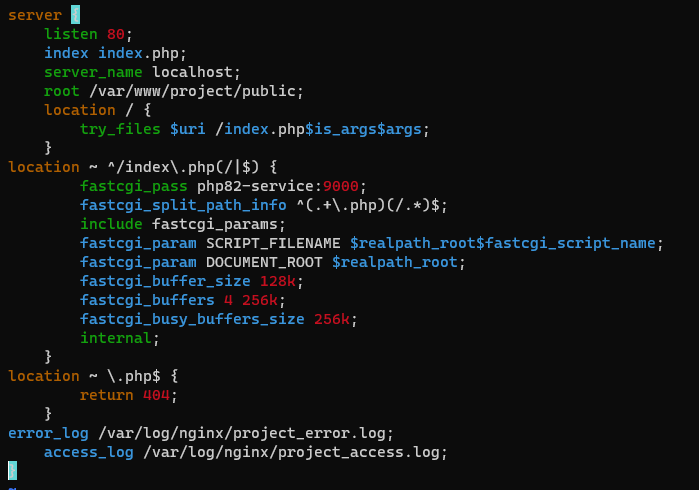
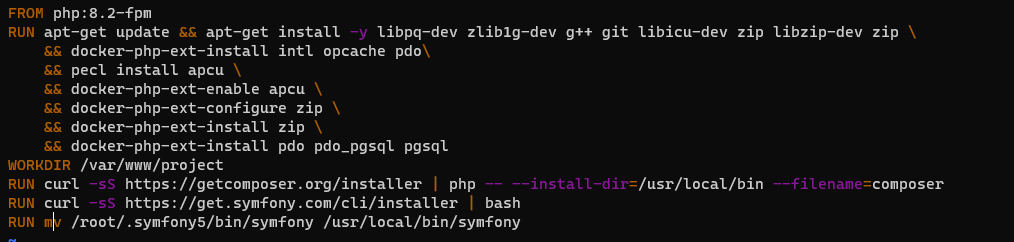


Рисунок 22 – файл default.conf (конфигурация сервера)

