**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по курсу "Операционная система Linux"

Контейнеризация

Поленников А.А.

|  |
| --- |
| Студент |
| Группа АИ-17 | Кургасов В.В. |  |
| Руководитель  Доцент |  |  |

Липецк 2019г.

Цель работы

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

Задание

Изучить теоретический материал и выполнить предложенные практические задания.

В результате необходимо:

- Знать назначение и возможности Docker;

- Знать особенности установки и настройки Docker;

- Владеть инструментом для определения и запуска многоконтейнерных приложений Docker – Docker Compose.

Задание:

1. 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.

Ход работы

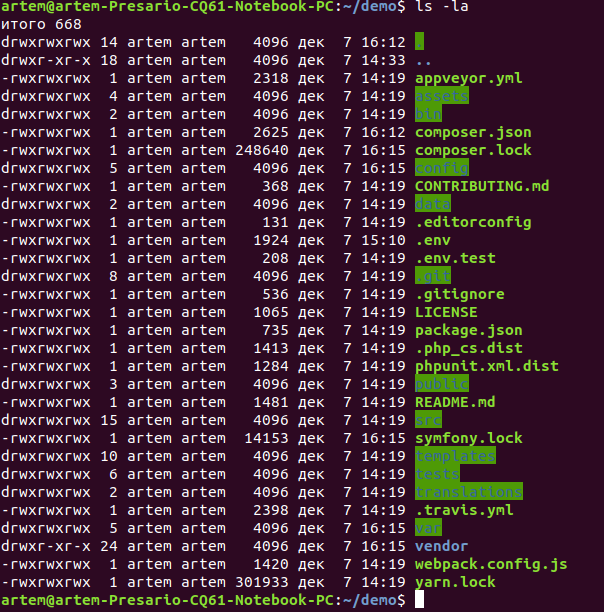


Рисунок 1 – клонирование тестового проекта

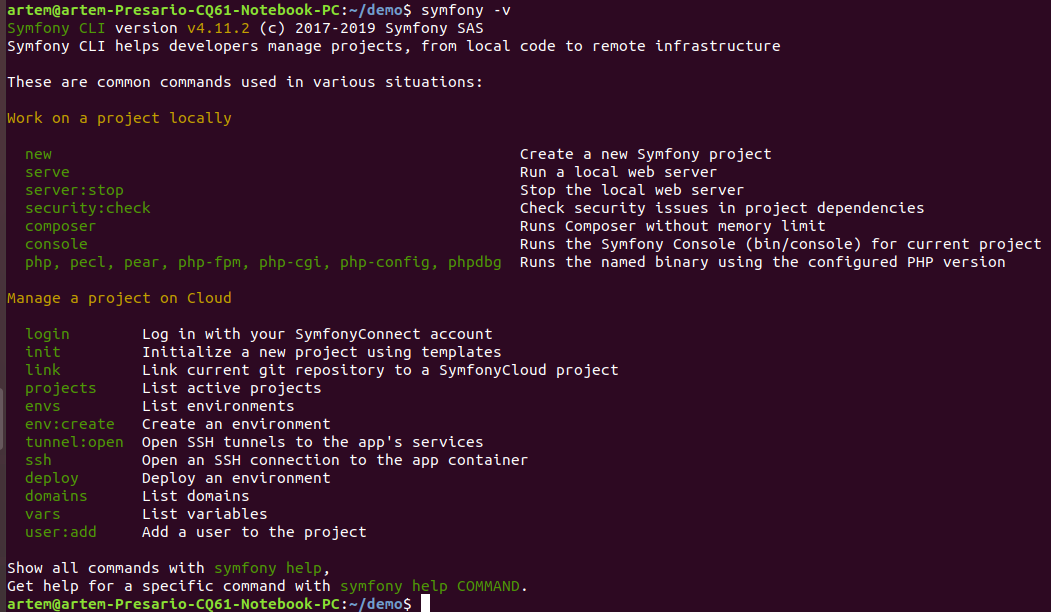


Рисунок 2 – установка Symfony

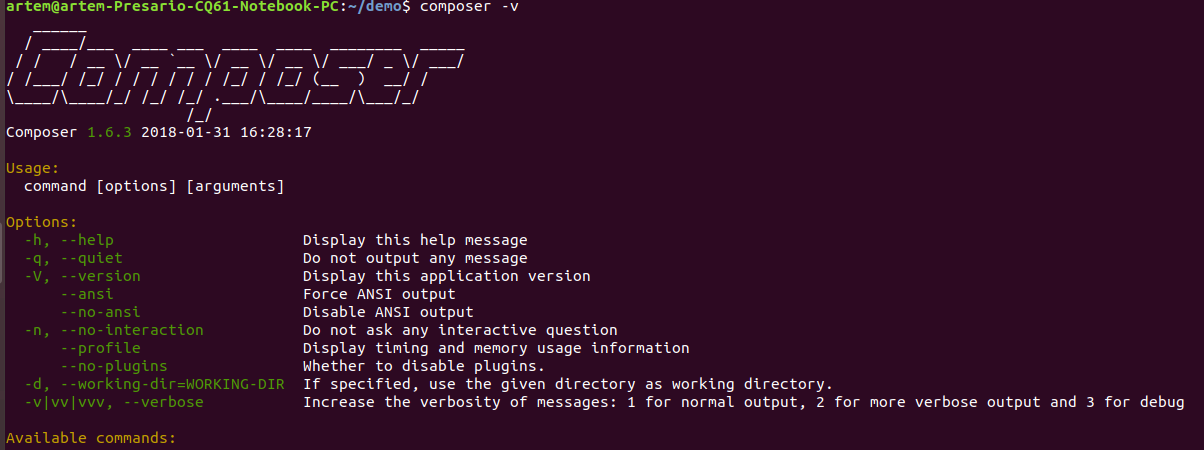


Рисунок 3 – установка Docker-compose

Создание БД для тестирования

sudo su – postgres

psql

create database db\_demo;

