**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра Автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по «ОПЕРАЦИОННАЯ СИСТЕМА LINUX»

Контейнеризация

Студент Самойлов В.Д.

Группа АИ-17

Руководитель

Доцент Кургасов В.В.

Липецк 2019г.

Цель работы

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

Задание

С помощью Docker Compose на своем компьютере поднять сборку nginx+php-fpm+postgres, продемонстрировать ее работоспособность, запустив внутри контейнера демо-проект на symphony (Исходники взять отсюда <https://github.com/symfony/demo/>).

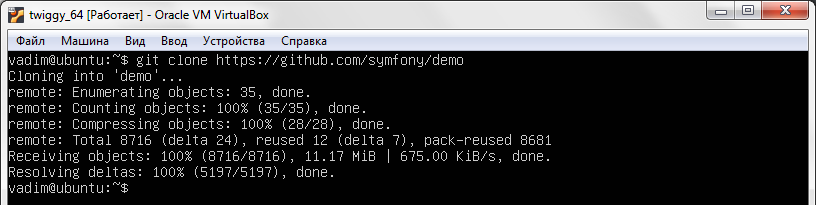
По умолчанию проект работает с sqlite-базой. Нужно заменить ее на postgres. Проект должен открываться по адресу <http://demo-symfony.local/> (Код проекта должен располагаться в папке на локальном хосте) крнтейнеры с fpm и nginx должны его подхватывать.

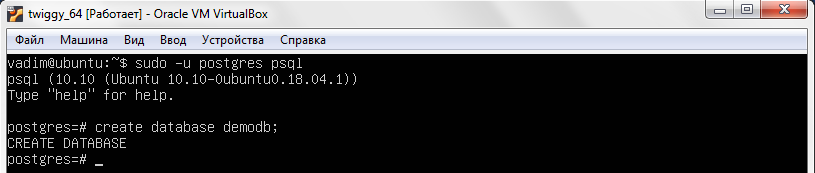
Для компонентов nginx, fpm есть готовые docker-образы, их можно и нужно использовать.

Нужно расшарить папки с локального хоста, настроить подключение к БД. В .env переменных для postgres нужно указать путь к папке, где будет лежать база, чтобы она не удалялась при остановке контейнера.

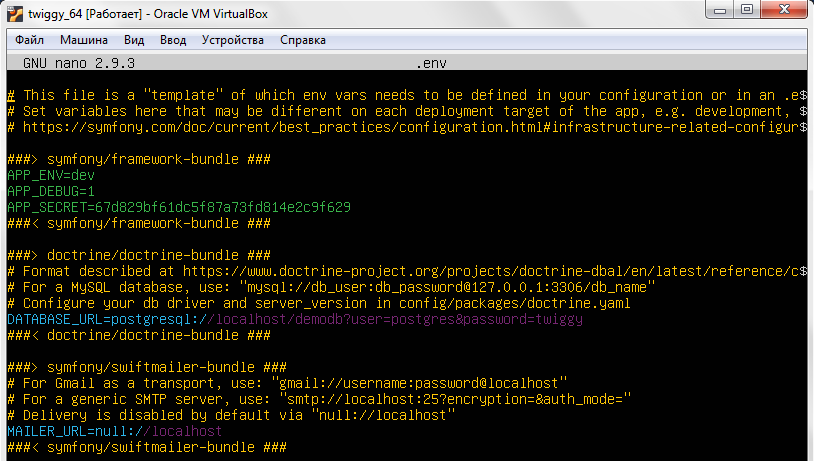
На выходе должен получиться файл конфигурации docker-compose.yml и .env файл с настройками переменных окружения.