**7)** Настроим DATABASE\_URL и заполним бд



Рисунок 23 – Содержимое DATABASE\_URL

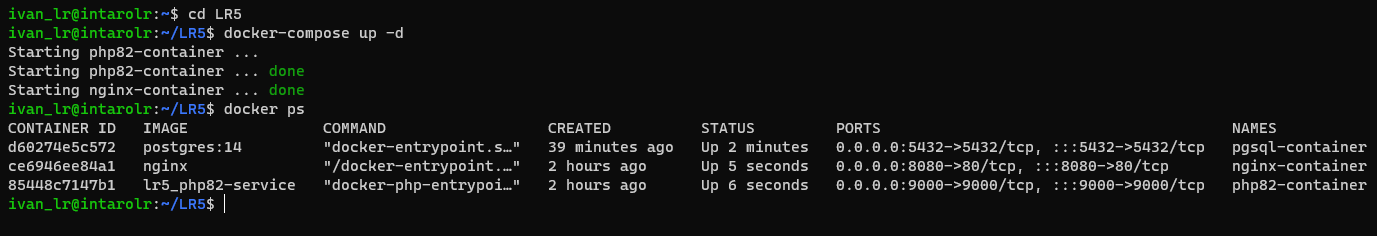
И заполним БД с помощью команд

php bin/console doctrine:schema:create

php bin/console doctrine:fixtures:load

**8)** Запустим команду docker-compose up –d

Рисунок 24 – Запуск команды docker-compose up –d



**9)** Откроем наш сайт в браузере

Перед этим нужно настроить проброс портов в VirtualBox

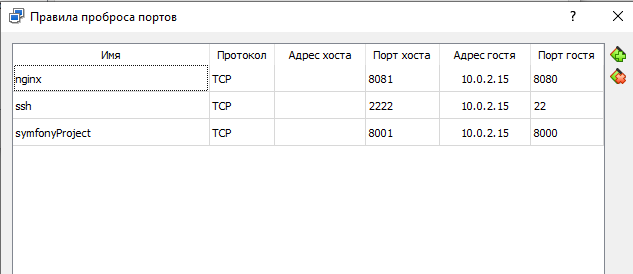


Рисунок 25 – Проброс портов в Virtualbox

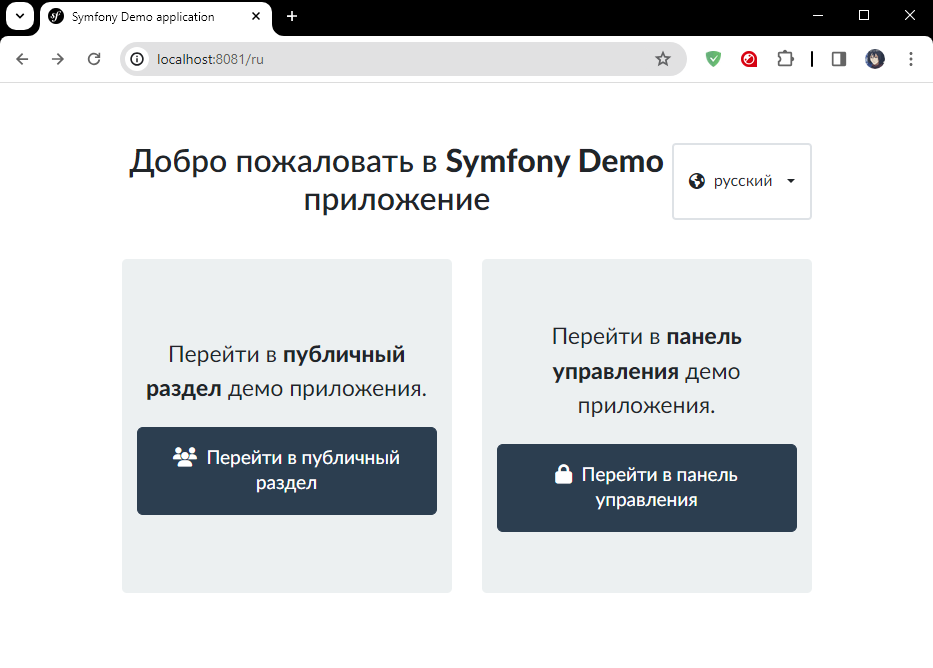


Рисунок 26 – Главная страница сайта

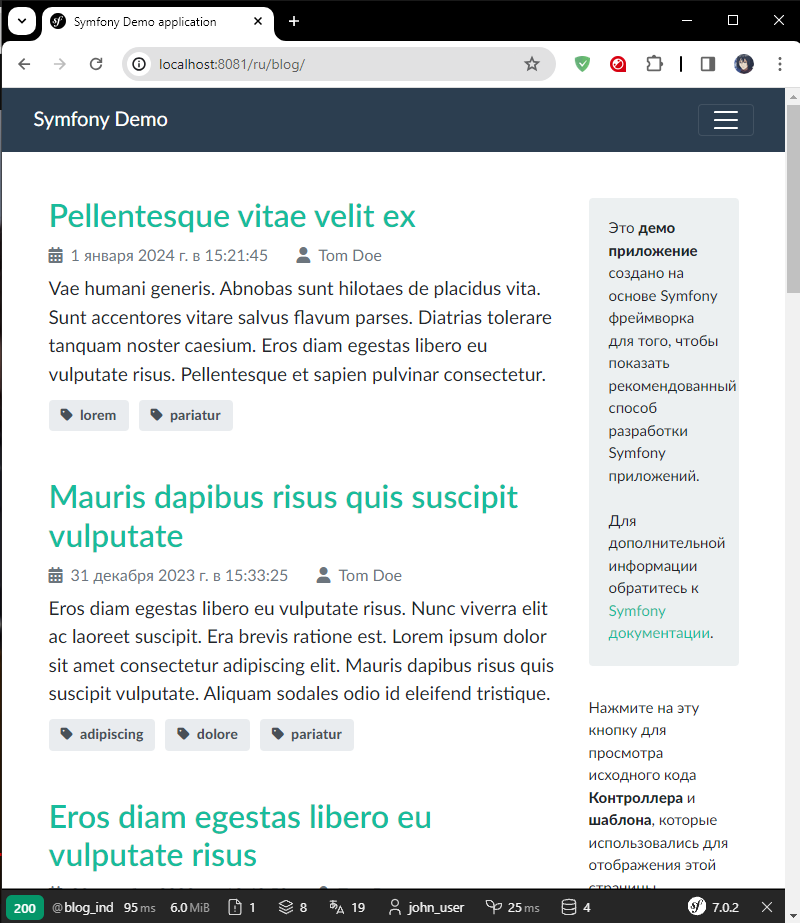


Рисунок 27 – Публичный раздел сайта (демонстрация работы postgres)