\password

\q

exit

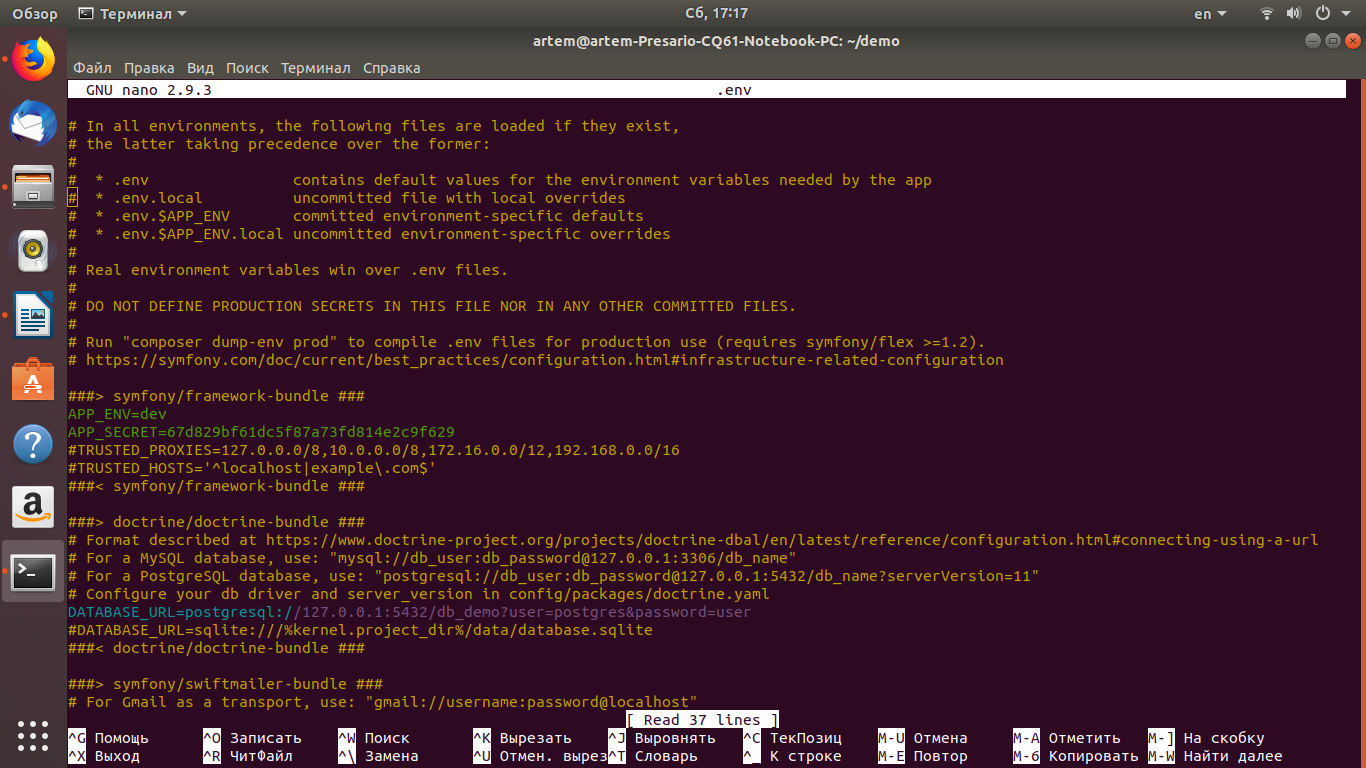


Рисунок 4 – редактирование файла .env

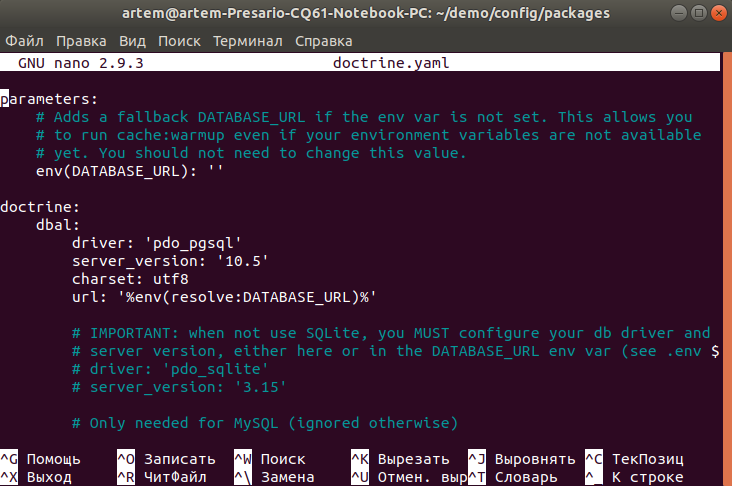


Рисунок 5 – редактирование файла doctrine.yaml

Заполнение БД

php bin/console doctrine:schema:create

php bin/console doctrine:fixtures:load

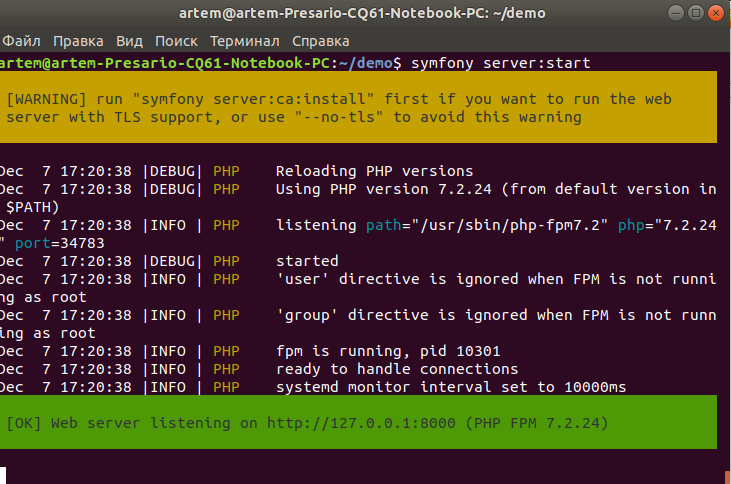


Рисунок 6 – запуск сервера

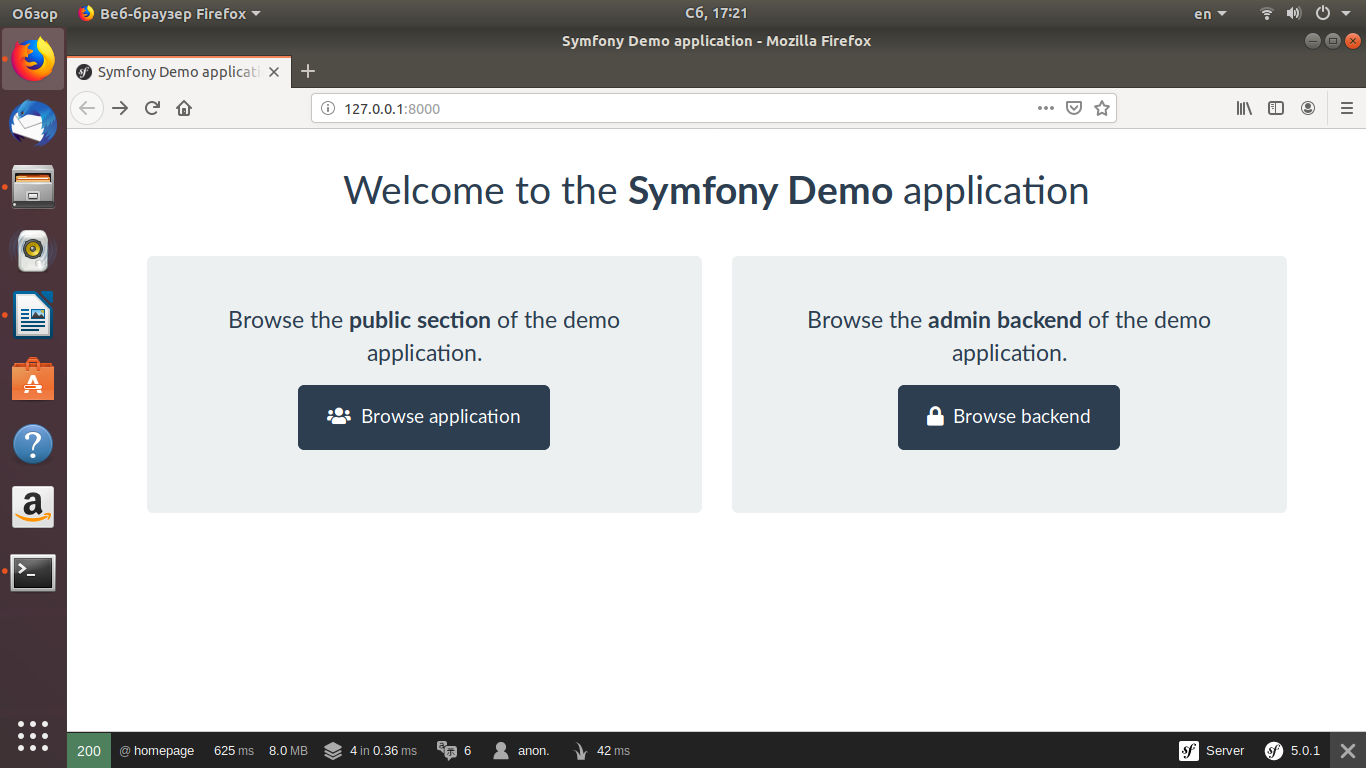


Рисунок 7 – запуск сайта без использования Docker контейнеров

