# Полезные команды Docker

## remove all stopped containers

docker rm $(docker ps -a -q)

## remove all untagged images

docker rmi $(docker images | grep "^<none>" | awk "{print $3}")

## Kill all running containers

docker kill $(docker ps -q)

## Delete all stopped containers (including data-only containers)

docker rm $(docker ps -a -q)

## Delete all 'untagged/dangling' (<none>) images

docker rmi $(docker images -q -f dangling=true)

## Delete ALL images

docker rmi $(docker images -q)

## remove unused containers

docker rm $(docker ps -q -f status=exited)

## nice docker stats

docker stats $(docker ps --format '{{.Names}}')

## delete volumes

docker volume ls -qf dangling=true | xargs -r docker volume rm

## copy files to and from docker container

docker cp <container>:<path> <localpath>

## log into running container from another shell

docker exec -it <container-name> /bin/bash

## create an image from a container and give it a name

docker commit <image*name*> *docker tag* <*image*id> <new*image*name>

## run a container from an image (image*name can be repo*name/tag_name)

docker run -it --name <container*name*> <*image*name> /bin/bash

## run a container while mounting pwd on host (must be under ~) to the

/var/shared directory in container

docker run -v $(pwd):/var/shared --name <container*name*> *-it* <*image*name> /bin/bash

## run a container from an image and delete it when done

docker run -it --rm <image_name> /bin/bash

## start an existing container and log in

docker start -ai <container-name>

### rename container

docker rename old*name new*name

### Docker Compose Commands

sudo docker-compose build
docker-compose stop conv
docker-compose rm conv
docker-compose build conv
docker-compose up conv

---

Check the website.

# Содержание




- Содержание
- Установка
- Реестры и репозитории Docker
- Первые действия с контейнерами
- Запуск и остановка контейнеров
- Получение информации о контейнерах
- Сеть
- Очистка Docker
- Docker Swarm
- Заметки


# Установка

## Linux

Больше информации здесь

```
curl -sSL https://get.docker.com/ | sh
```

## Mac

Больше информации здесь

Скачайте dmg по этой ссылке.

```
https://download.docker.com/mac/stable/Docker.dmg
```

## Windows

Больше информации здесь

Используйте MSI-инсталлятор:

```
https://download.docker.com/win/stable/InstallDocker.msi
```

# Реестры и репозитории Docker

## Вход в реестр

```
docker login
```

```
docker login localhost:8080
```

### Выход из реестра.

```
docker logout
```

```
docker logout localhost:8080
```

### Поиск образа

```
docker search nginx
```

```
docker search --filter stars=3 --no-trunc nginx
```

### Pull (выгрузка из реестра) образа

```
docker image pull nginx
```

```
docker image pull eon01/nginx localhost:5000/myadmin/nginx
```

### Push (загрузка в реестр) образа

```
docker image push eon01/nginx
```

```
docker image push eon01/nginx localhost:5000/myadmin/nginx
```

## Первые действия с контейнерами

### Создание и запуск простого контейнера

> - *Запустите образ ubuntu:latest*
> - *Свяжите порт `80` **КОНТЕЙНЕРА** с портом `3000` **ХОСТА***
> - *Смонтируйте текущую директорию в `/data` на КОНТЕЙНЕРЕ*
> - *Заметка: на **windows** вы должны изменить `-v ${PWD}:/data` на `-v "C:\Data":/data`*

```
docker container run --name infinite -it -p 3000:80 -v ${PWD}:/data ubuntu:latest
```

### Создание контейнера

```
docker container create -t -i eon01/infinite --name infinite
```

### Запуск контейнера

```
docker container run -it --name infinite -d eon01/infinite
```

### Переименование контейнера

```
docker container rename infinite infinity
```

### Удаление контейнера

```
docker container rm infinite
```

### Обновление контейнера

```
docker container update --cpu-shares 512 -m 300M infinite
```

## Запуск и остановка контейнеров

### Запуск

```
docker container start nginx
```

### Остановка

```
docker container stop nginx
```

### Перезапуск

```
docker container restart nginx
```

### Пауза (приостановка всех процессов контейнера)

```
docker container pause nginx
```

### Снятие паузы

```
docker container unpause nginx
```

### Блокировка (до остановки контейнера)

```
docker container wait nginx
```

### Отправка SIGKILL (завершающего сигнала)

```
docker container kill nginx
```

### Отправка другого сигнала

```
docker container kill -s HUP nginx
```

### Подключение к существующему контейнеру

```
docker container attach nginx
```

## Получение информации о контейнерах

### Работающие контейнеры

```
docker container ls
docker container ls -a
```

### Логи контейнера

```
docker logs infinite
```

### Следовать логам контейнера (вывод логов с обновлениями в реальном времени)

```
docker container logs infinite -f
```

