#Docker

Docker run

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

```
[ OPTIONS ]

-it – интерактивный режим. Перейти в контейнер и запустить внутри контейнера ко
-d – запустить контейнер в фоне (демоном) и вывести его ID
-p port_localhost:port_docker_image – порты из докера на локалхост
-e «TZ=Europe/Moscow» – указываем нашему контейнеру timezone
-h HOSTNAME – присвоить имя хоста контейнеру
– link <имя контейнера> – связать контейнеры с другим
-v /local/path:/container/path/ – прокидываем в контейнер докера директорию с л
--name CONTAINERNAME – присвоить имя нашему контейнеру
--restart=[no/on-failure/always/unless-stopped] – варианты перезапуска контейне
```

```
# Для больше читабельности можно при переносе строки ставить символ \
docker run \
–restart=always -it \
```

Простые действия с контейнерами

Создание контейнера

```
docker create -t -i eon01/infinite --name <CONTAINERNAME or CONTAINERID>
```

Переименование контейнера

```
docker rename <OLD CONTAINERNAME> <NEW CONTAINERNAME>
```

Удаление контейнера

```
docker rename <OLD CONTAINERNAME> <NEW CONTAINERNAME>
```

Обновление контейнера

```
docker update --cpu-shares 512 -m 300M <CONTAINERNAME or CONTAINERID>
```

Запуск и обстановка контейнеров

Запуск остановленного контейнера

```
docker start <CONTAINERNAME>
```

Остановка

```
docker stop <CONTAINERNAME>
```

Перезагрузка

```
docker restart <CONTAINERNAME>
```

Пауза (приостановка всех процессов контейнера)

```
docker pause <CONTAINERNAME>
```

Снятие паузы

```
docker unpause <CONTAINERNAME>
```

Блокировка (до остановки контейнера)

```
docker wait <CONTAINERNAME>
```

Отправка SIGKILL (завершающего сигнала)

```
docker kill <CONTAINERNAME>
```

Отправка другого сигнала

```
docker kill -s HUP <CONTAINERNAME>
```

Подключение к существующему контейнеру

```
docker attach <CONTAINERNAME>
```

Выход через Ctrl+p+q

Информация о контейнере

Работающие контейнеры

Вывести работающие контейнеры

```
docker ps
```

Вывести все контейнеры

```
docker ps -a
```

Логи контейнера

```
docker logs
```

```
docker logs <CONTAINERNAME or CONTAINERID>
```

Информация о контейнере

```
# Все параметры контейнера
docker inspect <CONTAINERNAME or CONTAINERID>
```

```
# Вывести величину конкретного параметра
docker inspect --format '{{ .NetworkSettings.IPAddress }}' $(docker ps -q)
```

События контейнера

```
docker events <CONTAINERNAME or CONTAINERID>
```

Публичные порты

```
docker port <CONTAINERNAME or CONTAINERID>
```

Выполняющиеся процессы

```
docker top <CONTAINERNAME or CONTAINERID>
```

Использование ресурсов

```
docker stats <CONTAINERNAME or CONTAINERID>
```

Изменения в файлах или директориях файловой системы контейнера

```
docker diff <CONTAINERNAME or CONTAINERID>
```

Работа с контейнерами

```
`docker compose -f docker-compose.yml up`
```

Зайти в уже запущенный контейнер.

(точнее выполнить команду внутри контейнера)

```
docker exec -it name_of_container /bin/bash
```

Запустить контейнер и открыть в нём bash

```
docker run -it -d --name my_container CONTAINER_ID /bin/bash
```

Копирование файлов внутрь контейнера.

```
docker cp some_files.conf docker_container:/home/docker/
```

При смене путей, можно копировать из контейнера.