# **II часть**

## **Шаг №1. Установка Nginx**

Создадим docker-compose.yml

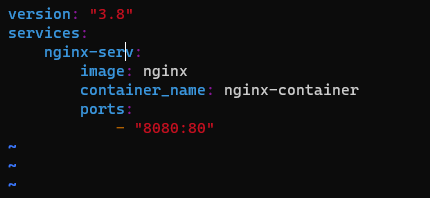


Рисунок 28 – Содержимое docker-compose.yml

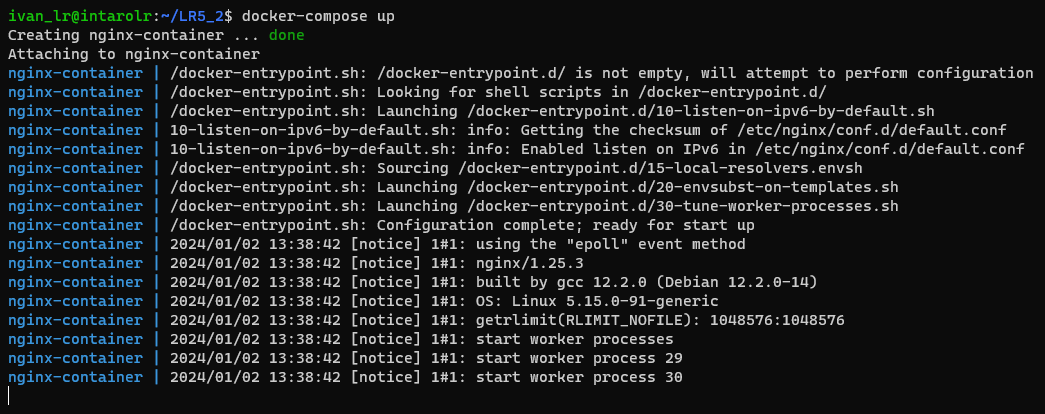


Рисунок 29 – Запуск docker-compose up

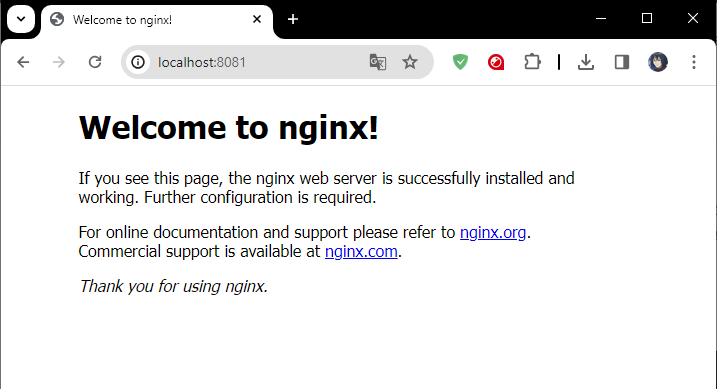


Рисунок 30 – Страничка Welcome to nginx!

## **Шаг №2. Передача в контейнер html-файлов**

Пропишем в docker-compose.yml пути для монтирования конфига nginx и html файлa.

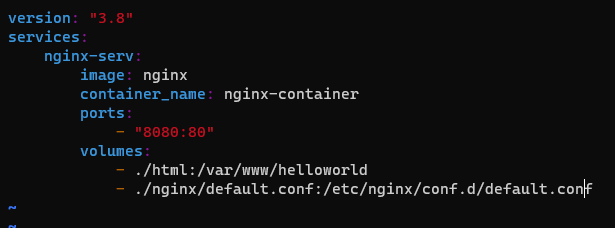


Рисунок 31 – Файл docker-compose.yml

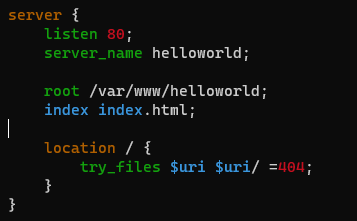


Рисунок 32 – Файл default.conf (конфиг nginx)