### Информация о контейнере

```
docker container inspect infinite
docker container inspect --format '{{ .NetworkSettings.IPAddress }}' $(docker ps -q)
```

### События контейнера

```
docker system events infinite
```

### Публичные порты

```
docker container port infinite
```

### Выполняющиеся процессы

```
docker container top infinite
```

### Использование ресурсов

```
docker container stats infinite
```

### Изменения в файлах или директориях файловой системы контейнера

```
docker container diff infinite
```

### Управление образами

### Список образов

```
docker image ls
```

### Создание образов

```
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 image rm nginx
```

### Загрузка репозитория в tar (из файла или стандартного ввода)

```
docker image load < ubuntu.tar.gz
docker image load --input ubuntu.tar
```

### Сохранение образа в tar-архив

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

### Просмотр истории образа

```
docker image history
```

### Создание образа из контейнера

```
docker container commit nginx
```

### Тегирование образа

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

### Push (загрузка в реестр) образа

```
docker image push eon01/nginx
```

# Сеть

## 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 container run -it -d --network=MyOverlayNetwork nginx
```

## Отключение контейнера от сети

```
docker network disconnect MyOverlayNetwork nginx
```

## Exposing Ports

Используя Dockerfile, вы можете раскрыть порт в контейнере используя:

```
EXPOSE <port_number>
```

You can also map порт контейнера to порт хоста используя:

Например,

```
docker run -p $HOST_PORT:$CONTAINER_PORT --name infinite -t infinite
```

# Очистка Docker

## Удаление работающего контейнера

```
docker container rm nginx
```

## Удаление контейнера и его тома (volume)

```
docker container rm -v nginx
```

## Удаление всех контейнеров со статусом exited

```
docker container rm $(docker container ls -a -f status=exited -q)
```

### Удаление всех остановленных контейнеров

```
docker container rm `docker container ls -a -q`
```

### Удаление образа

```
docker image rm nginx
```

### Удаление неиспользуемых (dangling) образов

```
docker image rm $(docker image ls -f dangling=true -q)
```

### Удаление всех образов

```
docker image rm $(docker image ls -a -q)
```

### Удаление всех образов без тегов

```
docker image rm -f $(docker image ls | grep "^<none>" | awk "{print $3}")
```

### Остановка и удаление всех контейнеров

```
docker container stop $(docker container ls -a -q) && docker container rm $(docker container ls -a -q)
```

### Удаление неиспользуемых (dangling) томов

```
docker volume rm $(docker volume ls -f dangling=true -q)
```

### Removing all unused (containers, images, networks and volumes)

```
docker system prune -f
```

### Полная очистка

```
docker system prune -a
```

## Docker Swarm

### Установка Docker Swarm

```
curl -ssl https://get.docker.com | bash
```

### Инициализация Swarm

```
docker swarm init --advertise-addr 192.168.10.1
```

### Подключение рабочего узла (worker) к Swarm

```
docker swarm join-token worker
```

### Подключение управляющего узла (manager) к Swarm

```
docker swarm join-token manager
```

### Список сервисов

```
docker service ls
```

### Список узлов

```
docker node ls
```

### Создание сервиса

```
docker service create --name vote -p 8080:80 instavote/vote
```

### Список заданий Swarm

```
docker service ps
```

### Масштабирование сервиса

```
docker service scale vote=3
```

### Обновление сервиса

```
docker service update --image instavote/vote:movies vote
```

```
docker service update --force --update-parallelism 1 --update-delay 30s nginx
```

```
docker service update --update-parallelism 5--update-delay 2s --image instavote/vote:indent vote
```

```
docker service update --limit-cpu 2 nginx
```

```
docker service update --replicas=5 nginx
```

## Заметки

Эта работа впервые была опубликована в Painless Docker Course