Работа с registry

Вход в реестр

```
# Вход на докерзаб
docker login

# Вход в WebUI локального registry
docker login localhost:8080
```

Выход из реестра

```
# Докерхаб
docker logout

# Локальный registry
docker logout localhost:8080
```

Поиск образа

```
# Простой поиск
docker search nginx

# Поиск с фильтрами
docker search nginx -- filter stars=3 --no-trunc busybox
```

Pull (выгрузка из реестра) образа

```
# Выгрузка из Докерхаба
docker pull nginx

# Из локального registry
docker pull eon01/nginx localhost:5000/myadmin/nginx
```

Push (загрузка в реестр) образа

```
# В докерхаб
docker push eon01/nginx

# В локальный registry
docker push eon01/nginx localhost:5000/myadmin/nginx
```

Работа с образами

Список образов

```
docker images
```

Создание образов

(файл опциями сборки образа), учитывая что мы находимся в папке где лежит этот файл. Через ключ -t назначаем имя нашему образу. Точка в конце означает что Dockerfile лежит в текущей директории.

```
docker build -t my_docker .
```

```
docker build .
```

```
docker build github.com/creack/docker-firefox
```

```
docker build - < Dockerfile
```

```
docker build - < context.tar.gz
```

```
docker build -t eon/infinite .
```

```
docker build -f myOtherDockerfile .
```

```
curl example.com/remote/Dockerfile | docker build -f - .
```

Удаление образа

```
docker rmi imagename (example: nginx)
```

Загрузка репозитория в tar (из файла или стандартного ввода)

```
docker load < ubuntu.tar.gz
```

```
docker load --input ubuntu.tar
```

Сохранение образа в tar-архив

```
docker save busybox > ubuntu.tar
```

Просмотр истории образа

```
docker history
```

Создание образа из контейнера

```
docker commit nginx
```

Тегирование образа

```
docker tag nginx eon01/nginx
```

Push (загрузка в реестр) образа

```
docker push eon01/nginx
```

Работа с томами

- Если монтируем пустой том с хоста, а в контейнере уже есть файлы, то они скопируются в том
- Если монтируем с хоста том с файлами, то они окажутся в контейнере
- Если мы монтируем не пустой том с хоста, а в контейнере по этому пути уже есть файлы, то они будут скрыты
- Можно монтировать с хоста любые файлы. В том числе и служебные. Например сокет docker. Что бы получился docker-in-docker (dind)

Создание тома

```
docker volume create <VOLUME-NAME>
```

Монтируем с хоста в контейнер

```
docker run --mount source=<VOLUME-NAME>,target=/path/to/folder/in/container -d
```

Монтируем с контейнера на хост

```
docker run --mount type=bind,source=/host/folder,target=/container/folder -d <I
```

Посмотреть настройки тома

```
docker volumes inspect <VOLUME-NAME>
```

Вывести список всех томов с их названиями.

```
docker volume ls
```

Удаление volumes по названию

```
docker volume rm <VOLUME-NAME>
```

Сети

Создание сети

```
docker network create -d overlay MyOverlayNetwork
```

```
docker network create -d bridge MyBridgeNetwork
```

```
docker network create -d overlay \
  --subnet=192.168.0.0/16 \
  --subnet=192.170.0.0/16 \
  --gateway=192.168.0.100 \
  --gateway=192.170.0.100 \
  --ip-range=192.168.1.0/24 \
  --aux-address="my-router=192.168.1.5" --aux-address="my-switch=192.168.1.6" \
  --aux-address="my-printer=192.170.1.5" --aux-address="my-nas=192.170.1.6" \
  MyOverlayNetwork
```

Удаление сети

```
docker network rm MyOverlayNetwork
```

Список сетей

```
docker network ls
```

Получение информации о сети

```
docker network inspect MyOverlayNetwork
```

Подключение работающего контейнера к сети

```
docker network connect MyOverlayNetwork nginx
```

Подключение контейнера к сети при его запуске

```
docker run -it -d --network=MyOverlayNetwork nginx
```

Отключение контейнера от сети

```
docker network disconnect MyOverlayNetwork nginx
```

