[**docker info**](https://www.docker.com/)

полная информация о docker: версия docker,контейнеры, образы, информация о сервере

**docker -v**

версия в краткой форме

**docker version**

версия в полной форме

**docker --help**

справка

**docker** сommand **--help**

справка по команде

**docker login**

login on docker hub

у аутентифицированной CLI в docker info есть секция Username: satnetuser001

**docker logout**

logout on docker hub

**docker run hello-world**

скачать и запустит тестовый образ

**docker search imageName**

поиск пакета

**docker image pull** imageName

**docker pull** imageName

загрузить образ

**docker image ls**

**docker images**

список скачанных (локальных) образов

**docker image history** imageName

**docker history** imageName

информация (история) образа

**docker image rm** imageName/imageID

**docker rmi** imageName/imageID

удалить образ

**docker image prune -a**

удалить ВСЕ образы

**docker system prune -a**

удалить ВСЕ образы, контейнеры, сети, кэш

**docker container run** imageName

**docker run** imageName

создаёт контейнер из образа, скачивает образ если его нет локально

docker run/pull/... imageName**:tag**

тэг указывает на версию пакета, например latest - последняя версия

**!Внимание!** Опции команды **run**, например **-it**, изменяют поведение контейнера не только

при его создании, а при каждом его вызове:

https://docs.docker.com/reference/cli/docker/container/run/

**docker run -it** imageName

создаёт контейнер из образа и подключится к нему в терминале

!Внимание, опция -it не дает закрываться контейнеру после его выполнения

**docker run -d** imageName

создаёт контейнер из образа в фоне (не подключается к нему)

**docker run --name** futureContainerNameimageName/imageID

задать имя контейнеру при создании

**docker run --rm** imageName/imageID

удалить контейнер после выхода

**docker run -p** hostPort**:**containerPortimageName

**docker run -p** 8080:80 nginx

создать контейнер из образа nginx с

системным портом localhost:8080 и связать его с внутриконтейнерным портом 80 nginx

**docker run -v** pathToIndexDirectoryOnHost**:**/usr/share/nginx/html nginx

**docker run -v** ${PWD}**:**/usr/share/nginx/html nginx

монтирование тома хоста к nginx в контейнере (и создание самого контейнера)

${PWD} - print work directory - монтирование необходимо производить в

директории с index файлом хоста

**docker run -v** ${PWD}:/usr/share/nginx/html **-p** 8080:80 **-d --name** containerName nginx

монтирование тома хоста + монтирование порта + задание имени контейнера nginx

**docker run \**

**--name** temp **\**

**-v** ${PWD}:/usr/share/nginx/html **\**

**-p** 8082:80 **\**

**-d \**

**--rm \**

**nginx**

разбить команду на несколько строк

**docker container rename** currentContainerName/containerID newContainerName

**docker rename** currentContainerName/containerID newContainerName

переименовать контейнер

**docker container ls**

**docker ps**

список запущенных контейнеров

ps - process status - сокращение, используемое в различных операционных системах для

обозначения запущенных процессов или программ.

**docker stats**

подробная информация о запущенных контейнерах

**docker attach** containerId

подключится к запущенному контейнеру в терминале

**docker container ls -a**

**docker ps -a**

список всех контейнеров

**docker ps -l**

последний созданный контейнер

**docker container start** containerId

**docker start** containerId

запустить контейнер

**docker start $(docker ps -a -q)**

запустить все контейнеры

**docker start -i** containerId

запустить контейнер и подключится к нему в терминале

!Внимание, у команды есть особенность в работе: если контейнер был создан без опций -it,

то эта команда не запрещает закрытие контейнера после выполнения его работы (после

подключения к контейнеру тебя может сразу выбрасывать)

**docker container stop** containerId

**docker stop** containerId

остановить контейнер

**docker stop $(docker ps -a -q)**

остановить все контейнеры

**docker container kill** containerId

**docker kill** containerId

принудительная остановка контейнера

**docker container pause** containerId

**docker pause** containerId

пауза контейнера

**docker container unpause** containerId

**docker unpause** containerId

восстановление работы приостановленного контейнера

**docker container rm** containerId

**docker rm** containerId

удалить контейнер

**docker container prune**

удалить все остановленные контейнеры

**docker rm $(docker ps -a -q)**

удалить ВСЕ контейнеры, включая запущенные

**docker container inspect** containerId

**docker inspect** containerId

отображение подробной информации об одном или нескольких контейнерах (IP)

**docker inspect** containerId **| grep** IPAddress

отображение информации только с названием IPAddress

**docker container exec** containerId command

**docker exec** containerId command

выполнить команду в работающем контейнере

**docker exec -it** containerId **bash**

запустить командную строку в работающем контейнере и подключится к ней

(in container bash!) **hostname**

имя контейнера

(in container bash!) **hostname -i**

IP контейнера

**docker network ls**

список сетей с краткой информацией

**docker network inspect** networkID/networkName

подробная информация о сети

**docker network rm** networkID/networkName

удалить сеть

**docker network prune**

удалить все неиспользуемые сети

**ctrl+c** or **exit**

выйти из контейнера с завершением процесса

**ctrl+p** и **ctrl+q**

выйти из контейнера не завершая процесс

**docker image build -t** imageName pathToDockerfile

**docker build** options pathToDockerfile

создать образ на основе Dockerfile

**+++++++++STORAGE+++++++++**

https://docs.docker.com/engine/storage/

**Inside container**

любой образ базируется на ОС, например Ubuntu, Debian, Alpine Linux, и наследует её

файловую систему, которую можно использовать для хранения данных.

доступ только из контейнера, удаляются вместе с контейнером.

можно использовать для временного хранения загружаемых на сервер файлов PHP apps.

**Volumes** (тома)

управляемые docker отдельные хранилища данных, которые можно смонтировать внутрь

контейнеров.

любое хранилище доступно неограниченному числу контейнеров, данные хранятся

независимо от контейнера.

можно использовать для базы данных СУБД которой развертывается в контейнере.

**Bind mounts**

каталог или файл localhost смонтированный внутри контейнера.

можно использовать при разработке: Laravel app лежит в папке на localhost и доступно для

IDE, а все изменения тут-же отображаются в контейнере.

строго не рекомендуется для production: нарушение принципа изоляции, уязвимость данных.

**tmpfs mounts**

!Внимание, физически данные сохраняемые этим методом находятся в ОЗУ, возможна утечка

памяти.

доступ только из контейнера, быстрая скорость доступа, удаляются при остановке контейнера.

можно использовать для кеширования данных.

**---------------STORAGE---------------**

**+++++++++OTHER+++++++++**

**docker cp my\_container:/path/to/file /path/on/host**

копировать из контейнера

**---------------OTHER---------------**