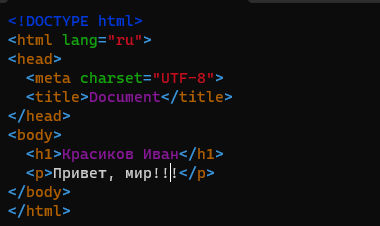


Рисунок 33 – Содержимое файла index.html

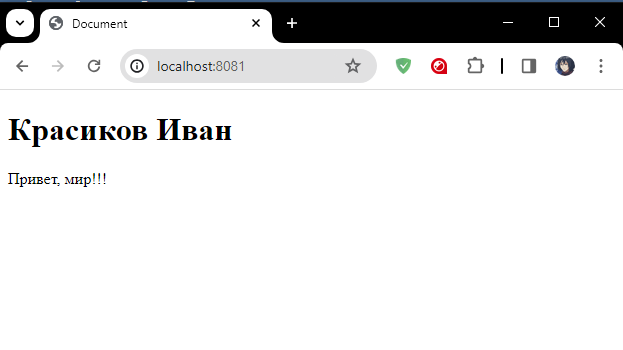


Рисунок 34 – Страница сайта

## **Шаг №3. Web-разработка**

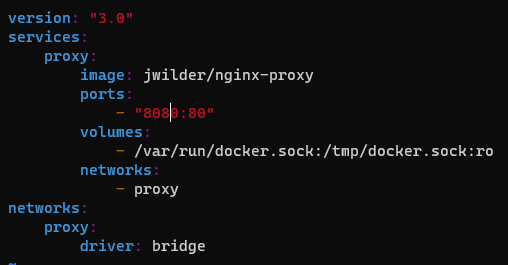


Рисунок 35 – docker-compose.yml для nginx-poxy

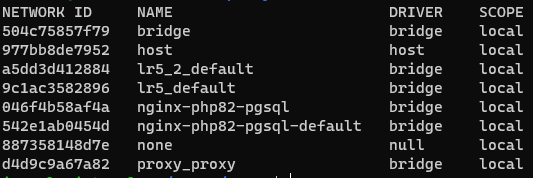


Рисунок 36 – docker network ls

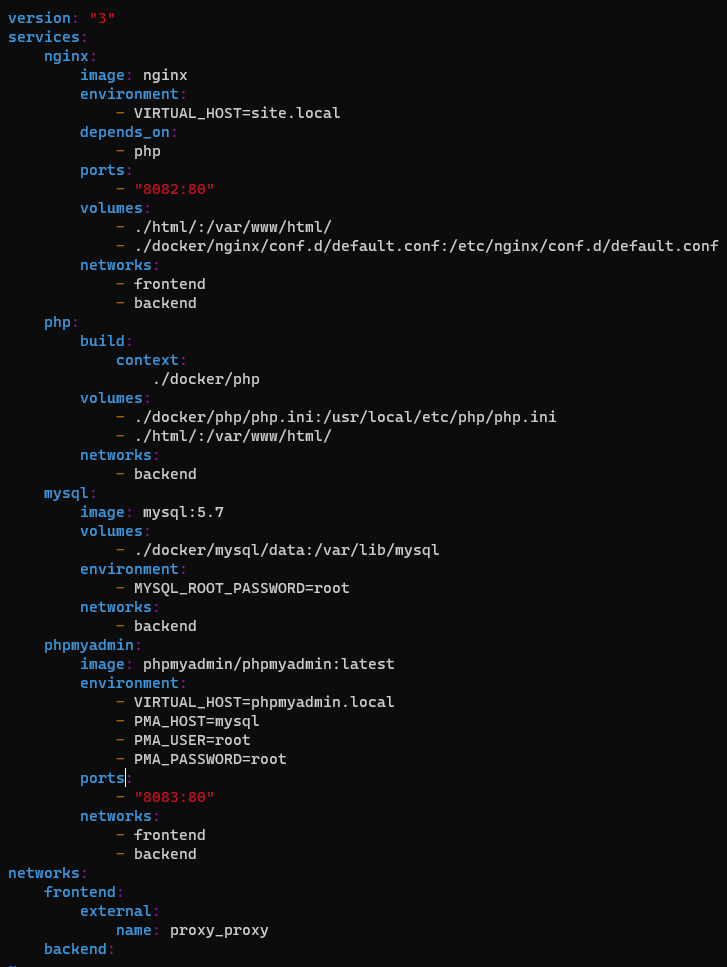


Рисунок 37 – Основной docker-compose.yml

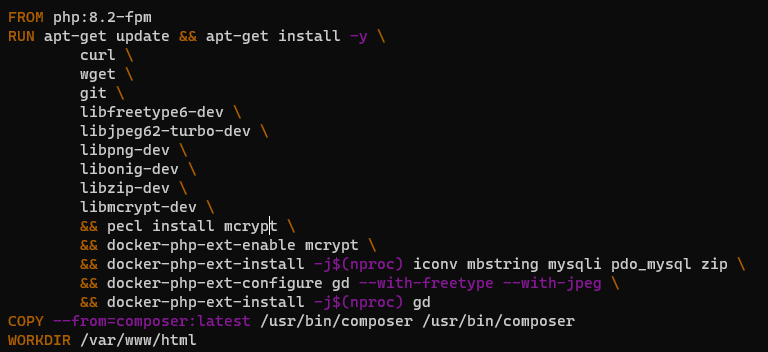


Рисунок 38 – Dockerfile для php

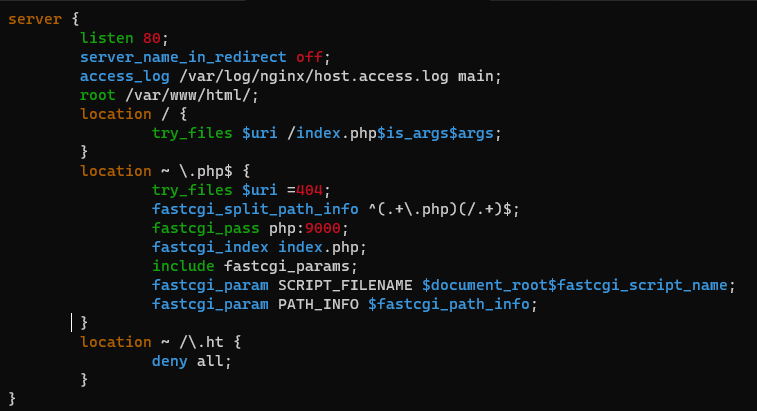


