

# Docker - справка

## 1 Основные команды

### 1.1 Общие

Создать пользователя для запуска docker:

```
$useradd -m -s /bin/bash <user>
$usermod -uG docker <user>
$su - <user>

$docker --version    # - просмотр версии docker
$docker version     # - подробная информация версии docker

$service docker status    # - проверить статус docker

$docker system prune -a --volumes    # - удалить ВСЕ данные в docker
```

### 1.2 Образы

```
$docker pull <image>    # - загрузить образ
$docker images    # - список всех образов
$docker rmi <image>    # - удалить образ
$docker image inspect <image>    # - информация об образе
```

Варианты значений тэгов:

latest, <version>, alpine (минимального размера)

```
$docker save <image>    # - Сохранить image в tar архив
$docker load    # - Загрузить image из tar архива или через stdin
```

### 1.3 Запуск из образов

```
$docker run <image>    # - запустить образ, будет загружен если его нет
$docker run hello-world    # - запустить базовый контейнер hello-world
$docker run <image> <command>    # - запустить образ и в нем команду
```

Дополнительные параметры:

- |                            |  |
|----------------------------|--|
| <b>-d</b>                  | запустить в режиме “detach mode”                       |
| <b>-it</b>                 | запустить интерактивно (i - interactive, t - terminal) |
| <b>--rm</b>                | удалить контейнер после завершения                     |
| <b>--name &lt;name&gt;</b> | задать имя для нового контейнера                       |

## 1.4 Контейнеры

```
$docker ps      # - список запущенных контейнеров  
$docker ps -a    # - список всех контейнеров  
$docker rm <container>    # - удалить контейнер  
$docker container prune    # - удалить все контейнеры  
$docker start <container>    # - запустить контейнер
```

Выполнить интерактивно команду в контейнере:

```
$docker exec -it <container> <command>

$docker pause container    # - поставить на паузу
$docker unpause container  # - возобновить работу
$docker stop container     # - остановить работу
$docker kill container     # - убить

$docker logs <container>   # - Последние log'и контейнера
$docker logs -f <container>  # - life-logs

$docker inspect <container>  # - просмотр подробной информации
$docker container inspect <container>  # - или так
$docker stats <container>    # - статистика использования ресурсов
```

Пример команды:

```
$ docker run -d --rm --name MyContainer ubuntu:20.04 echo "Hello"
```

## 2 Port Mapping

Все запросы на <port1> сервера пробрасывать в контейнер на порт <port2>:

```
$ docker run -p <port1>:<port2> <container>
```

## 3 Environment variables

Задать в контейнере переменную окружения:

```
$docker run -e <var>=<name>
```

```
env    # - получить все системные переменные окружения  
export <var>=<value>   # - задать переменную окружения
```

## 4 Docker Volumes

### Host Volumes

Монтировать фиксированный каталог сервера в каталог контейнера:

```
$docker run -v <dir_server>:<dir_container>[:ro] ...  
:ro    # - read-only
```

### Anonymous Volumes

Монтировать каталог сервера в каталог контейнера:

```
$docker run -v <dir_container> ...
```

Каталог сервера находится в:

```
/var/lib/docker/volumes/<HASH>/_data
```

### Named Volumes

Монтировать именованный volume в каталог контейнера:

```
$docker run -v <volume_name>:<dir_container> ...
```

Каталог сервера находится в:

```
/var/lib/docker/volumes/<volume_name>/_data
```

Команды управления volumes:

```
$docker volume ls    # - Список volumes  
$docker volume create <volume_name>    # - Создать named volume  
$docker volume rm <volume_name>    # - Удалить volume
```

## 5 Docker Network

Создается default сеть типа bridge:

- Сетевой интерфейс docker0 (172.17.0.1/16)
- bridge docker0: 172.17.0.0/16

Если контейнеры в одной сети (кроме default), то они доступны по dns по именам.