Настройка Docker для запуска приложения. Docker будет использоваться для запуска всех служб, таких как Nginx, PostgreSQL и PHP-FPM.



Рисунок 8 – содержимое папки docker

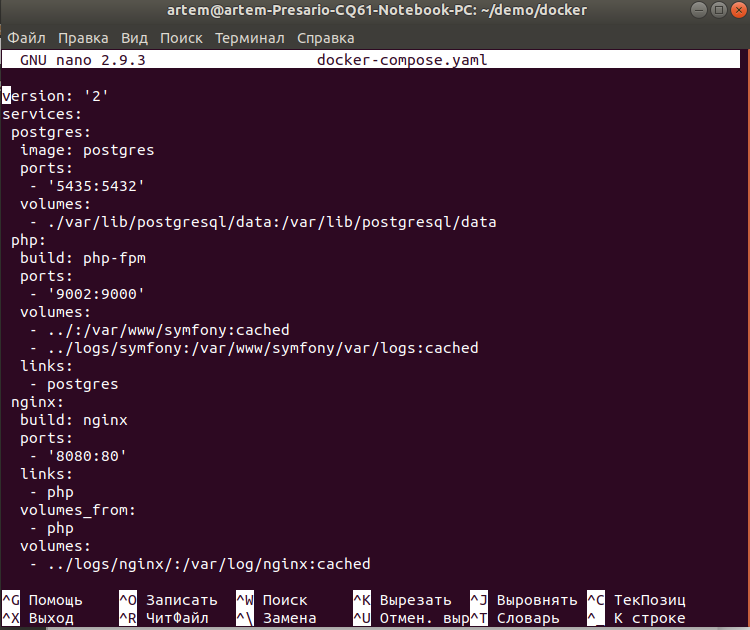


Рисунок 9 - конфигурационный файл docker-compose.yaml внутри каталога docker

Это файл конфигурации, используемый Docker Compose, где определяются сервисы, отношения между ними и многое другое.

Создание файла конфигурации PHP-FPM