Рисунок 39 – Файл default.conf

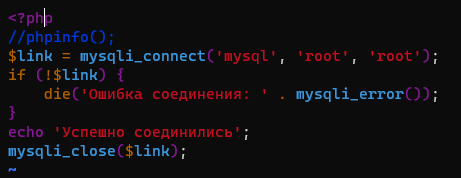


Рисунок 40 – Файл index.php

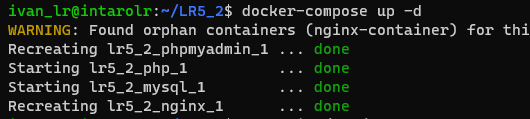


Рисунок 41 – Запуск docker-compose

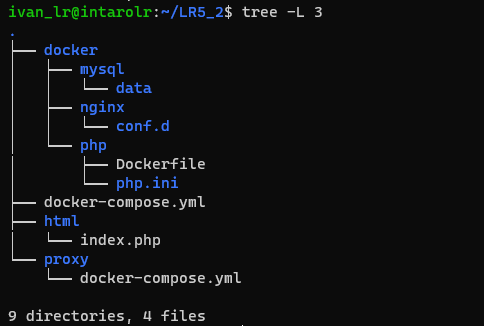


Рисунок 42 – Структура проекта

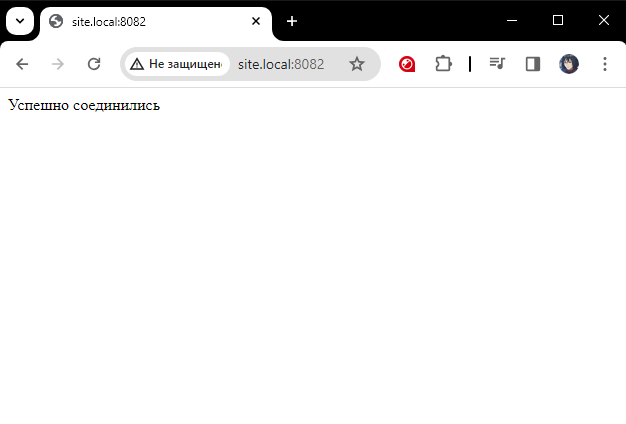


Рисунок 43 – Проверка работоспособности

## **Шаг №4. Запуск Wordpress.**

Для запуска Wordpess на уже готовом нашем сервере, добавим папку wordpress и допишем пару моментов в docker-compose.yml, добавив туда загрузку wordpress.

