Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторным работам №9**

Дисциплина: «Разработка профессиональных приложений»

Docker

Выполнил:

студент группы ИВТАПбд-21

Старостин К. А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, [2023](https://github.com/ghub-ayrtom/DPA-lab5)

# [Постановка задачи](https://github.com/ghub-ayrtom/DPA-lab5)

Для лабораторной работы №9 необходимо подготовить Docker Compose файл, позволяющий запустить 2 контейнера: Первый с автоматическим развертыванием всех необходимых библиотек и запуска gunicorn сервера, а второй для СУБД postgresql.

# Описание реализации

Для начала в папке с проектом был создан файл requirements.txt. В данный файл были прописаны все необходимые библиотеки, которые понадобятся в дальнейшем для развёртывания Docker-контейнеров.

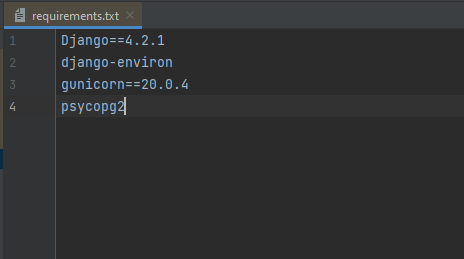


Рисунок 1 Файл с необходимыми библиотеками

Также в папке с проектом был создан файл Dockerfile. Этот файл нужен для записи образа. В нем находится набор инструкций, описывается, что должно находиться в образе, какие команды, зависимости и процессы он будет содержать. Запустив команду docker run программа проверяет, есть ли нужный образ в локальном хранилище:

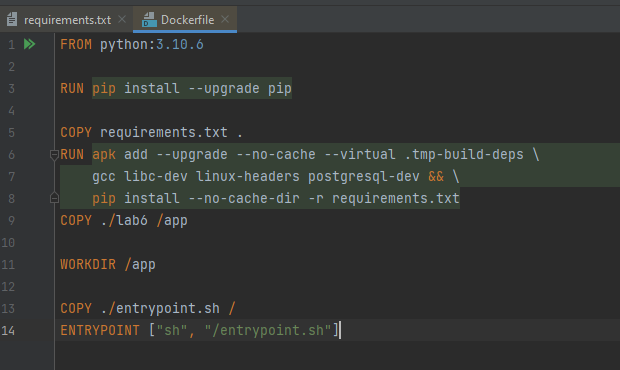


Рисунок 2 Файл Dockerfile

В его конце Dockerfile запускается скрипт entrypoint.sh:

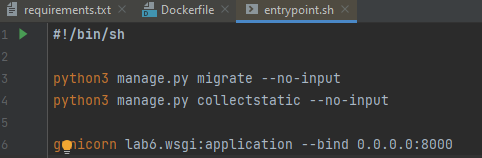


Рисунок 3 Скрипт entrypoint.sh

После этого был создан самый основной файл при работе с технологией Docker – это docker-compose.yml — конфигурационный файл в YAML-формате, описывающий логику запуска и взаимодействия контейнеров между собой. В сущности, инструкции, заложенные в docker-compose.yml, по логике работы идентичны ключам команды docker run:

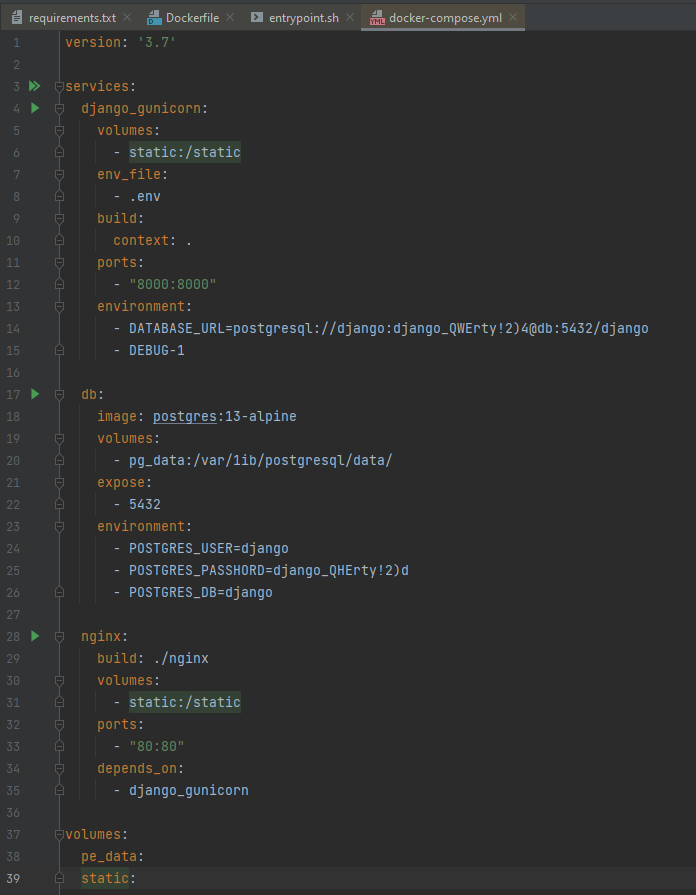
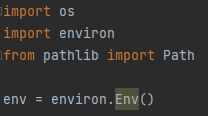
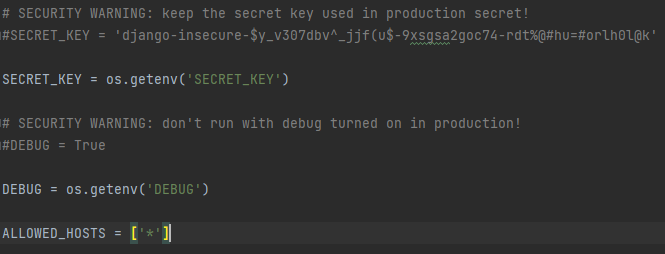
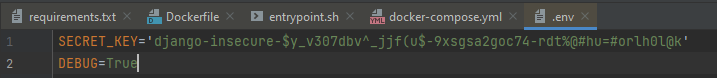


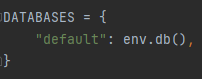
Рисунок 4 Файл docker-compose.yml

Для работы с Docker Compose также следующим образом был изменён файл settings.py нашего Django-приложения:











Дополнительно был создан 3-ий контейнер для развёртывания и запуска nginx сервера.

Для начала, в папке с python проектом DPA-lab6 был создан подкаталог nginx и в нём 2 файла: default.conf и ещё один Dockerfile, который имеет следующую структуру:

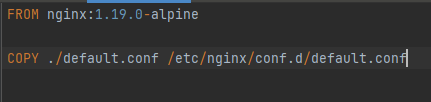
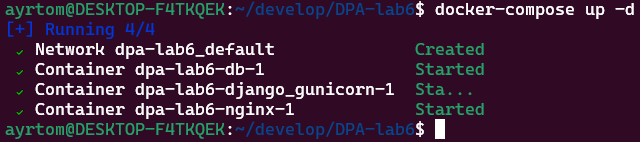


