

Docker compose

Инструмент для управления несколькими контейнерами при помощи одного файла.

Настройка прав на директорию

Создаем группу, которой будут предоставлены права на управление

```
sudo groupadd docker-users
sudo usermod -aG docker-users user1
sudo usermod -aG docker-users user2
```

Создаем директорию проекта

```
sudo mkdir /var/dockerimages
sudo chown root:docker-users /var/dockerimages      # Владелец – root, группа – docker-users
sudo chmod 775 /var/dockerimages                  # Права: rwxrwxr-x
```

Права на создаваемые директории

```
sudo chmod g+s /var/dockerimages
```

Добавление пользователей в группу Docker. В этом случае пользователи получают полный доступ к Docker.

```
sudo usermod -aG docker user1 user2
```

Политики перезапуска

Применение compose файла с политикой restart always/unless-stopped нужно быть внимательным. Повторное применение может создать копию, которая будет постоянно перезагружаться и забивать ресурсы. Для диагностики docker ps Если такое произошло, то

```
docker update --restart=no 1945fa12ce27
```

Это обновит политику перезагрузки для контейнера и позволит разобраться в проблеме.

Консольные команды:

Основная команда	Доп. параметры	Описание
------------------	----------------	----------

docker compose up		Запуск контейнеров через Compose файл. Вывод журнальных записей объединяется в один поток. Создает образы, если они не существовали ранее
	-d	запуск в фоновом режиме
	-f	определяет имя compose файла
	& в конце	возвращает консоль, но продолжает выводить логи
docker compose build		Пересоздание образов из Dockerfile.
docker compose ps		Вывод информации о состоянии контейнеров. Работает только в контексте файла в текущей директории.
docker compose run		Одноразовый запуск с выполнением одной команды (не в качестве сервиса). Также запускаются все контейнеры, с которыми должны быть установлены соединения, если не задан аргумент --no-deps. (Команды, передаваемые через run, заменяют команды, определенные в файле конфигурации сервиса. Кроме того, по умолчанию команда run не создает портов, определенных в файле конфигурации сервиса.)
docker compose top		процессы во всех контейнерах
docker compose stop		Останов контейнеров без их удаления
docker compose logs		Вывод журнальных записей с цветной подсветкой, объединенный для всех контейнеров
docker compose restart		запускает остановленные контейнеры
docker compose rm		Удаление остановленных контейнеров.
	-v	удаляет тома, управляемые механизмом Docker
docker compose down		это stop+rm
	--volumes	удаляет и тома

	--rmi all	удаляет образы
--	-----------	----------------

Формат файла docker-compose.yml

Имя файла docker-compose.yml

Язык yaml. Уровни вложенности определяются пробелами, параметр:значение, - список Compose перекрывает настройки docker файла при пересечении.

Если микросервисы в одной сети (networks), то могут взаимодействовать по имени.

комментарий, пустые строки игнорируются

сначала создается сеть и тома, затем сервисы.

Блоки конфигурации:

```
services #настройки запускаемых образов
  firstapp:
    build: . #Расположение dockerfile, в данном случае в текущей папке
    command: python app.py #исполняемая команда (точка входа)
    hostname: your-name
    ports:
      - target: 8080 #внутренний порт
        published: 5001 #внешний порт
    environment:
      - DEBUG=${DEBUG}
    networks:
      - counter-net #название подключенной сети, т е может быть разные подключенные сети.
        Подключенные к одной сети контейнеры видят друг друга по имени. Д б определена в networks
    volumes:
      - type: volume
        source: counter-vol #Д б определена в volumes
        target: /app
networks #Сеть
  counter-net:
    over-net: #имя сети
    driver: overlay #драйвер
    attachable: true #разрешено подключаться из других докер контейнеров
volumes #Тома
```

Внешние переменные окружения

По умолчанию файл .env Формат файла:

```
VERSION=v1.3
PG_USERNAME="superadmin"
```

Кавычки:

- Тип кавычек после = определяет внешние кавычки, которые убираются.
- Внутри строки не может быть внешних неэкранированных кавычек
- Одинарные не разыменовывают \$ (и скорее всего остальные спец. символы), двойные разыменовывают
- Одинарные поддерживают multiline

Использование внутри yaml файла (без доступа во время выполнения):

```
services:
  web:
    image: webapp:${VERSION}
```

Для использования внутри сервиса необходимо создать переменную внутри environment

```
services:
  db:
    image: postgres
    environment:
      POSTGRES_USER: ${PG_USERNAME}
```

Использование различных файлов с переменными окружения

```
docker compose --env-file <file> up
```

Предварительный вывод итогового yaml файла:

```
docker compose config
```

Приоритет переменных (от высокого к низкому):

1. Файл Compose.
2. Переменные среды оболочки.
3. Файл среды.
4. Dockerfile.
5. Переменная не определена.

Именованние контейнеров.

В случае использования image, имя контейнера берется из имени сервиса. В случае build <имя проекта>-<имя сервиса> Переменные в имени сервиса не поддерживаются.

Политика перезагрузки контейнера

- always - всегда
- uness-stopped - если контейнер был остановлен, затем перезагрузилась служба Docker - он не будет запущен. При перезагрузке работающего контейнера с unless-stopped будет запущен.
- on-failure - перезагрузка при аварийном завершении работы. Контейнер запускается если был остановлен и служба Docker была перезапущена.

```
services:  
  postgres:  
    image: ${PROJECT_NAME}_db:${VERSION}  
    restart: always
```

Revision #11

Created 30 June 2024 09:56:23 by Admin

Updated 15 August 2025 16:38:12 by Admin