version: "3"

services:

nginx:

image: nginx

environment:

- VIRTUAL\_HOST=site.local

depends\_on:

- php

ports:

- "8082:80"

volumes:

- ./wordpress/:/var/www/html/

- ./docker/nginx/conf.d/default.conf:/etc/nginx/conf.d/default.conf

networks:

- frontend

- backend

php:

build:

context:

./docker/php

volumes:

- ./docker/php/php.ini:/usr/local/etc/php/php.ini

- ./wordpress/:/var/www/html/

networks:

- backend

mysql:

image: mysql:5.7

volumes:

- ./docker/mysql/data:/var/lib/mysql

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=wordpress

networks:

- backend

phpmyadmin:

image: phpmyadmin/phpmyadmin

ports:

- "8083:80"

environment:

- VIRTUAL\_HOST=phpmyadmin.local

- PMA\_HOST=mysql

- PMA\_USER=root

- MYSQL\_ROOT\_PASSWORD=root

networks:

- frontend

- backend

wordpress:

depends\_on:

- mysql

image: wordpress:5.1.1-fpm-alpine

container\_name: wordpress

restart: unless-stopped

env\_file: .env

environment:

- WORDPRESS\_DB\_HOST=mysql:3306

- WORDPRESS\_DB\_USER=root

- WORDPRESS\_DB\_PASSWORD=root

- WORDPRESS\_DB\_NAME=wordpress

volumes:

- ./wordpress/:/var/www/html/

networks:

- frontend

- backend

networks:

frontend:

external:

name: proxy\_proxy

backend:

После чего запустим docker-compose и перезагрузим наш сервер nginx

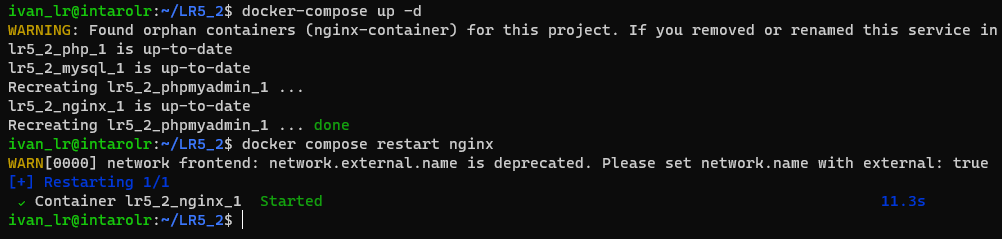


Рисунок 44 – Запуск docker-compose и перезагрузка nginx

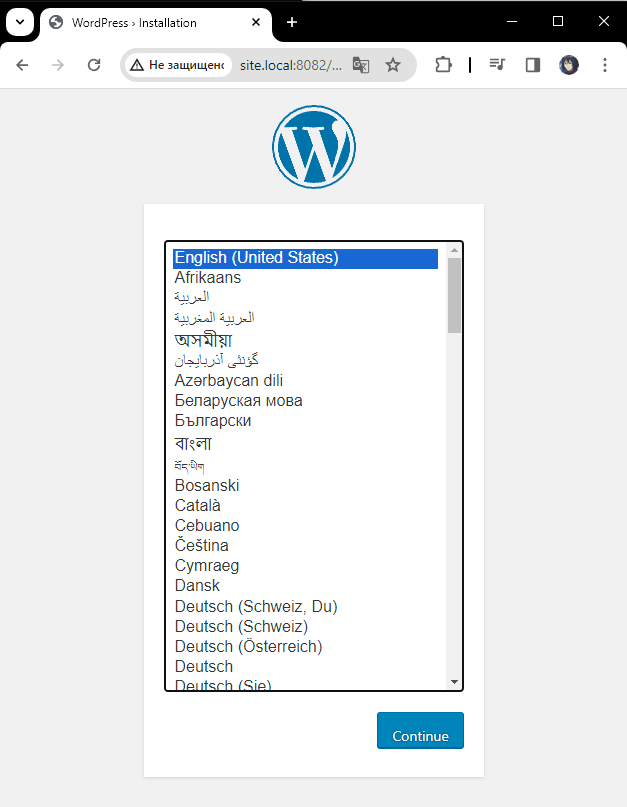


Рисунок 45 – Окно загрузки wordpress

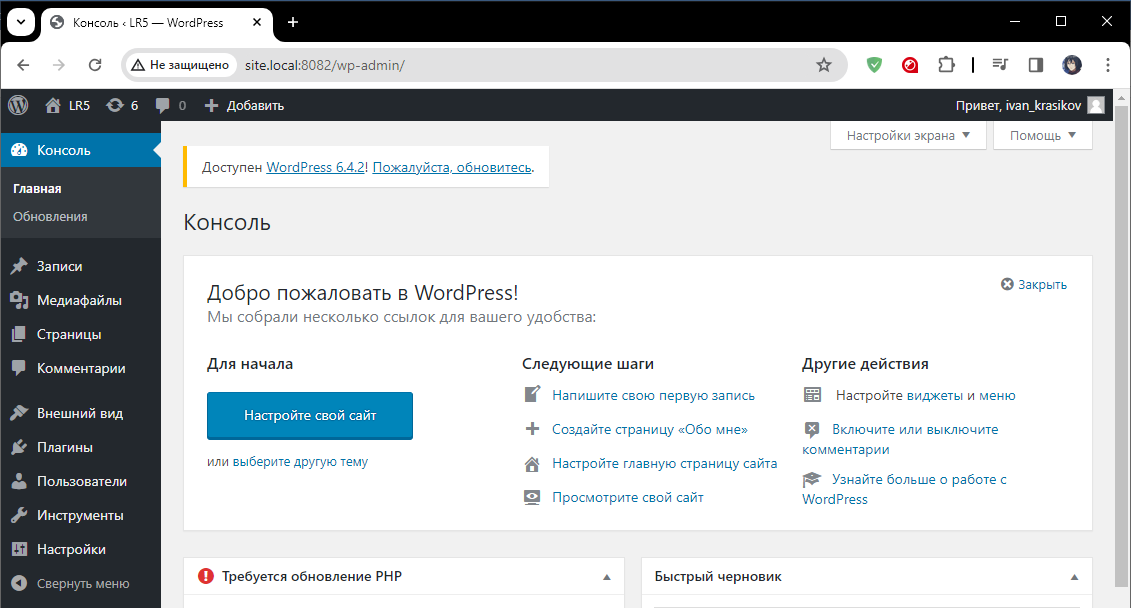


Рисунок 46 – Wordpress

# **Вопросы для самопроверки**

1. Назовите отличия использования контейнеров по сравнению с виртуализацией:

A. Меньшие накладные расходы на инфраструктуру

C. Невозможность запуска GNU/Linux- и Windows-приложений на одном хосте

2. Назовите основные компоненты Docker:

В. Контейнеры

D. Реестры – так называются публичные и приватные хранилища докер образов, например, Docker Hub.

3. Какие технологии используются для работы с контейнерами?

A. Пространства имен (Linux Namespaces) – изоляция в виртуализации обеспечивается через ограничения со стороны процессора, а в контейнеризации это делается через пространство имен: при создании контейнера докер создает набор пространства имен для данного контейнера.

C. Контрольные группы (cgroups) – позволяют разделать доступные ресурсы железа и создавать ограничения при необходимости

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 обеспечивают легковесную виртуализацию, работая на уровне операционной системы и разделяя ее ядро с хост-системой. Они обеспечивают быстрое развертывание и экономичное использование ресурсов, так как контейнеры делят общие ресурсы и изолируют приложения на уровне пользовательского пространства. В отличие от этого, виртуальные машины полностью изолированы, имеют собственные виртуальные ресурсы и требуют больше времени на запуск, что делает их более подходящими для приложений, где требуется высокая степень изоляции. Контейнеры облегчают масштабирование и управление приложениями, в то время как виртуальные машины предоставляют более высокий уровень изоляции, но могут быть более тяжеловесными в использовании ресурсов.

