

Docker Compose



АЛЕКСАНДР УЛЬЯНЦЕВ



АЛЕКСАНДР УЛЬЯНЦЕВ

G-Core labs, Backend Software Engineer

 alex@uliantsev.name

 [@res1d3nt](https://t.me/res1d3nt)



План занятия

1. [Оркестрация](#)
2. [Конфигурация docker-compose](#)
 - a. [volumes](#)
 - b. [networks](#)
 - c. [services](#)
3. [Управляющие команды](#)
4. [Итоги](#)
5. [Домашнее задание](#)



Оркестрация

Оркестрация контейнеров

Оркестрация контейнеров - это автоматизация и управление жизненным циклом контейнеров и услуг.

Выполняется для:


- обеспечения развертывания;
- масштабируемости;
- балансировки нагрузки;
- доступности;
- организации сетей контейнеров.

Оркестрация контейнеров

Оркестратор - это дополнительное ПО, которое занимается оркестрацией контейнеров.

Имеется множество инструментов для оркестрации **docker-контейнеров**:

- Docker Compose;
- Docker Swarm;
- Kubernetes;
- другие инструменты.



Конфигурация docker-compose

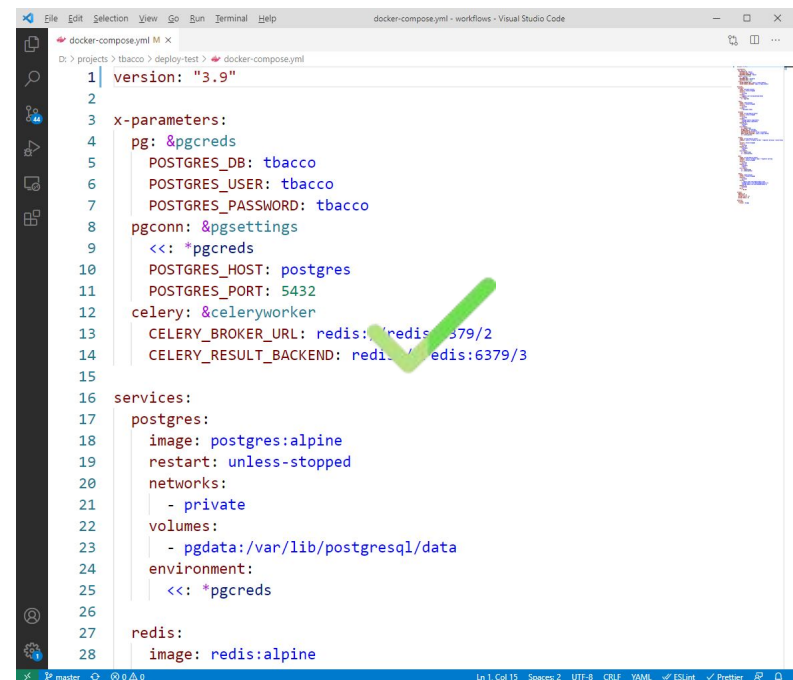
Файл docker-compose.yml

Все описание желаемой конфигурации делается **не командами в терминале**, а в файле **docker-compose.yml** ([дополнительный материал про .yml](#)).



```
C:\WINDOWS\system32\cmd.exe  
C:\SDK\Redis\3.0.503>redis-server.exe redis.windows.conf
```

терминал




```
1 version: "3.9"  
2  
3 x-parameters:  
4   pg: &pgcreds  
5     POSTGRES_DB: tbacco  
6     POSTGRES_USER: tbacco  
7     POSTGRES_PASSWORD: tbacco  
8   pgconn: &pgsettings  
9  
10  <<: *pgcreds  
11    POSTGRES_HOST: postgres  
12    POSTGRES_PORT: 5432  
13  celery: &celeryworker  
14    CELERY_BROKER_URL: redis://redis:6379/2  
15    CELERY_RESULT_BACKEND: redis://redis:6379/3  
16  
17  services:  
18    postgres:  
19      image: postgres:alpine  
20      restart: unless-stopped  
21      networks:  
22        - private  
23      volumes:  
24        - pgdata:/var/lib/postgresql/data  
25      environment:  
26        <<: *pgcreds  
27    redis:  
28      image: redis:alpine
```

docker-compose.yml

Файл `docker-compose.yml`

В первой строке всегда описывается **версия Docker Compose**, который сможет запустить все необходимые контейнеры:

```
docker-compose.yml  
  
1 version: "3.9"  
2 ...
```



А затем идут **секции** с описанием желаемого состояния.

Секция volumes

В этой секции описываем желаемые **volume**. При этом **Docker Compose** все создаст автоматически - мы лишь указываем желаемую конфигурацию.

docker-compose.yml

```
version: "3.9"
```

```
volumes:
```

```
  pgdata:
```

```
  redis_data:
```

```
    external: true
```

Комментарий

Здесь так и оставляем пустоту - будут использованы параметры по умолчанию.

Секция volumes

В этой секции описываем желаемые **volume**. При этом **Docker Compose** все создаст автоматически - мы лишь указываем желаемую конфигурацию.

docker-compose.yml

```
version: "3.9"
```

volumes:

```
  pgdata:
```

```
  redis_data:
```

```
    external: true
```

Комментарий

Использовать внешний volume, то есть не создавать его и не удалять, а рассчитывать на существующий.

