

# Docker Swarm

Кластеризация приложений, упрощенный K8s.

Типы Nodes (хост Docker в Swarm кластере):

- manager: управление состоянием кластера и распределение задач по workers
- workers: получают и выполняют задачи

Конфигурация и состояние кластера хранится в распределенной хранимой в ОП БД, реплицированной по всем manager. Сервис (service) является атомарным элементом для управления. Сверху накручиваются фишки типа масштабирования, постоянного обновления и восстановления.

Для Node желательно настройка dns имен (в моем случае manager1, manager2, worker1, worker2)

Роли можно совмещать на одном VPS, актуально для маленьких кластеров.

## Инициализация кластера

Инициализация первого manager - добавление остальных manager - добавление worker.  
Инициализация первого manager:

```
docker swarm init
```

Затем при помощи команд ... join-token смотрим инструкцию по подключению node.

При размещении где-то, должны быть доступны следующие порты:

- 2377/tcp: для защищенного взаимодействия между нодами
- 7946/tcp and udp: взаимодействие менеджеров
- 4789/udp: VXLAN-based overlay сеть

## Доступность кластера

Один manager активен в каждый момент времени. Это Leader manager. Остальные (Follower managers) проксируют команды на лидера. Ситуации split-brain (в результате сбоя сети при котором одинаковое количество manager осталось в каждом сегменте и последующего восстановления связи) необходимо избегать. Желательно 3-5 manager.

Подключение manager после перезагрузки / сбоя сети может привести к проблемам. Поэтому желательно установить правило блокировки при перезагрузке.

```
docker swarm update --autolock=true
```

Она выдаст ключ разблокировки. После перезагрузки потребуется разблокировать

```
docker swarm unlock
```

Есть нюанс: для работы кластера требуется доступность более половины manager. Поэтому кластер из 2 manager точно плохой вариант)

### Сервисы (пользовательские приложения)

По умолчанию исполняются на всех нодах. Для исключения manager нужно ввести команду

```
docker node update --availability drain mgr1
```

В списке node Active поменяется на Drain

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE
m1bhltzuvvzipoetl7b26m3j	manager1.bobrobotirk.ru	Ready	Drain	Leader	28.0.1
yjmubzqzvrvbhqaw70rlorzk *	manager2.bobrobotirk.ru	Ready	Drain	Reachable	28.0.1
3s0jzf6fcw9ik5g9sqlrpmngo	worker1.bobrobotirk.ru	Ready	Active		28.0.1

Сервисы создаются через интерактивные команды или через описание (compose file + доп. настройки).

Стандартный режим создания - реплика (количество, распределено по активным worker). Есть глобальный режим (mode global) при котором на каждом worker создается по одной реплике.

### Архивация и восстановление

Ключ разблокировки очень важен. При его утере и перезагрузке всех manager node хер что сделаешь.

```
tar -czvf swarm.bkp /var/lib/docker/swarm/  
  
rm -r /var/lib/docker/swarm  
tar -zxvf swarm.bkp -C /  
docker swarm init --force-new-cluster
```

После восстановления директорий обязательна реинициализация кластера.

### Примеры

## 5 реплик, доступ через любую ноду

```
docker service create --name web-fe -p 8080:8080 --replicas 5 nigelpoulton/ddd-book:web0.1  
или  
docker service create --name web-fe -p 8080:8080 --mode global nigelpoulton/ddd-book:web0.1  
docker service rm web-fe
```

## Через overlay сеть + обновление

```
docker network create -d overlay uber-net  
docker service create --name web-fe --network uber-net -p 8080:8080 --replicas 5 nigelpoulton/ddd-book:web0.1  
#без указания сети, запрос на manager:8080 обрабатываться не будет  
  
docker service update --image nigelpoulton/ddd-book:web0.2 --update-parallelism 2 --update-delay 20s web-fe
```

## Compose файл

```
networks:  
  counter-net:  
    driver: overlay  
    driver_opts:  
      encrypted: 'yes'  
volumes:  
  counter-vol:  
  
services:  
  web-fe:  
    image: nigelpoulton/ddd-book:swarm-app  
    mem_limit: 250m  
    command: python app.py  
    deploy:  
      replicas: 4  
      update_config:  
        parallelism: 2  
        delay: 10s  
        failure_action: rollback  
    placement:  
      constraints:  
        - 'node.role == worker'  
    restart_policy:
```

```
condition: on-failure
delay: 5s
max_attempts: 3
window: 120s
networks:
- counter-net
ports:
- published: 5001
  target: 8080
volumes:
- type: volume
  source: counter-vol
  target: /app
redis:
image: "redis:alpine"
networks:
counter-net:
```

### Основные команды

Команда	Доп. пар.	Описание
docker swam init		Инициализация первого manager кластера
	--advertise-addr 10.0.0.1:2377	Необязательный параметр. Нужен если есть внешний балансировщик нагрузки - тогда адрес балансировщика, и указать listen-addr.
	--listen-addr 10.0.0.1:2377	Необязательный параметр. Нужен если много ip адресов.
docker node ls		Список node в кластере
docker swarm join-token	worker	Инструкция по подключению worker
	manager	Инструкция по подключению manager
docker swarm leave		Исключение node из кластера
docker swarm update	--autolock=true	Блокировка после перезагрузки / потере связи

Команда	Доп. пар.	Описание
	--availability drain name_manager	Исключение manager из исполнения клиентских приложений.
docker swarm unlock		Разблокировка manager
docker service	ls	список сервисов
	ps name	список работающих контейнеров с именем name
	inspect --pretty name	детальная информация по сервису name
	scale name=10	Изменение количества реплик сервиса name в режиме реального времени
	rm name	Удалить сервис
	update --image imname --update-parallelism 2 --update-delay 20s name	Обновление сервиса name до образа imname
	logs nameserv	Отобразить логи сервиса nameserv
docker stack	deploy -c name.yml nameofstack	<div>Создает стэк nameofstack из файла name.yml</div> <div>docker stack deploy -c compose.yaml ddd</div> <div>Также используется для обновления существующего сервиса при обновлении compose файла</div>
	rm nameofstack	Удаление стэка nameofstack
	ls	Список
	ps	Детализация