Рисунок 5 Второй Dockerfile

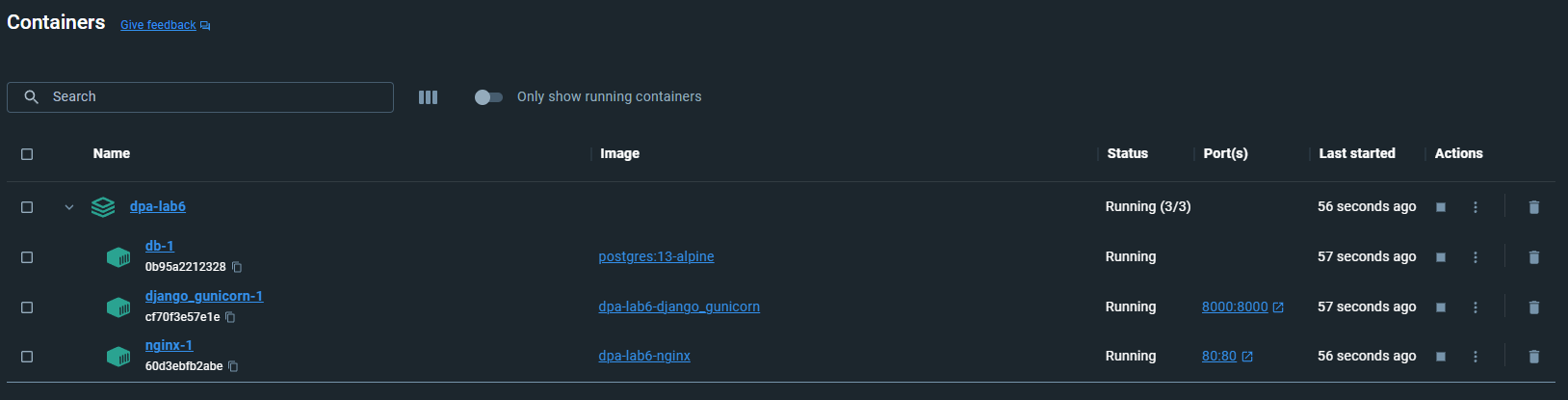
В нём мы командой FROM импортируем nginx версии 1.19.0. и заменяем конфиг nginx по умолчанию тем конфигом, который мы создали сами и который выглядит следующим образом:



Как видно, веб-сервер nginx занимает 80 порт, в свою очередь WSGI-сервер gunicorn занимает стандартный порт 8000, он рассчитан на более серьёзную нагрузку, нежели локальный django-сервер, а также является более безопасным.

После того как все шаги выше были успешно выполнены, можно попробовать запустить docker-compose файл следующей командой в терминале WSL:

Развернулось 3 Docker-контейнера, которые выглядят следующим образом:

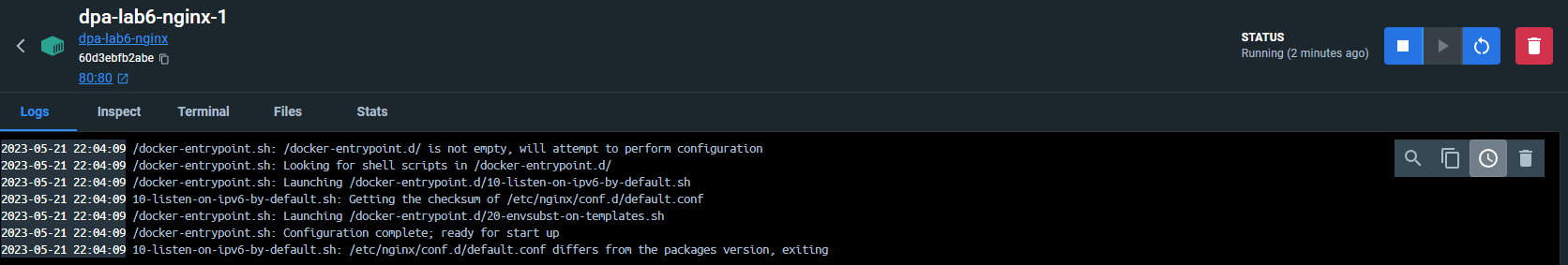


Контейнер dpa-lab6-db-1 отвечает за развёртывание СУБД postgresql:



Контейнер dpa-lab6-django-gunicorn-1 отвечает за автоматическое развёртывание всех необходимых библиотек и запуска gunicorn сервера: 

Контейнер dpa-lab6-nginx-1 отвечает за запуск nginx-сервера:



Теперь можно попробовать зайти на сам локальный сервер gunicorn (порт: 8000) и посмотреть, всё ли работает.

**Описание возникших затруднений**

Затруднения возникли из-за того, что была впервые использована технология контейнеризации Docker. Трудно понять то, с чего необходимо начать, какой именно файл необходимо создать изначально. Также трудно было разобраться во внутренней структуре файлов, в порядке создания файлов и как их связать друг с другом.

Также возникли проблемы с подключением и установкой необходимых библиотек и их совместимостью.