UIT

Оркестрация приложений с Docker Swarm

Латышев Григорий
29.05.2024

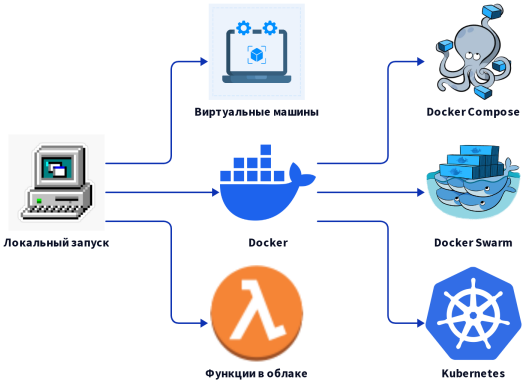
Содержание

- Требования к среде выполнения
- Docker Compose vs Docker Swarm
- Создание кластера и сервисов
- Управление состоянием
- Рекомендации

Требования к среде выполнения

- **Простота**
  - *Локальная разработка*
  - *Администрирование*
  - *Решение проблем*
- **Изолированность**
- **Переносимость**
- **Управляемость**
  - *Переменные окружения*
  - *Управление версиями*
  - *Логирование и мониторинг*
- **Масштабируемость**
  - *Вертикальное масштабирование*
  - *Горизонтальное масштабирование*

Эволюция деплоя приложений



Docker Compose vs Docker Swarm

Качество	Docker Compose	Docker Swarm
Сложность	легко	чуть сложнее
Поддержка зависимостей между сервисами	Да	Нет
Поддержка репликации	Нет	Да
Поддержка кластеризации	Нет	Да
Прозрачное обновления сервисов	Нет	Да
Балансировщик нагрузки	Нет	Да
Дополнительные требования к приложению	Нет	Да

Требования Docker Swarm к приложению и деплою:

- Обработка недоступности других сервисов (БД, Redis, ...)
- Наличие healthcheck у сервисов
- Обработка сигналов `SIGTERM`/`SIGKILL`

Инициализация кластера

Docker Swarm необходимо предварительно инициализировать

```
$ docker swarm init

Swarm initialized: current node (r3ozii9s3hhwsv03p79gqzieb) is now a manager.

    To add a worker to this swarm, run the following command:

docker swarm join --token SWMTKN-1-0q5ehdrl1649kfbpkma9ss5s1nlw1wvcmhuj1pcnnqnkquf9b9-8
    ftp1oriocxj9f50xi90wgj9z 192.168.0.103:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the
    instructions.
```

Удаление инстанса из кластера

```
$ docker swarm leave
```

Добавление сервисов

Сначала добавим сервис вручную

```
$ docker service create --name=nginx nginx

w6w7uq1rfhug4kmhsrqh4jb3a
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service w6w7uq1rfhug4kmhsrqh4jb3a converged
```

```
$ docker service ls
ID             NAME            MODE          REPLICAS   IMAGE          PORTS
w6w7uq1rfhug   nginx           replicated    1/1        nginx:latest
```

Довольно утомительно выполнять для всех сервисов по-отдельности, не правда ли?

Docker Swarm и compose файлы

Compose файлы используются *в основном* только для первичной инициализации кластера

```
$ docker stack deploy test -c docker-compose.yml
Creating network test_default
Creating service test_nginx
```

```
services:
    nginx:
        image: nginx:latest
        deploy:
            replicas: 3
            update_config:
                parallelism: 1
                failure_action: rollback // pause, continue
                order: start-first // stop-first
                delay: 5s
            endpoint_mode: dnsrr // vip
```

Swarm не будет обновлять сервис, если **шаблон запуска** не изменился

Docker Swarm и compose файлы: одиночные задания

Часто бывает необходимо запустить контейнер **один раз** при деплое или обновлении приложений в кластере

```
services:
    job:
        image: alpine
        deploy:
            mode: global-job
```

```
$ docker service ls
ID             NAME          MODE          REPLICAS                IMAGE           PORTS
qv535u46tsvc   test_job      global job    0/0 (1/1 completed)     alpine:latest
```

Подходит для выполнения миграций базы данных, создания индексов, очистки кэша и т.д.