Ход работы

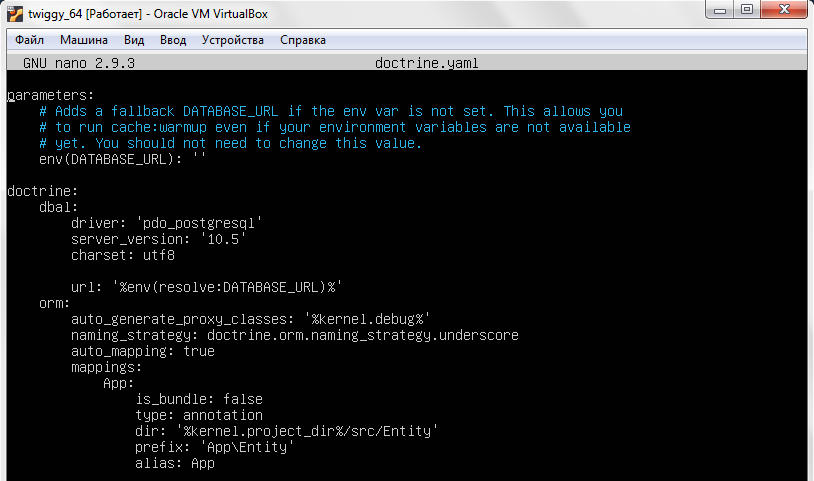
1. Клонируем текстовый проект
2. Результат операции
3. Настройка подключения к базе данных
   1. Создадим базу данных postgresql для проекта



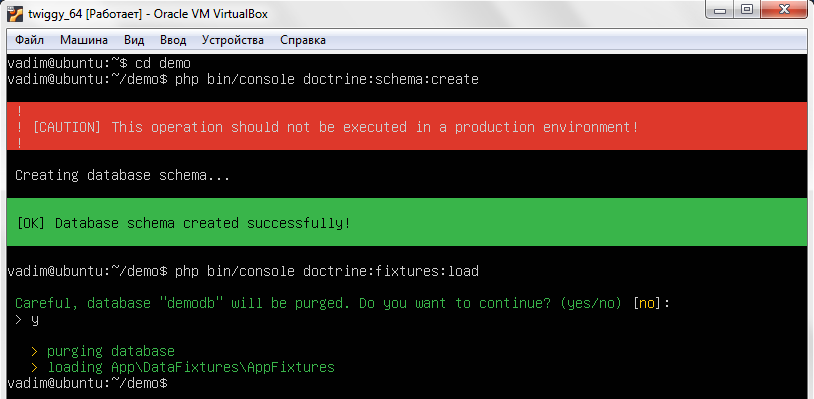
* 1. Поменяем DATABASE\_URL в файле .env



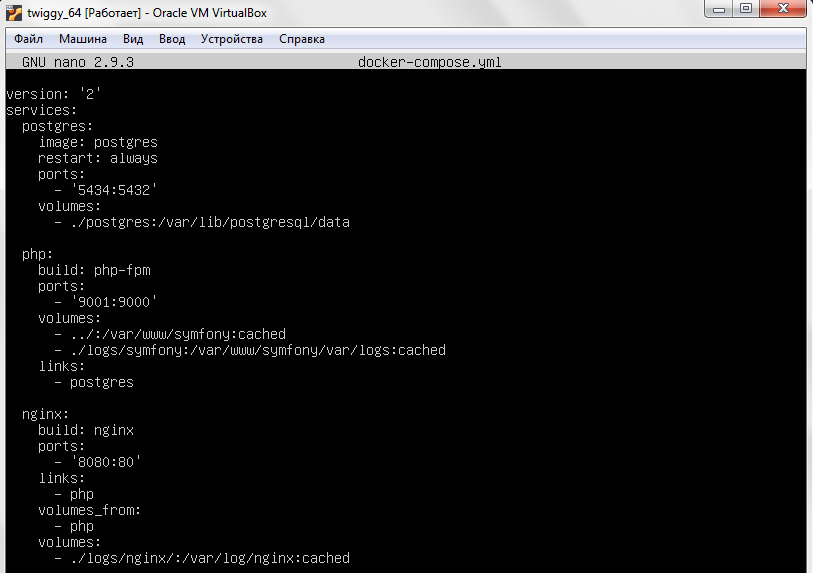
1. Новый database\_url
   1. Изменим файл config/packages/doctrine.yaml



