Работа с docker-swarm

До начала работ, убедитесь что у вас установлен docker. Docker-swarm является встроенным в дистрибутив docker решением по оркестрации, поэтому предпринимать дополнительных действий по его установке не потрубуется.
Если docker установлен, переходите к следующему разделу по работе с docker-swarm

--

Запускаем кластер docker swarm:

```
$ docker swarm init
```

```
filipp@filipp-notebook:~$ docker swarm init
Swarm initialized: current node (z77a9tx2xf0jzzw4nn1b2a1sb) is now a manage
r.

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

    docker swarm join --token SWMTKN-1-1uy75uqdbi7n1yc3qjkm6qpycvncmsr9dxmj
vchpp2tda3t9qz-7c082a8o3wazhigod54sy23qn 192.168.18.11:2377

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

Кластер docker-swarm успешно проинициализирован

Вывполним команду просмотра статуса узлов запущенного кластера:

```
$ docker node ls
```

```
filipp@filipp-notebook:~$ docker node ls
ID                            HOSTNAME          STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
z77a9tx2xf0jzzw4nn1b2a1sb *   filipp-notebook   Ready     Active         Leader           20.10.17
```

Убеждаемся в работоспособности узлов кластера docker-swarm

Запускаем деплой сервиса `nginx` в docker swarm:

```
$ docker service create --name nginx -p 8080:80 nginx:alpine
```

```
filipp@filipp-notebook:~$ docker service create --name nginx -p 8080:80 nginx:alpine
yf0ajaqka5j4qy082c33tkie8
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
```

Сервис успешно запущен

Выполним команду просмотра запущенных docker-контейнеров:

```
$ docker ps
```

```
filipp@filipp-notebook:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
     NAMES
2810d436afbe   nginx:alpine   "/docker-entrypoint.…"   15 seconds ago   Up 14 seconds   80/tc
p    nginx.1.ijcw5mpnaj7tntolph8fuyd2p
```

docker-контейнер успешно запущен

Проверяем доступность сервиса, отправляя команду `curl`:

```
$ curl localhost:8080
```

```
filipp@filipp-notebook:~$ curl localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Убеждаемся в работоспособности сервиса

Запускаем деплой еще одного сервиса (`redis` версии `3.0.5`):

```
$ docker service create --name redis redis:3.0.5
```

```
filipp@filipp-notebook:~$ docker service create --name redis redis:3.0.5
dwxz2glp47uj5qrttbv6jaxjm
overall progress: 1 out of 1 tasks
1/1: running
verify: Service converged
```

Сервис успешно запущен

Выполним команду просмотра запущенных docker-контейнеров:

```
$ docker ps
```

```
filipp@filipp-notebook:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS
   PORTS      NAMES
2fda176742c3   redis:3.0.5    "/entrypoint.sh redi…"   About a minute ago   Up About a minute
   6379/tcp   redis.1.8z47jebktchz1ze20gdn1h4xc
2810d436afbe   nginx:alpine   "/docker-entrypoint.…"   6 minutes ago        Up 6 minutes
   80/tcp     nginx.1.ijcw5mpnaj7tntolph8fuyd2p
```

Оба сервиса (`nginx` и `redis`) успешно запуещны

Выполним команду для просмотра списка запущенных в docker-swarm сервисов:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   1/1        redis:3.0.5
```

Сервисы (`nginx` и `redis`) в docker-swarm успешно запущены

--

Проинспектируем сервис redis в docker swarm в `pretty`-формате вывода:

```
$ docker service inspect --pretty redis
```

```
filipp@filipp-notebook:~$ docker service inspect --pretty redis

ID:             dwxz2glp47uj5qrttbv6jaxjm
Name:           redis
Service Mode:   Replicated
 Replicas:      1
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         redis:3.0.5@sha256:f8829e00d95672c48c60f468329d6693c4
08682e
 Init:          false
Resources:
Endpoint Mode:  vip
```

Обратим внимание на количество репликаций сервиса

Выполним команду скалирования репликаций сервиса `redis`:

```
$ docker service scale redis=2
```

```
filipp@filipp-notebook:~$ docker service scale redis=2
redis scaled to 2
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
```

Репликации сервиса успешно скалирован до двух

Выполним команду просмотра запущенных docker-контейнеров:

```
$ docker ps
```

```
filipp@filipp-notebook:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
      NAMES
be5da7aa2b6c   redis:3.0.5    "/entrypoint.sh redi…"   17 seconds ago   Up 16 seconds   6379/
tcp   redis.2.vjr68cj7q10zdqhlvlluk6v6i
2fda176742c3   redis:3.0.5    "/entrypoint.sh redi…"   6 minutes ago    Up 6 minutes    6379/
tcp   redis.1.8z47jebktchz1ze20gdn1h4xc
2810d436afbe   nginx:alpine   "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   80/tc
p     nginx.1.ijcw5mpnaj7tntolph8fuyd2p
```

Обращаем внимание, что контейнеров `redis` теперь два

Повторим команду для просмотра списка запущенных в docker-swarm сервисов:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   2/2        redis:3.0.5
```

В выводе отображена информация об изменившимся количестве репликаций сервиса `redis`

--

Обновим сервис `redis` на новую версию используемого образа (была `3.0.5`, станет `3.0.6`):

```
$ docker service update --image redis:3.0.6 redis
```

```
filipp@filipp-notebook:~$ docker service update --image redis:3.0.6 redi
redis
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
```

Обращаем внимание на минимальную задержку между последовательным обновлением двух репликаций сервиса

Выполним команду для просмотра списка запущенных в docker-swarm сервисов, чтобы убедиться в успешности обновления версии `redis`:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   2/2        redis:3.0.6
```

Версия `redis` успешно обновлена до `3.0.6`

--

Модифицируем сервис для введения задержки при обновлении реплик `redis` при совершении `rolling-update`:

```
$ docker service update --update-delay 10s redis
```

```
filipp@filipp-notebook:~$ docker service update --update-delay 10s redis
redis
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
```

Команда успешно применена

Проинспектируем сервис `redis` в docker-swarm в `pretty`-формате вывода:

```
$ docker service inspect --pretty redis
```

```
filipp@filipp-notebook:~$  docker service inspect --pretty redis

ID:             dwxz2glp47uj5qrttbv6jaxjm
Name:           redis
Service Mode:   Replicated
 Replicas:      2
Placement:
UpdateConfig:
 Parallelism:   1
 Delay:         10s
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         redis:3.0.6@sha256:6a692a76c2081888b589e26e6ec83574311
d69842
 Init:          false
Resources:
Endpoint Mode:  vip
```

Изменение успешно применено к сервису (см. `UpdateConfig.Delay`)

Обновляем сервис `redis` на новую версию используемого образа (была `3.0.6`, станет `3.0.7`) с применением задержки при обновлении реплик (в `10` секунд):

```
$ docker service update --image redis:3.0.7 redis
```

```
filipp@filipp-notebook:~$ docker service update --image redis:3.0.7 redis
redis
overall progress: 1 out of 2 tasks
1/2: running
2/2:
```

Наблюдаем задержку между обновлениями репликаций сервиса

Выполним команду для просмотра списка запущенных в docker-swarm сервисов, чтобы убедиться в успешности обновления версии `redis`:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   2/2        redis:3.0.7
```

Версия `redis` успешно обновлена до `3.0.7`

--

Отработаем `rollback` обновления на последнюю стабильную версию:

```
$ docker service rollback redis
```

```
filipp@filipp-notebook:~$ docker service rollback redis
redis
rollback: manually requested rollback
overall progress: rolling back update: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
```

`rollback` изменений последнего `rolling-update` был произведен

Выполним команду для просмотра списка запущенных в docker-swarm сервисов, чтобы убедиться в успешности `rollback` обновления `redis` назад к версии `3.0.6`:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   2/2        redis:3.0.6
```

`rollback` успешно выполнен

Дескалируем сервис `redis` до `0` репликаций:

```
$ docker service scale redis=0
```

```
filipp@filipp-notebook:~$ docker service scale redis=0
redis scaled to 0
overall progress: 0 out of 0 tasks
verify: Service converged
```

Скалирование успешно завершно

Выполним команду для просмотра списка запущенных в docker-swarm сервисов, чтобы убедиться в успешности дескалирования сервиса `redis`:

```
$ docker service ls
```

```
filipp@filipp-notebook:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
yf0ajaqka5j4   nginx     replicated   1/1        nginx:alpine   *:8080->80/tcp
dwxz2glp47uj   redis     replicated   0/0        redis:3.0.6
```

Сервис успещно дескалирован до `0` репликаций

Выполним команду просмотра запущенных docker-контейнеров:

```
$ docker ps
```

```
filipp@filipp-notebook:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS
     NAMES
2810d436afbe   nginx:alpine   "/docker-entrypoint.…"   33 minutes ago   Up 33 minutes
p    nginx.1.ijcw5mpnaj7tntolph8fuyd2p
```

docker-контейнеры `redis` отсутствуют в выводе команды просмотра запущенных контейнеров

--

Выполним команду удаления сервисов `nginx` и `redis` в кластере docker-swarm:

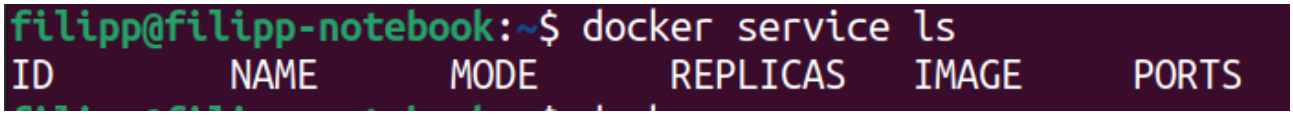
```
$ docker service rm redis nginx
```

```
filipp@filipp-notebook:~$ docker service rm redis nginx
redis
nginx
```

Сервисы успешно удалены

Выполним команду просмотра запущенных в docker-swarm сервисов, чтобы убедиться в отсутствии запущенных сервисов `nginx` и `redis`:

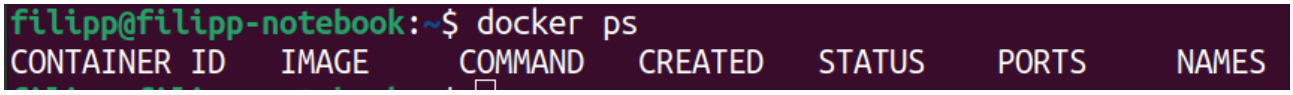
```
$ docker service ls
```



Сервисы отсутствуют в выводе команда просмотра запущенных сервисов

Выполним команду просмотра запущенных docker-контейнеров:

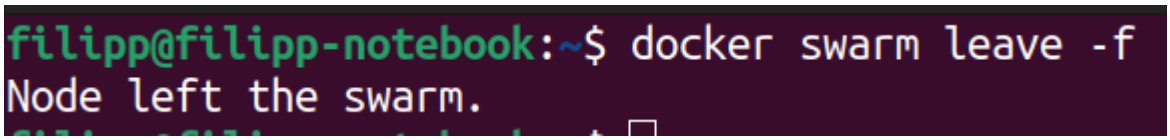
```
$ docker ps
```



Убеждаемся в том, что ни одного docker-контейнера на запущено

Выводим из кластера узел кластера (при условии, что кластер состоял из одного узла, кластер будет остановлен и удален):

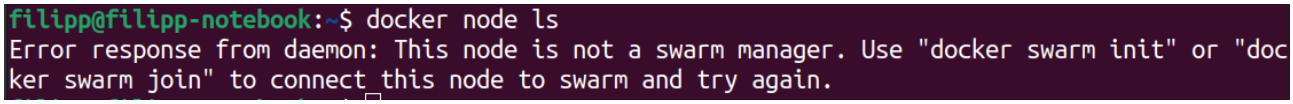
```
$ docker swarm leave -f
```



Единственный узел кластера был выведен из эксплуатации

Вывполним команду просмотра статуса узлов запущенного кластера:

```
$ docker node ls
```



Убеждаемся, что кластер docker-swarm был успешно остановлен и удален

--