Docker

Шпаргалка Docker

dev-ops-notes.ru

Базовое команды

Запуск контейнера в фоне

```
[~]$ docker run -d nginx
```

Запуск контейнера с шелл

```
[~]$ docker run -it ubuntu bash
```

Запуск контейнера, удаляющегося после установки

```
[~]$ docker run -rm ubuntu bash
```

Экспорт порта контейнера

```
[~]$ docker run -p 80:80 -d nginx
```

Задать имя контейнера

```
[~]$ docker run --name frontend nginx
```

Запустить остановленный контейнер

```
[~]$ docker start frontend
```

Остановить контейнер

```
[~]$ docker stop frontend
```

Управление контейнерами

Показать работающие контейнеры

```
[~]$ docker ps
```

Показать работающие и остановленные контейнеры

```
[~]$ docker ps -a
```

Показать метаданные контейнера

```
[~]$ docker inspect c31337
```

Показать доступные локально образы

```
[~]$ docker images
```

Сборка образа

На основе *Dockerfile* в текущей директории

```
[~]$ docker build --tag my-image .
```

"Жесткая" пересборка

```
[~]$ docker build --no-cache my-image .
```

Преобразовать контейнер в образ

```
[~]$ docker commit c31337 my-image
```

Удалить все не используемые образы

```
[~]$ docker rmi $(docker images -q -f "dangling=true")
```

Отладка

Зайти в работающий контейнер

```
[~]$ docker exec -ti c31337 bash
```

Просмотр логов запущенного контейнера

```
[~]$ docker logs -f c31337
```

Показать экспортированные порты

```
[~]$ docker port c31337
```

Удалить все остановленные контейнеры

```
[~]$ docker rm $(docker ps --filter status=expired -q)
```

Показать все контейнеры со специальной меткой

```
[~]$ docker ps --filter label=traefik.backend
```

Показать конкретные метаданные контейнера

```
[~]$ docker inspect -f '{{ .NetworkSettings.IPAddress }}' \
c31337
```

Запустить ничего не делающий контейнер

```
[~]$ docker run -d busybox /bin/sh -c "while true; do sleep
2; done"
```

Диски

Создать локальный диск

```
[~]$ docker volume create --name my-volume
```

Монтирование диска при старте контейнера

```
[~]$ docker run -v my-volume:/data nginx
```

Удалить диск

```
[~]$ docker volume rm my-volume
```

Показать все созданные диски

```
[~]$ docker volume ls
```

Сети

Создать локальную сеть

```
[~]$ docker network create my-net
```

Подключить контейнер к сети при старте

```
[~]$ docker run -d --net my-net nginx
```

Подключить работающий контейнер к сети

```
[~]$ docker network connect my-net c31337
```

Отключить работающий контейнер от сети

```
[~]$ docker network disconnect my-net c31337
```

Управление Docker Machine

Запустить Docker Machine

```
[~]$ docker-machine start machine_name
```

Остановить Docker Machine

```
[~]$ docker-machine stop machine_name
```

Настроить Docker на работу с удаленной Docker Machine

```
[~]$ eval "$(docker-machine env machine_name)"
```

Сборка образа из файла Dockerfile (файл опциями сборки образа), учитывая что мы находимся в папке где лежит этот файл.

```
#$ docker build -t my_docker .
```

Через ключ -t назначаем имя нашему образу
Точка в конце означает что Dockerfile лежит в текущей директории.

Просмотреть существующие на нашем хосте образы докера

```
#$ docker images
```

Просмотреть существующие на нашем хосте контейнеры докера

```
#$ docker ps -a
```

-a — по умолчанию показывает только включённые, вместе с ключом выводит все контейнеры.