Для построения образа PHP-FPM необходимо создать Dockerfile конфигурационный файл.

Внутри каталога docker создать php-fpm подкаталог, внутри него создать Dockerfile, который содержит все необходимые расширения PHP для запуска приложения

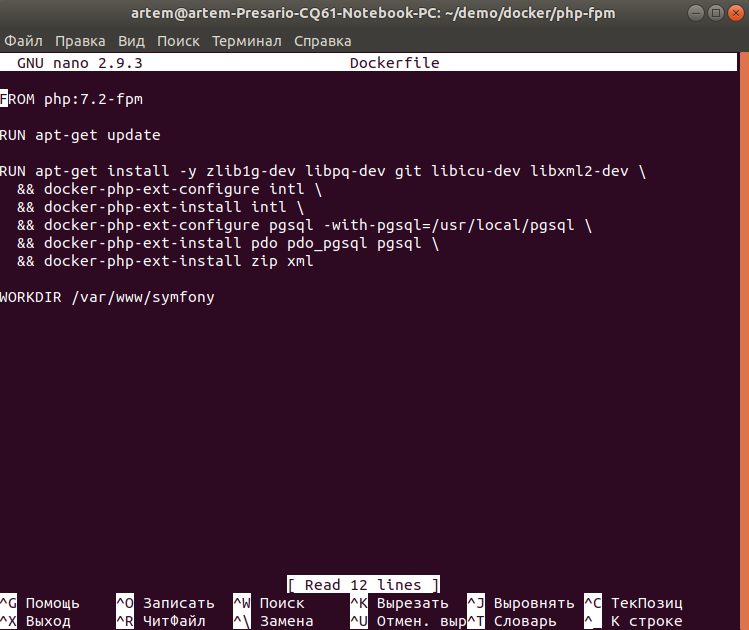


Рисунок 10 – Dockerfile в подкаталоге php-fpm

Создание конфигурационного файла Nginx

Внутри docker/nginx каталога, создается Dockerfile файл конфигурации для сервера Nginx.