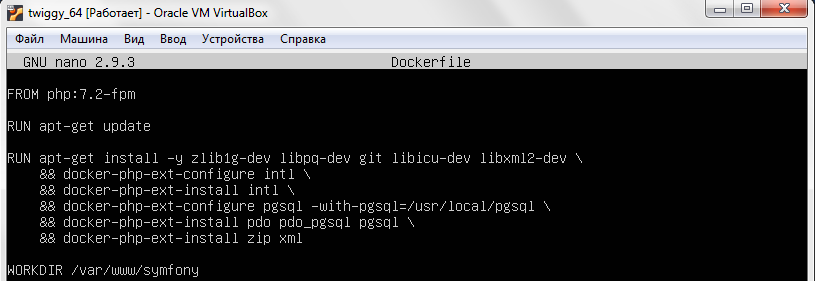
1. Результат изменения
   1. Создадим схему базы данных и заполним ее фикстекстур



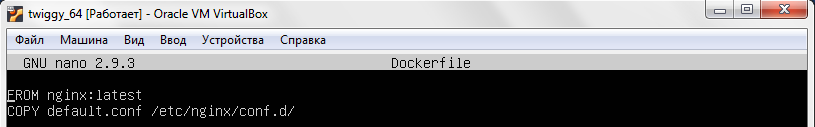
1. Результат создания схемы бд
2. Создадим в папке с проектом папку c файлами конфигурации, необходимые для Docker, назвав ее docker.
   1. Создадим файл конфигурации docker-compose.yml



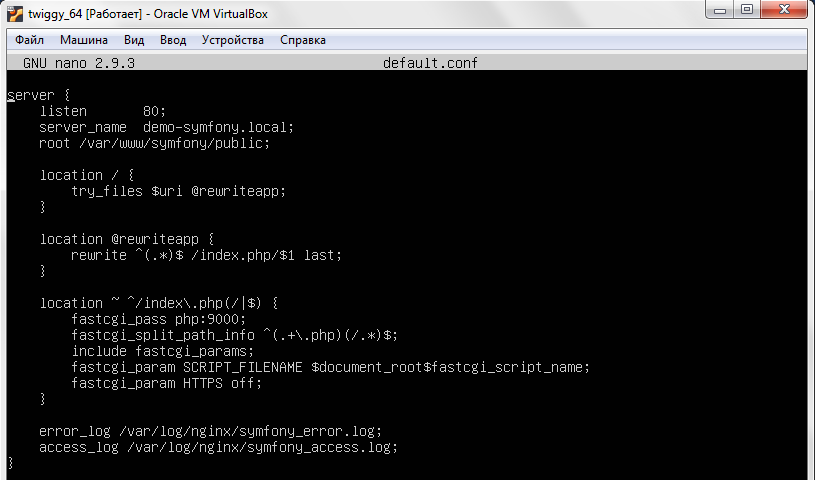
1. Файл docker-compose.yml
   1. Создадим подкаталог php-fpm для Dockerfile контейнера php-fpm и напишем его.



1. Файл Dockerfile для контейнера php-fpm
   1. Создадим подкаталог nginx для Dockerfile контейнера nginx и напишем его.



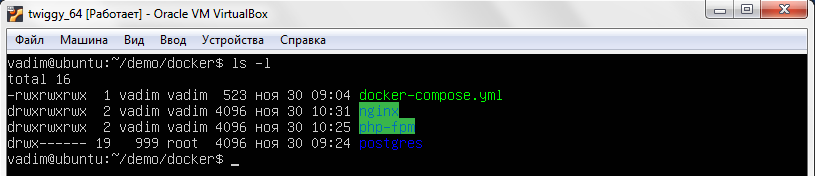
1. Файл Dockerfile для контейнера nginx
   1. Создадим в подкаталоге файл default.conf, который настроен для обслуживания приложения demo-symfony.