`#$ docker rm` — Удалить контейнер
`#$ docker rmi` — Удалить образ, с ключом —force удалит контейнер и образ
`#$ docker rm $(docker ps -a -q)` — Удалить все существующие контейнеры
`#$ sudo docker save -o linux-nginx.img linux-nginx` — Экспортировать образ docker
`#$ sudo docker load -i linux-nginx.img` — Импортировать образ docker

Запустить контейнер и открыть в нём bash

```
#$ docker run -it -d --name my_container 397bd34237 /bin/bash
```

run — команда запуска контейнера
-it — перейти в контейнер и запустить внутри контейнера команду
-d запустить контейнер в фоне и вывести его ID
—name — присвоить имя нашему контейнеру
397bd34237 — имя образа
/bin/bash — выполняемая команда в контейнере

Запустить контейнер который мы уже запускали , с предыдущими параметрами

```
#$ docker start ID-контейнера
```

Bash скрипт для запуска докер образа

```
#!/bin/bash
CONFIG=/home/isavel/ELK+KAFKA/for_docker_image/config/
PATTERN=/home/isavel/ELK+KAFKA/for_docker_image/pattern/
LOG=/home/isavel/ELK+KAFKA/for_docker_image/logs

docker run \ # команда запуска
—restart=always -it \ # в случае падения перезапускать контейнер
-p 443:443 -p 80:80 -p 5601:5601 \ # порты из докера на локалхост
port_localhost:port_docker_image
-e «TZ=Europe/Moscow» \ # указываем нашему контейнеру timezone
-v $CONFIG:/etc/logstash/conf.d/ \ # прокидываем в контейнер докера директорию с локальной машины
-v $LOG:/var/log/my_log/ \ # прокидываем в контейнер докера директорию с локальной машины
-v $PATTERN:/opt/logstash/patterns/ \ # прокидываем в контейнер докера директории с локальной машины
—name sberteh \ # названием нашего контейнера
5e89f9aa5754 \ # ID запускаемого образа
```

Копирование внутрь контейнера

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

Зайти в уже запущенный контейнер (точнее выполнить команду внутри контейнера)

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

Выгрузить контейнер в файл и импортировать его на другой хост докера.

```
docker save -o=file.tar CONTAINER
docker load --input=file.tar
```

Загрузить докер образ в локального репозитория:

```
docker tag [name_image] [repo_name]:[port]/[name_image]
docker push [repo_name]:[port]/[name_image]
```

Linux

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

Mac

Скачайте dmg по этой ссылке:

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

Windows

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 nginx --filter stars=3 --no-trunc busybox
```

Pull (выгрузка из реестра) образа

```
docker pull nginx
```

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

Push (загрузка в реестр) образа

```
docker push eon01/nginx
```

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

Первые действия с контейнерами

Создание контейнера

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

Первый запуск контейнера

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

Переименование контейнера

```
docker rename infinite infinity
```

Удаление контейнера

```
docker rm infinite
```

Обновление контейнера

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

Запуск и остановка контейнеров

Запуск остановленного контейнера

```
docker start nginx
```

Остановка

```
docker stop nginx
```

Перезагрузка

```
docker restart nginx
```

Пауза (приостановка всех процессов контейнера)

```
docker pause nginx
```

Снятие паузы

```
docker unpause nginx
```

Блокировка (до остановки контейнера)

```
docker wait nginx
```

Отправка SIGKILL (завершающего сигнала)

```
docker kill nginx
```

Отправка другого сигнала

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

Подключение к существующему контейнеру

```
docker attach nginx
```

Получение информации о контейнерах

Работающие контейнеры

```
docker ps
```

```
docker ps -a
```

Поиск образа

```
docker search nginx
```

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

Pull (выгрузка из реестра) образа

```
docker pull nginx
```

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

Push (загрузка в реестр) образа

```
docker push eon01/nginx
```

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

Первые действия с контейнерами

Создание контейнера

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

Первый запуск контейнера

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

Изменения в файлах или директориях файловой системы контейнера

```
docker diff infinite
```

Управление образами

Список образов

```
docker images
```

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 rmi 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
```

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

Отключение контейнера от сети

```
docker network disconnect MyOverlayNetwork nginx
```