Кроме bridge возможные типы сети:

```
--network=host    # - получают адрес хоста  
--network=none    # - без сети
```

Также доступны типы сети:

```
macvlan    # - каждый контейнер получает свой mac адрес  
ipvlan     # - один и тот же mac адрес у всех контейнеров  
overlay    # - Docker Swarm Cluster
```

Создать сеть с типом Bridge:

```
$docker network create --driver bridge <net>  
--driver host    # - с типом host  
--driver none    # - без сети
```

Запустить контейнер в сети:

```
$docker run --net <net> ...
```

```
$docker network ls    # - список сетей  
$docker network inspect <net>    # - информация о сети
```

Пример создания сети с заданными адресами:

```
$docker network create -d bridge --subnet 192.168.10.0/24 \  
--gateway 192.168.10.1 <net>  
$docker network rm <net>    # - удалить сеть
```

Docker образ с установленными сетевыми инструментами:

```
nicolaka/netshoot
```

Подключить работающий контейнер к сети:

```
$docker network connect <net> <container>
```

Отключить от сети:

```
$docker network disconnect <net> <container>
```

Создать сеть с адресами из того же диапазона, что и сам сервер:

```
$docker network create -d macvlan \  
--subnet 192.168.100.0/24 --gateway 192.168.100.1 \  
--ip-range 192.168.100.99/32 -o parent=ens18 <net>
```

Присвоить фиксированный ip адрес:

```
$docker run ... --ip <ip_address> --net <net> ...
```

## 6 Создание контейнеров, Dockerfile

Структура Dockerfile:

Раздел	Пример
Базовый образ	FROM ubuntu:22.04
Описание образа	LABEL author=...
Команды	RUN apt update RUN apt install ...
Рабочие директории	WORKDIR <dir>
Файлы	COPY <src> <dst> RUN chmod +x
Указание переменных	END var=value
Порты	EXPOSE 80
Команды при запуске контейнера	CMD [<cmd>, <arg1>, ...] ENTRYPOINT [<cmd>, <arg1>, ...]

Пример простого Dockerfile:

```
FROM UBUNTU:22.04  
CMD ["echo", "Hello"]
```

Последнюю строку можно записать: CMD ECHO "Hello"

Создать образ, в каталоге должен быть файл Dockerfile:

```
$docker build <dir>
```

Создать образ указав путь к Dockerfile:

```
$docker build -f <docker_file>
```

Изменить имя и тэг у существующего образа:

```
$docker tag <image> <name>:<tag>
```

Создать образ и задать имя и тэг:

```
$docker build -t <name>:<tag> <dir>
```

Отличия команд при запуске:

ENTRYPOINT # - неизменяемая команда

CMD # - команда может быть изменена при запуске контейнера

Пример Dockerfile для nginx:

```
FROM UBUNTU:22.04  
RUN apt-get update  
RUN apt-get install nginx -y
```

```
CMD [“nginx”, “-g”, “daemon off”]
```

Открытие портов в контейнер (EXPOSE ports:

```
EXPOSE 80  
EXPOSE 443/TCP
```

Пробросить случайные порты сервера в контейнер

```
$docker -P ...
```

## 7 Docker Compose

Сравнение запуска docker и используя docker-compose.yaml:

Docker CLI	docker-compose.yaml
<pre>docker run  nginx:stable --name my_nginx  -v ....: -v ....:  -e ... -e ...  -p 80:80 -p 443:443</pre>	<pre>version: "3.5" services:   web-server:     image: nginx:stable     container_name: my_nginx     volumes:       - '/opt/web/html:/var/www/html'       - '/opt/web/pics:/var/www/pics'     environment:       - NGINX_HOST=web       - NGINX_PORT=80     ports:       - "80:80"       - "443:443"     restart: unless-stopped # always, on-failure     networks:       default:         driver: bridge         name: webnet</pre>

Запустить через Docker Compose:

```
$docker compose up [-d]
```

Остановить:

```
$docker compose stop
```

Вместо:

```
image: <image_name>
```

Можно собирать из Dockerfile:

```
build .
```

## 8 Docker Portainer

Удобный web интерфейс для управления Docker. Упрощает жизнь с Docker.  
Есть бесплатная версия - Community Edition.

Статья по установке Docker Portainer:  
<https://losst.pro/ustanovka-docker-portainer>