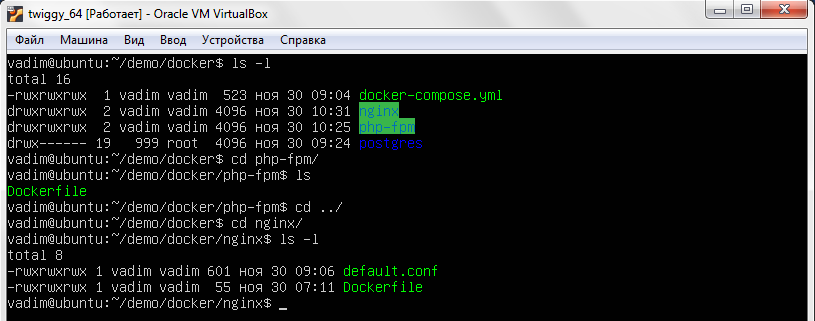
1. Файл default.conf
   1. Создадим каталог для хранения базы данных postgres
   2. Дадим полный доступ на папку demo

Sudo chmod –R 777 demo

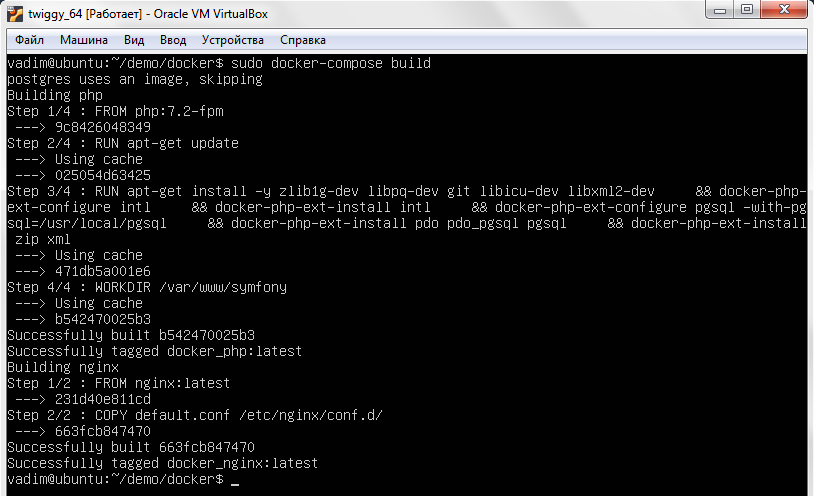
В результате мы получим следующую структуру каталога docker



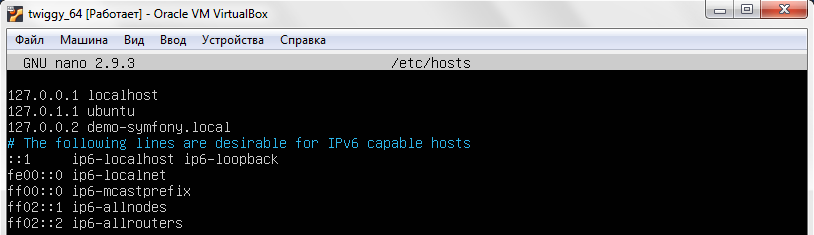
1. Структура каталога docker



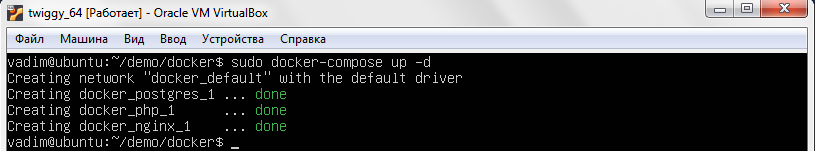
1. Структура каталогов php-fpm и nginx
2. Сборка проекта