Секция networks

В этой секции описываем **желаемые сети**. При этом Docker Compose также все создаст автоматически.

docker-compose.yml

```
version: "3.9"
```

```
volumes:
```

```
...
```

networks:

```
  frontend:
```

```
    driver: host
```

```
  backend:
```

Комментарий

Так и оставляем пустоту - будут использованы параметры по умолчанию.

Секция networks

Стоит отметить, что если **не описывать эту секцию**, то будет создана сеть по умолчанию.

```
<название папки>_default
```

и все контейнеры будут подключены к **этой сети**.

Секция services

Здесь будем описывать **желаемое состояние сервисов** (контейнеров).

`docker-compose.yml`

```
version: "3.9"
```

```
volumes:
```

```
...
```

```
networks:
```

```
...
```

```
services:
```

```
  db:
```

```
  ...
```

Комментарий

Описание каждого сервиса начинается с указания названия, а внутри уже описываются параметры сервиса.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

services:

db:

image:

restart:

networks:

volumes:

depends_on:

ports:

env_file:

Комментарий

Как уже писалось - название сервиса.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image: postgres:alpine
    restart:
    networks:
    volumes:
    depends_on:
    ports:
    env_file:
```

Комментарий

Образ, который необходимо запустить.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart: unless-stopped
    networks:
    volumes:
    depends_on:
    ports:
    env_file:
```

Комментарий

Необязательный параметр. Политика перезапуска, то есть правила, до каких пор надо перезапускать контейнер. Перезапуск может потребоваться, если, например, наш контейнер упал из-за ошибки.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart:
    networks:
      - backend
    volumes:
    depends_on:
    ports:
    env_file:
```

Комментарий

Необязательный параметр. Список сетей для контейнера - здесь указываем сети из секции networks.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart:
    networks:
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./logs:/var/log
    depends_on:
    ports:
    env_file:
```

Комментарий

Необязательный параметр. Здесь указываем volume из секции volumes или монтируем напрямую в хостовую систему.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart:
    networks:
    volumes:
    depends_on:
      - redis
    ports:
    env_file:
```

Комментарий

Необязательный параметр. От каких сервисов зависим, то есть текущий сервис не запустится, пока не запустятся указанные здесь сервисы.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart:
    networks:
    volumes:
    depends_on:
    ports:
      - "5432:5432"
    env_file:
```

Комментарий

Необязательный параметр. Указываем проброс портов.

Секция services

Разберем построчно, что включает в себя **секция services**.

docker-compose.yml

```
services:
  db:
    image:
    restart:
    networks:
    volumes:
    depends_on:
    ports:
    env_file:
      -.env
```

Комментарий

Необязательный параметр. Указываем файлы с переменными окружения для контейнера.

Секция services

docker-compose.yml

services:

db: # название сервиса

image: postgres:alpine # образ, который необходимо запустить

restart: unless-stopped # (необязательное) политика перезапуска, например если наш
контейнер упал из-за ошибки

networks: # (необязательное) сети для контейнера - здесь указываем сети из секции networks
- backend

volumes: # (необязательное) volume для контейнера - здесь указываем volume из секции volumes
или монтируем напрямую в хостовую систему

- pgdata:/var/lib/postgresql/data # используем volume pgdata

- ./logs:/var/log # используем монтирование в папку logs рядом с docker-compose.yml

depends_on: # (необязательное) от каких сервисов зависим, то есть текущий сервис не запустится,
пока не запустятся указанные здесь сервисы

- redis

ports: # (необязательное) указываем проброс портов

- "48881:5432"

env_file: # (необязательное) указываем файлы с переменными окружения для контейнера

- .env



Управляющие команды

Команды в терминале

Описав конфигурационный файл, **запустить всю систему** можно всего одной командой (надо ее выполнять в той же папке, где лежит файл **docker-compose.yml**):

```
docker-compose up
```

Если хочется запустить процесс в отрыве от терминала:

```
docker-compose up -d
```

Еще команды

Все команды **Docker Compose** начинаются с **docker-compose**.

Аргументы команды	Описание
ps	статус и сводная информация по всем контейнерам из текущей конфигурации
start	запустить все остановленные контейнеры
start <name>	запустить остановленный контейнер с именем <name>
stop	остановить все запущенные контейнеры
stop <name>	остановить запущенный контейнер с именем <name>
down	остановить все запущенные контейнеры и удалить все компоненты (контейнеры и сети)
rm	удалить все остановленные контейнеры
logs	вывести в терминал логи по всем контейнерам
logs <name>	вывести в терминал логи контейнера с именем <name>

Дополнительные материалы

Документация по **Docker Compose** с примером:

- <https://docs.docker.com/compose/gettingstarted/>.





Итоги

Итоги

- **Узнали**, что такое оркестрация контейнеров;
- **Узнали** про Docker Compose;
- **Рассмотрели** его конфигурационный файл и управляющие команды.

Домашнее задание

- Обязательного домашнего задания по лекции **нет**
- Есть **необязательное** [домашнее задание](#)

Вопросы по работе с Docker Compose задаём в чате Slack!

**Задавайте вопросы и
пишите отзыв о лекции!**

Александр Ульянов