Чистка мусора

Показать образы и контейнеры

```
# Вывести все образы (images)
docker images -a

# Вывести все неиспользуемые образы
docker images -f dangling=true

# Выведет список запущенных контейнеров, если добавить ключ -a, выведет список
docker ps
```

Удаление образов (images)

```
# Для удаления используется команда docker rmi с добавлением ИД или тега, напри
# с ключом --force удалит контейнер и образ
docker rmi abb461727af5

# Удаление всех образов
docker rmi $(docker images -a -q)

# Удаление всех неиспользуемых образов
docker images prune
docker rmi $(docker images -f dangling=true -q)

# Удаление всех неиспользуемых (не связанных с контейнерами) образов:
```

```
# Если добавить к команде ключ -a, то произойдет удаление всех остановленных ко
docker system prune

# Удаление всех образов без тегов
docker rmi -f $(docker images | grep "^<none>" | awk "{print $3}")

# Удаление всех образов
docker rmi $(docker images -a -q)
```

Удаление контейнеров (containers)

```
# Для удаления контейнера, его необходимо сначала остановить командой ниже с ук
docker stop CONTAINER_ID

# Для удаления контейнера используется команда docker rm с добавлением ИД или н
docker rm CONTAINER_ID

# Удаление контейнера и его тома (volume)
docker rm -v CONTAINER_ID

# Удаление всех контейнеров со статусом exited
docker rm $(docker ps -a -f status=exited -q)

# Удаление всех остановленных контейнеров
docker container prune
docker rm `docker ps -a -q`

# Удаление контейнеров, остановленных более суток назад
docker container prune --filter "until=24h"

# Остановка и удаление всех контейнеров
docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q)
```

Удаление томов (volumes)

```
#Вывести список всех томов с их названиями.
docker volume ls

# Удаление volumes по названию
docker rm <volume_name>
```

```
# Вывести список всех томов не связанных с контейнерами
docker volume ls -f dangling=true

# Удаление томов (volumes) несвязанных с контейнерами
docker volume prune
docker volume rm $(docker volume ls -f dangling=true -q)

# Удаление неиспользуемых (dangling) томов по фильтру
docker volume prune --filter "label!=keep"
```

Удаление сетей (networks)

```
# Вывести список всех сетей с их ИД и названиями.
docker network ls

# Для удаления используется команда  с добавлением ИД или названия:
docker network rm NETWORK_ID

# Удалит все сети не используемые хотя бы одним контейнером.
docker network prune
```

Удаление всех неиспользуемых объектов

```
docker system prune

# По умолчанию для Docker 17.06.1+ тома не удаляются. Чтобы удалились и они тож
docker system prune --volumes
```

+++++

Новый билд:

```
docker-compose up --build
```

Остановки и удаления всех контейнеров, сетей и заного билд:

```
docker-compose down &&  docker-compose up --build
```

Перезапустить все контейнеры:

```
docker-compose restart
```

Посмотреть список всех активных контейнеров Docker:

```
docker ps
```

Чтобы увидеть список всех контейнеров, включая остановленные, используйте флаг `-a` (или `--all`):

```
docker ps -a
```

Удаления неиспользуемых образов:

```
docker image prune --all --force
```

`docker image prune`: Эта команда начинает процесс удаления неиспользуемых образов.
`--all`: Этот флаг указывает Docker удалить все неиспользуемые образы, включая те, которые связаны с контейнерами, которые не активны в данный момент.
`--force`: Этот флаг указывает Docker на принудительное выполнение удаления образов, без запроса подтверждения. Если его не использовать, Docker запросит подтверждение перед удалением.
Таким образом, команда `docker image prune --all --force` удалит все неиспользуемые Docker-образы без дополнительных запросов на подтверждение. (Образы нельзя будет восстановить)

```
docker volume prune --all --force
```

```
docker builder prune --all --force
```

Остановки и удаления всех контейнеров и сетей:

```
docker compose down
```

Она также удаляет все связанные с контейнерами ресурсы, такие как объемы данных.

docker compose