Благодаря этому файлу Docker сможет построить образ Nginx, который будет использоваться для веб-обслуживания нашего приложения.

Dockerfile определяет, что последняя версия Nginx должна быть установлена, и содержимое из default.conf файла должно использоваться в качестве основной конфигурации сервера Nginx.

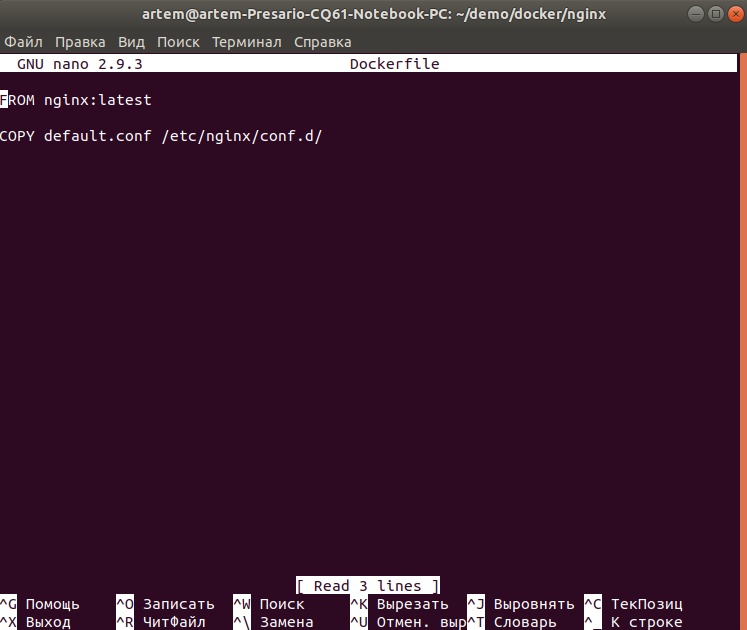


Рисунок 11 - Содержимое файла Dockerfile

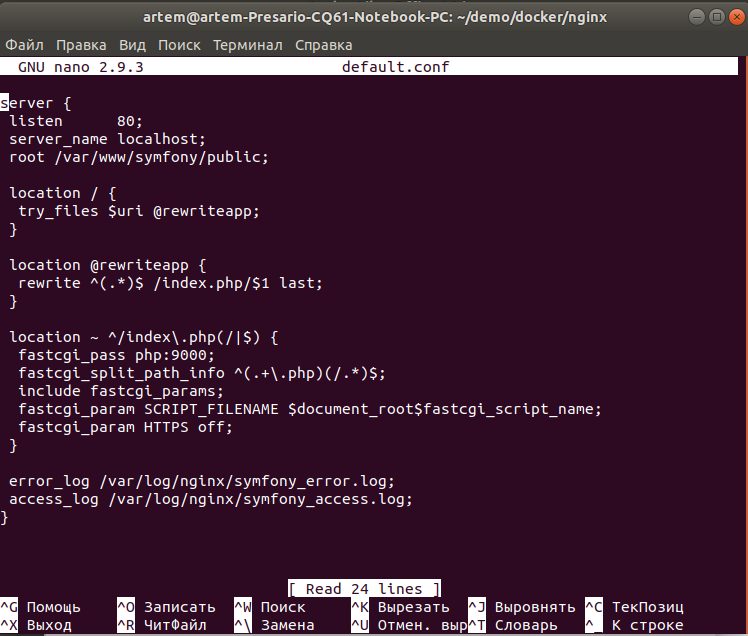


Рисунок 12 - default.conf файл, который настроен для обслуживания приложения

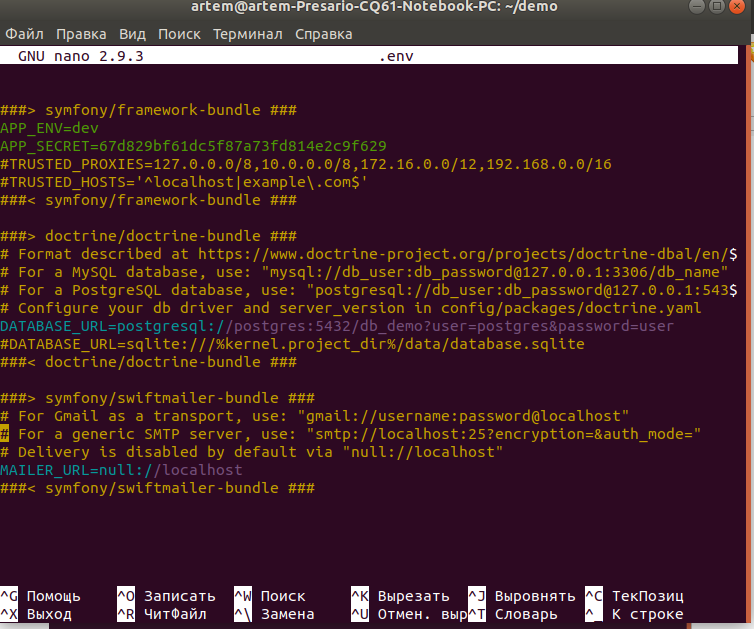


Рисунок 13 – Редактирование строки DATABASE\_URL для подключение к postgres

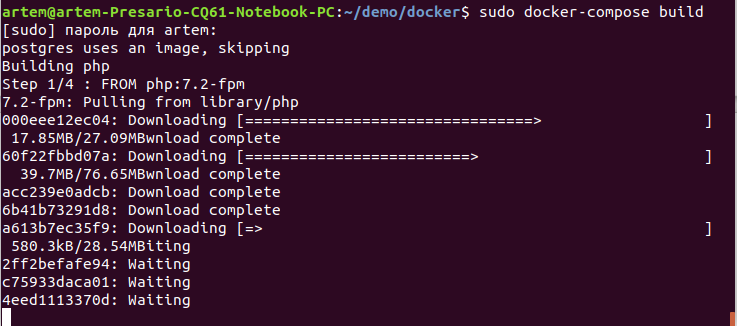


Рисунок 14 – cборка образов и установка зависимостей между контейнерами

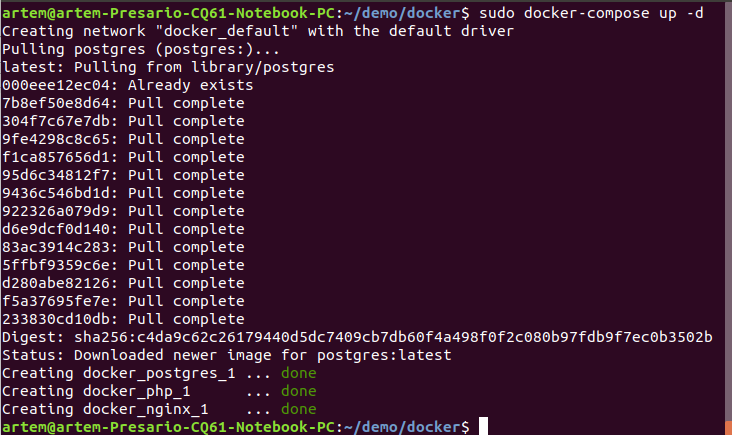


Рисунок 15 – запущенные контейнеры

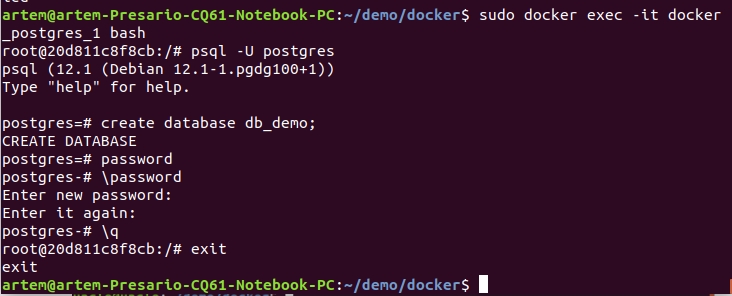


Рисунок 16 – создание БД в контейнере postgres

Заполнение БД

php bin/console doctrine:schema:create

php bin/console doctrine:fixtures:load

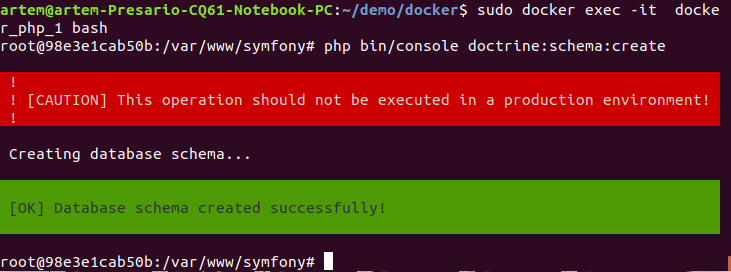


Рисунок 17 – заполнение БД данными

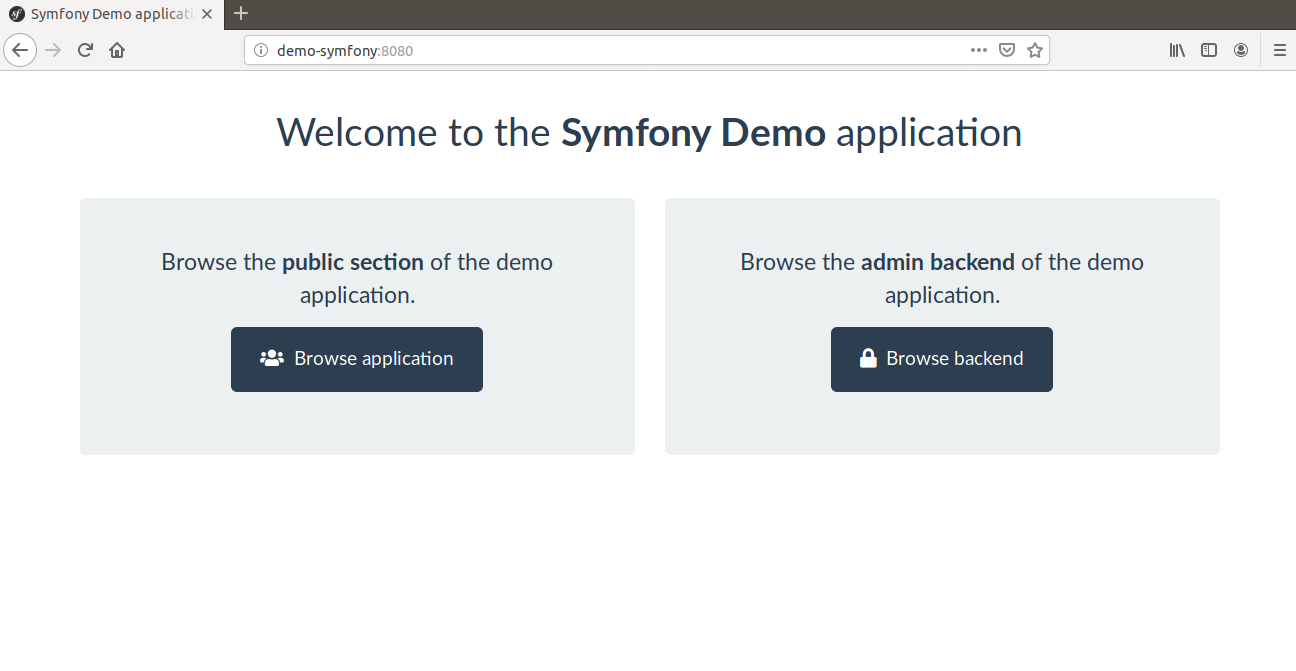


Рисунок 18 – информация по адресу demo-symfony:8080

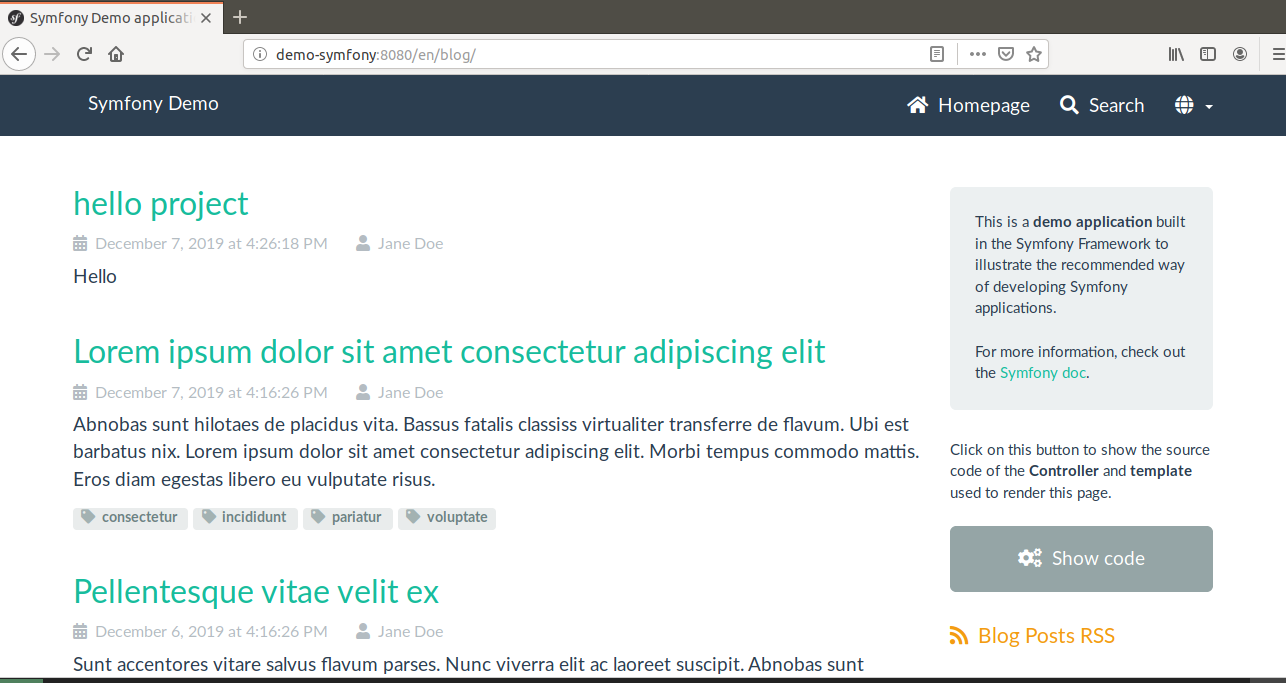


Рисунок 19 – добавление записи

Вывод

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

Вопросы для самопроверки