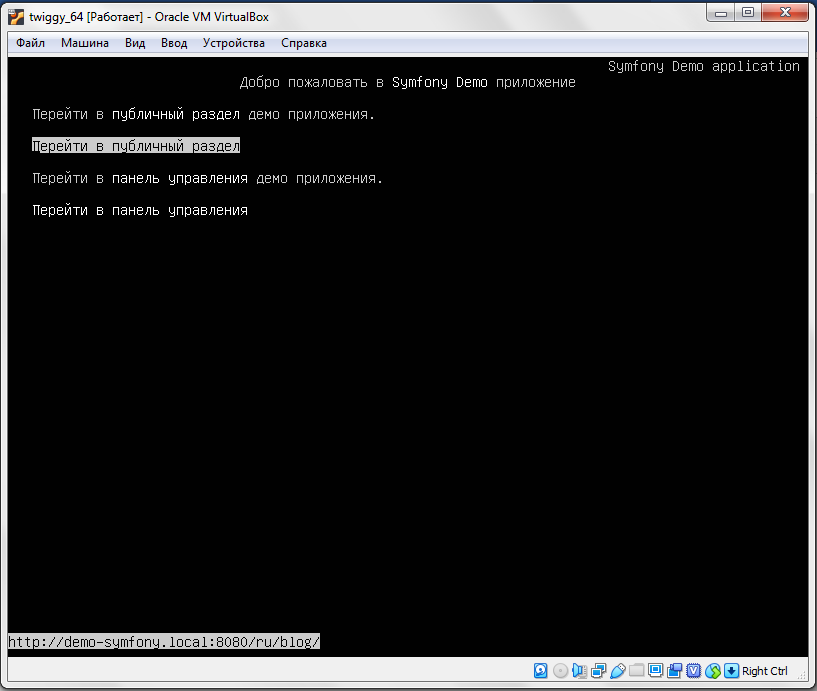
1. Результат сборки проекта
2. Отредактируем файл /etc/hosts



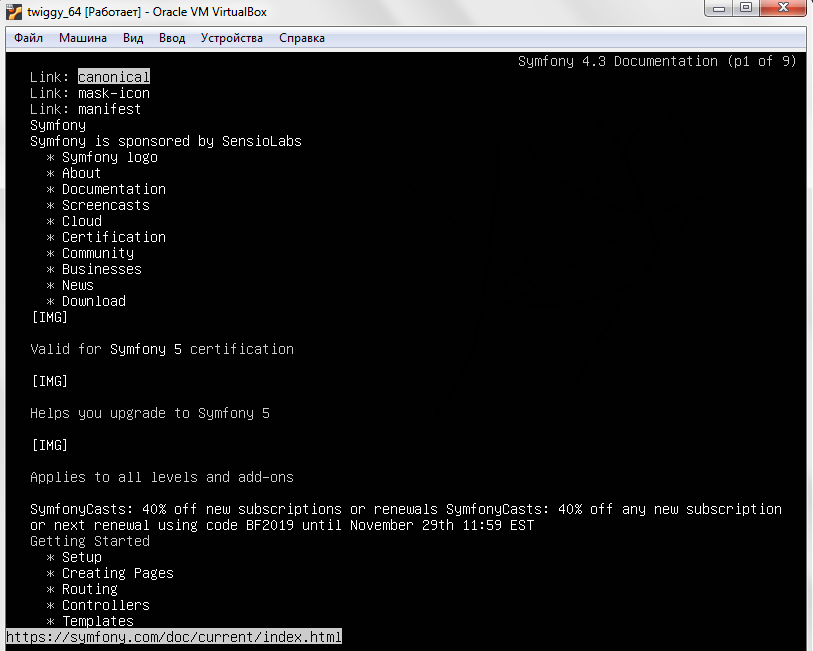
1. Результат изменения файла /etc/hosts
2. Запустим проект demo-symfony