6. Перечислите основные команды утилиты Docker с их кратким описанием

run <название образа>– запуск контейнера из образа

* –it – позволяет войти в контейнер, чтобы работать изнутри. Например, в контейнер с Ubuntu мы может работать на bash.
* --name – позволяет задать имя контейнеру
* –v или --mount – указывает для контейнера папки монтирования
* –p – позволяет пробросить порты из контейнера на хост
* –d – запуск контейнера в фоновом режиме
* –e – устанавливает переменную окружения
* --link – позволяет связать контейнер с другим

pull <название образа>– скачивание образа из реестра

stop - остановка контейнера

ps – просмотр списка работающих контейнеров

* –а – весь список контейнеров (запущенных и нет)

rm - удаляет остановленный или жестко завершает контейнер

images – просмотр списка образов на хосте

rmi – удаляет образ (перед удалением обязательно остановить все контейнеры этого образа)

exec <контейнер> <команда> - позволяет выполнить команду на работающем контейнере

build – сборка образа на основе файлов

push – позволяет отправить образ в реестр (нужны имя и адрес)

inspect – просмотр полной информации о контейнере

network ls – просмотр сетей на хосте

compose up – запуск мультиконтейнерного приложения

* –d – запуск в фоновом режиме
* --scale <name>=<num> - создает кол-во реплик контейнера 29

compose build – сборка мультиконтейнерного приложения на основе файла docker-compose.yml

7. Каким образом осуществляется поиск образов контейнеров?

Сначала докер ищет указанный образ на хосте, и если его не находит, то идет на доступные реестры и скачивает их оттуда. Реестр может быть частным, или запрашиваемый образ может быть закрытым – для доступа к таким образам нужно авторизоваться – команда docker login.

8. Каким образом осуществляется запуск контейнера?

Контейнер запускается на основе образа. Образ может быть скачен из реестра или создан на хосте. Созданные на хосте образы обычно включает в себя инструкцию FROM, указывающую на основе какого образа осуществляется создание текущего. В образе указаны все необходимые зависимости и другие инструкции. При запуске контейнера из образа на хосте в первый раз его сначала нужно собрать командой docker build. Сборка осуществляется по слоям, каждый слой имеет свой размер. Если осуществляется пересборка образа, то идентичные инструкции будут взяты из кеша.

9. Что значит управлять состоянием контейнеров?

Управление состоянием контейнеров включает в себя контроль за запуском, остановкой, мониторингом и обновлением контейнеризованных приложений. Этот процесс позволяет эффективно использовать ресурсы, обеспечивает стабильность работы приложения и предоставляет возможность динамической конфигурации, такой как изменение переменных среды и настроек сети. Управление состоянием также включает в себя миграцию контейнеров между хостами и масштабирование приложения путем добавления или удаления экземпляров контейнеров в зависимости от потребностей. Общее управление состоянием обеспечивает гибкость и надежность в эксплуатации контейнеризованных сред.

10. Как изолировать контейнер?

Контейнеры изолированы по умолчанию, но есть возможность отключить все механизмы изоляции Docker – в таком случае запуск 30 приложения в контейнере не будет отличаться от запуска на хосте. Две основы изоляции – это Linux namespase и cgroup.

11. Опишите последовательность создания новых образов, назначение Dockerfile?

Создание новых образов в Docker включает в себя использование файла, называемого Dockerfile. Dockerfile представляет собой текстовый файл, в котором определяются инструкции и шаги для построения контейнера. В начале Dockerfile обычно указывается базовый образ, на основе которого будет строиться новый контейнер. Затем добавляются инструкции для установки зависимостей, копирования файлов приложения, настройки переменных среды и других параметров. Каждая инструкция Dockerfile создает новый слой образа, оптимизируя кэширование и упрощая процесс обновления. В конечном итоге, Dockerfile служит для автоматизации процесса сборки контейнера, делая его воспроизводимым и легко управляемым.

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

Да. Например, облачные сервисы: Fly.io, Stackpath, Deno.land, Vercel.app.

13. Опишите назначение системы оркестрации контейнеров Kubernetes. Перечислите основные объекты Kubernetes?

Система оркестрации контейнеров (не обязательно Kubernetes) нужны для управления контейнерами. Она позволяет их создавать (планировать время создания тоже), масштабировать, распределять между несколькими хостами, выделять ресурсы, следить за статусами контейнеров, перезапускать и многое другое. Kubernetes – одна из систем оркестрации, но очень хорошая. Kubernetes не зависит от языка программирования, платформы и операционной системы, он предлагает широкий спектр вариантов развертывания. Кроме того, он предоставлят множество разных удобных функций, которых нет, например, в Docker Swarm – бесплатной встроенной в докер системе окрекстации. Основные объекты Kubernetes:

* 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.