1. Назовите отличия использования контейнеров по сравнению с виртуализацией.

A. Меньшие накладные расходы на инфраструктуру

B. Время старта приложений больше

2. Назовите основные компоненты Docker.

D. Реестры

3. Какие технологии используются для работы с контейнерами?

A. Пространства имен (Linux Namespaces)

C. Контрольные группы (cgroups)

D. Аппаратная поддержка виртуализации

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 - сетевые хранилища образов

(репозитории)

5. В чем отличие контейнеров от виртуализации?

Контейнеры в отличие от виртуальной машины, обеспечивающей аппаратную виртуализацию, обеспечивает виртуализацию на уровне операционной системы с помощью абстрагирования «пользовательского пространства». В целом, контейнеры выглядят как виртуальные машины. Одна большая разница между контейнерами и виртуальными машинами в том, что контейнеры разделяют ядро хоста с другими контейнерами. Каждый контейнер получает свое собственное изолированное пользовательское пространство для обеспечения возможности запуска нескольких контейнеров на одном хосте. Архитектура уровня операционной системы разделяется между контейнерами, именно поэтому контейнеры настолько легковесны.

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

create - создание контейнера из образа.

start - запуск существующего контейнера.

run - создание контейнера и его запуск.

ls - вывод списка работающих контейнеров.

inspect - вывод подробной информации о контейнере.

logs - вывод логов.

stop - остановка работающего контейнера с отправкой главному процессу контейнера сигнала SIGTERM, и, через некоторое время, SIGKILL.

kill - остановка работающего контейнера с отправкой главному процессу контейнера сигнала SIGKILL.

rm - удаление остановленного контейнера.

7. Каким образом осуществляется поиск образов контейнеров?

Используя команду «docker search», можно запустить поиск Docker образов на сервере регистра с терминала

8. Каким образом осуществляется запуск контейнера?

Когда вы запускаете Docker контейнер, вот какие действия Docker на самом деле выполняет:

* Загрузка образа.
* Распаковка образа в «пакет» (bundle). Это действие выравнивает слои в одну файловую систему.
* Запуск контейнера из пакета.

9. Что значит управлять состоянием контейнеров?

Контейнеры докер могут находится в нескольких состояниях:

Исполняется - контейнер работает. В выводе docker ps увидите статус «Up» и время, в течение которого он исполняется.

Создан - контейнер создан, но в настоящий момент не выполняется. Такое состояние будет у контейнера после команды docker create.

Завершил исполнение - контейнер завершил исполнение.

10. Как изолировать контейнер?

Изолирование контейнеров конфигурируется в файлах Dockerfile и docker-compose.yaml

11. Опишите последовательность создания новых образов, назначение Dockerfile?

Определим, какой образ требуется создать.

В начале файла добавьте строку с указанием базового образа, который мы будем использовать:

#Загрузить базовый образ Ubuntu 18.04

FROM ubuntu:18.04

Обновим программный репозиторий Ubuntu внутри Dockerfile при помощи команды ‘RUN‘.

#Обновить программный репозиторий Ubuntu

RUN apt-get update

Теперь установим приложения, который требуются для нашего образа. При помощи apt можно установить Nginx, PHP-FPM.

В файлах Dockerfile содержатся инструкции по созданию образа. С них, набранных заглавными буквами, начинаются строки этого файла. После инструкций идут их аргументы. Инструкции, при сборке образа, обрабатываются сверху вниз

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

Да возможно с помощью движка Kubernetes

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

Kubernetes является проектом с открытым исходным кодом, предназначенным для управления кластером контейнеров Linux как единой системой. Kubernetes управляет и запускает контейнеры Docker на большом количестве хостов, а так же обеспечивает совместное размещение и репликацию большого количества контейнеров.

Краткий обзор ключевых объектов API 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.