Docker compose

Инструмент для управления несколькими контейнерами при помощи одного файла.

Рекомендуемая директория:

```
/opt/docker/
```

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 получат 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
	-f	Вывод логов в реальном времени на консоль

Основная команда	Доп. параметры	Описание
sudo truncate -s 0 /var/lib/docker/containers/*/*-json.log		Очистка всех файлов логов без удаления. Но оно ломает вывод в реальном времени, поэтому нужно после рестартовать контейнеры. docker compose restart
docker compose restart		Перезапускает 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.yaml

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 комманда (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 yml 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}
```

Чтобы добавить значение по умолчанию, нужно поставить двоеточие тире значение (без пробелов).

```
      AUTH_TOKEN_PUBLIC_KEY: ${AUTH_TOKEN_PUBLIC_KEY:------BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA+HICDZMfimMIiGdd0dPr\n2N3zxHOJqiVXuvUj/aBBS7c876
```

```
0rJUh4344sP/0Gid558yH/v1MtzZ0R9w09v6gb\nSNrSiNwIUNGPhbnm9Jf+kMezsI/rkcIdoVm3KJ8CFUYy6MRPzW7iJm
IVRHBb82v1\nmAYCCQFU6IRtFIa9hQ8wedWwxqXZekSNS+6NB37dmmQB9kz2E3MY+KRLigOh4i3p\nCZ1ti3HVvMa9Rlgk
9FmoWExzub5ECChqwm+vn8yFXjYW7kUSClcV2xx4nbQWqjrR\nyyLk6W2BNOuCFVvz0j+5XkpLAt7ljtVZc+572HBEKpF
2mAdmAip5NzeDhKqucJ+\nZQIDAQAB\n-----END PUBLIC KEY-----}
      AUTH_SIGNUP_DISABLED: ${AUTH_SIGNUP_DISABLED:-false}
    ports:
      - ${UI_PORT:-8080}:8080
```

В принципе удобная функция, но я столкнулся с заполненным файлом со значениями по умолчанию, однако не было примера .env

Использование различных файлов с переменными окружения

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

Предварительный вывод итогового yml 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.

Именование 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
```

Архивация и восстановление

Очень желательно остановить контейнеры

```
docker compose down
```

Архивируем проект

```
cd ..
tar -czvf project-backup.tar.gz myproject/
```

Определяем факт наличия примонтированных томов (не в виде директорий)

```
docker volume ls
```

Элементы с драйвером local определяют наличие этих томов:

```
[dockeruser@TVM214 datalens]$ docker volume ls
DRIVER    VOLUME NAME
local     datalens_db-postgres
```

Сохраняем тома (повторяем операцию для всех томов)

```
docker run --rm \
  -v datalens_db-postgres:/data \
  -v $(pwd):/backup \
  alpine tar -czvf /backup/db-postgres.tar.gz -C /data .
```

Переносим на новую машину, остальные команды для новой машины. Распаковываем проект

```
tar -xzvf project-backup.tar.gz
cd myproject
```

Восстановливаем volume до поднятия докера (docker compose up) — иначе compose создаст пустой volume и Postgres инициализируется с нуля:

```
docker volume create datalens_db-postgres


docker run --rm \
```

```
  -v datalens_db-postgres:/data \
  -v $(pwd):/backup \
  alpine tar -xzvf /backup/db-postgres.tar.gz -C /data
```

Поднимаем контейнеры

```
docker compose up -d
```

Имя volume на новой машине должно совпасть с тем, что создаст compose. Если папку проекта назовёшь иначе — префикс будет другим. Можно зафиксировать имя проекта в docker-compose.yml:

```
name: myproject
```

Тогда volume всегда будет datalens_db-postgres, независимо от имени папки.

Для баз данных естественно можно использовать стандартные средства архивации / восстановления и не париться с volume.

Revision #16
Created 30 June 2024 09:56:23 by Admin
Updated 1 October 2026 01:35:00 by Admin