Управление сервисами

Основные команды для управление сервисами

```
$ docker service

Usage:  docker service COMMAND

Manage Swarm services

Commands:
  create      Create a new service
  inspect     Display detailed information on one or more services
  logs        Fetch the logs of a service or task
  ls          List services
  ps          List the tasks of one or more services
  rm          Remove one or more services
  rollback    Revert changes to a service configuration
  scale       Scale one or multiple replicated services
  update      Update a service
```

Управление сервисами: inspect

Принимает те же аргументы, что и `docker inspect`
Выводит информацию о сервисе в JSON формате

```
$ docker service inspect nginx
[
    {
        "ID": "3uj6ue7i38369edtc9zq86bap",
        "Version": {
            "Index": 281
        },
        "CreatedAt": "2024-05-25T15:43:13.961333368Z",
        "UpdatedAt": "2024-05-25T15:43:50.922669814Z",
        "Spec": {
            "Name": "nginx",
...
```

В выводе также доступны шаблон запуска и предыдущая конфигурация

Управление сервисами: logs

Принимает те же аргументы, что и `docker logs`

```
$ docker service logs nginx

nginx.1.l1cmmvdtwgrr:home    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty,
     will attempt to perform configuration
```

- Название сервиса: `nginx`
- Номер реплики: `1`
- Идентификатор контейнера: `l1cmmvdtwgrr`
- Инстанс: `home`

Управление сервисами: `ls`

```
$ docker service ls
ID             NAME      MODE          REPLICAS   IMAGE          PORTS
l2dmhictze8d   alpine    replicated    1/1        alpine:latest
3uj6ue7i3836   nginx     replicated    3/3        nginx:latest
```

- Идентификатор сервиса: `3uj6ue7i3836`
- Название сервиса: `nginx`
- Тип сервиса: `replicated`
- Статус реплик: 3 (успешно запущенных) из 3 (запланированных)
- Образ: `nginx:latest`

Управление сервисами: scale

Делается вручную или из compose файла

```
$ docker service scale nginx=5
nginx scaled to 5
overall progress: 5 out of 5 tasks
1/5: running   [==================================================>]
2/5: running   [==================================================>]
3/5: running   [==================================================>]
4/5: running   [==================================================>]
5/5: running   [==================================================>]
verify: Service nginx converged
```

```
$ docker service ls
ID             NAME            MODE         REPLICAS   IMAGE          PORTS
w6w7uq1rfhug   nginx           replicated   5/5        nginx:latest
```

Управление сервисами: update --env-add/rm

Типичная задача — обновление переменные окружения сервиса

```
$ docker service update --env-add SOMEVAR=foobar nginx
1/1: running   [==================================================>]
verify: Service nginx converged
```

```
$ docker service inspect nginx
[{
    "Spec": {
        "Name": "nginx",
        "TaskTemplate": {
            "ContainerSpec": {
                "Image": "nginx:latest@sha256:a484...",
                "Env": [
                    "SOMEVAR=foobar"
                ],
...
```

Управление сервисами: update --image

Типичная задача — обновление образа сервиса

```
$ docker service update --image=nginx:1.25.5 nginx
1/1: running   [==================================================>]
verify: Service nginx converged
```

```
$ docker service inspect nginx
[{
    "Spec": {
        "TaskTemplate": {
            "ContainerSpec": {
                "Image": "nginx:1.25.5@sha256:
                    a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c",
...
```

- Jenkins: `env.BUILD_NUMBER`
- Gitlab CI: `$CI_PIPELINE_IID` и `$CI_COMMIT_SHA`

Swarm не будет обновлять сервис, если образ не изменился

Рекомендации

- Первоначальная инициализация
  `docker stack deploy`
- Добавление сервисов
  - `docker stack deploy`
  - `docker service create`
- Обновление образов
  - `docker stack deploy`
  - `docker service update --image`
- Обновление переменных окружения
  `docker service update --env-add`
- Масштабирование
  - Вертикальное масштабирование
    `docker service scale`
  - Горизонтальное масштабирование
    - `docker swarm join`
    - требуется кластер по бизнес-требованиям — лучше использовать Kubernetes