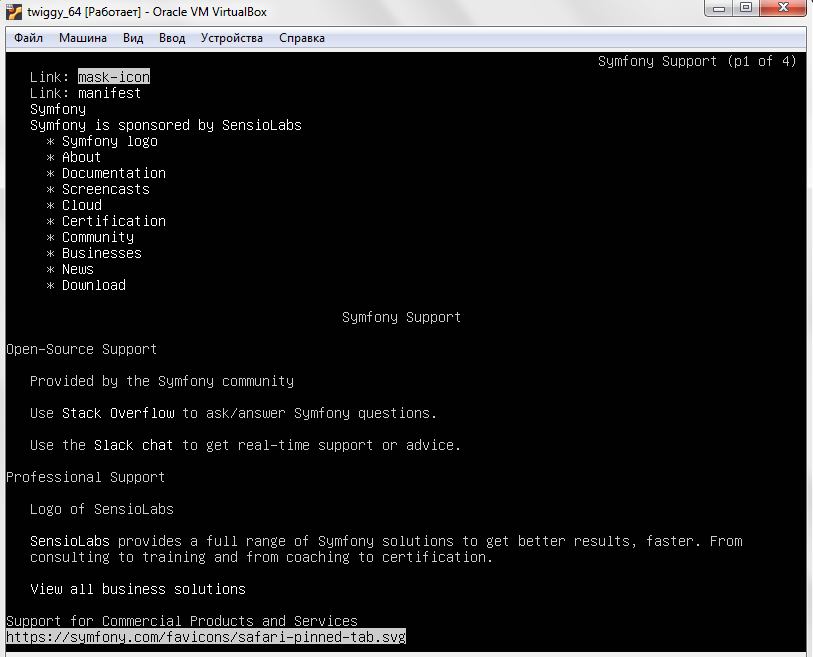
1. Результат
2. В консольном браузере links перейдем по адресу <http://demo-fymfony.local:8080/>



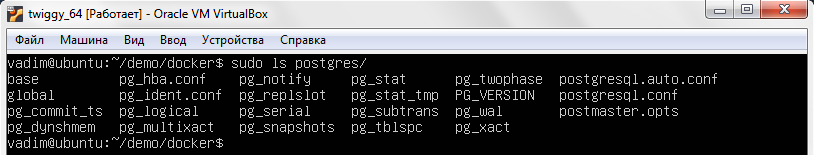
1. Приложение demo-symfony



1. Приложение demo-symfony



1. Приложение demo-symfony
2. Структура каталога postgres, в котором хранится база данных



1. Структура каталога postgres

Ответы на вопросы

1. Отличия контейнеров по сравнению с виртуализацией - меньшие накладные расходы на инфраструктуру и время старта приложений больше.
2. Основные компоненты Docker – Реестры
3. Технологии для работы с контейнерами – пространства имен (Linux Namespaces), контрольные группы (cgroups), аппаратная поддержка виртуализации.
4. Соответствие между компонентами 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).

1. Контейнеры обеспечивают аппаратную виртуализацию, обеспечивает виртуализацию на уровне операционной системы с помощью абстрагирования пользовательского пространства. Контейнеры разделяют ядро хоста с другими контейнерами, каждый из них получает свое собственное изолированное пространство для обеспечения запуска нескольких контейнеров на одном хосте.
2. Основные команды Docker:

* Create – создание контейнера из образа
* Start – запуск существующего контейнера
* Run – создание контейнера и его запуск
* Ls – вывод списка работающих контейнеров
* Inspect – вывод информации о контейнере
* Logs – вывод логов
* Stop – остановка контейнера

1. С помощью команды «docker-search» можно осуществить поиск Docker контейнеров на сервере регистра.
2. Запуск контейнера
   * Загрузка образа
   * Распаковка образа
   * Запуск контейнера из пакета
3. Контейнеры могут находиться в нескольких состояниях:
   * Создан – контейнер создан и в текущий момент не выполняется
   * Исполняется – контейнер работает
   * Завершил исполнение – контейнер завершил исполнение
4. Изолирование контейнеров конфигурируется в файлах Dockerfie и docker-compose
6. С Docker можно работать с помощью движка Kubernetes.
7. Kubernetes является проектом открытым исходным кодом, предназначенным для управления кластером контейнеров Linux как единой системой. Kubernetes запускает и управляет контейнерами Docker на большом количестве хостов, а также обеспечивает совместное размещение и репликацию большого количества контейнеров.

Объекты API Kubernetes:

* Кластеры
* Ноды
* Поды
* Сервисы
* Набор реплик

Вывод

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