Очистка Docker

Удаление работающего контейнера

```
docker rm nginx
```

Удаление контейнера и его тома (volume)

```
docker rm -v nginx
```

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"
```

Удаление образа

```
docker rmi nginx
```

Удаление неиспользуемых (dangling) образов

```
docker image prune
```

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

Удаление неиспользуемых (dangling) образов даже с тегами

```
docker image prune -a
```

Удаление всех образов

```
docker rmi $(docker images -a -q)
```

Удаление всех образов без тегов

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

Остановка и удаление всех контейнеров

```
docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q)
```

Удаление неиспользуемых (dangling) томов

```
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 неиспользуемых сетей

```
docker network prune
```

Удаление всех неиспользуемых объектов

```
docker system prune
```

По умолчанию для Docker 17.06.1+ тома не удаляются. Чтобы удалились и они тоже:

```
docker system prune --volumes
```

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:in
dent vote
```

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

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

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

Mac

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

Windows

MSI- : ~~~ https://download.docker.com/win/stable/InstallDocker.msi ~~~

Docker

~~~ docker login ~~~ ~~~ docker login localhost:8080 ~~~

~~~ docker logout ~~~ ~~~ docker logout localhost:8080 ~~~

~~~ docker search nginx ~~~ ~~~ docker search nginx –filter stars=3 –no-trunc busybox ~~~

Pull ( ) ~~~ docker pull nginx ~~~ ~~~ docker pull eon01/nginx localhost:5000/myadmin/nginx ~~~
~~~

Push ( ) ~~~ docker push eon01/nginx ~~~ ~~~ docker push eon01/nginx localhost:5000/myadmin/nginx ~~~

~~~ docker create -t -i eon01/infinite –name infinite ~~~

~~~ docker run -it –name infinite -d eon01/infinite ~~~

~~~ docker rename infinite infinity ~~~

~~~ docker rm infinite ~~~

~~~ docker update –cpu-shares 512 -m 300M infinite ~~~

~~~ docker start nginx ~~~

~~~ docker stop nginx ~~~

~~~ docker restart nginx ~~~

( ) ~~~ docker pause nginx ~~~

~~~ docker unpause nginx ~~~

( ) ~~~ docker wait nginx ~~~

SIGKILL ( ) ~~~ docker kill nginx ~~~

~~~ docker kill -s HUP nginx ~~~ ~~~ docker attach nginx ~~~

~~~ docker ps ~~~ ~~~ docker ps -a ~~~

~~~ docker logs infinite ~~~

~~~ docker inspect infinite ~~~ ~~~ docker inspect –format ‘{{ .NetworkSettings.IPAddress }}’ $(docker ps -q) ~~~

~~~ docker events infinite ~~~

~~~ docker port infinite ~~~

~~~ docker top infinite ~~~

~~~ docker stats infinite ~~~

~~~ docker diff infinite ~~~

~~~ docker images ~~~

~~~ 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 - . ~~~

~~~ docker rmi nginx ~~~

tar ( ) ~~~ docker load < ubuntu.tar.gz ~~~ ~~~ docker load –input ubuntu.tar ~~~

tar- ~~~ docker save busybox > ubuntu.tar ~~~

~~~ docker history ~~~

~~~ docker commit nginx ~~~

~~~ docker tag nginx eon01/nginx ~~~

Push ( ) ~~~ docker push eon01/nginx ~~~

~~~ 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 ~~~

~~~ docker network rm MyOverlayNetwork ~~~

~~~ docker network ls ~~~

~~~ docker network inspect MyOverlayNetwork ~~~

~~~ docker network connect MyOverlayNetwork nginx ~~~

~~~ docker run -it -d –network=MyOverlayNetwork nginx ~~~

~~~ docker network disconnect MyOverlayNetwork nginx ~~~
~~~

Docker

~~~ docker rm nginx ~~~

(volume) ~~~ docker rm -v nginx ~~~

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

~~~ docker container prune ~~~ ~~~ docker rm `docker ps -a -q` ~~~

, ~~~ docker container prune –filter “until=24h” ~~~

~~~ docker rmi nginx ~~~

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

(dangling) ~~~ docker image prune -a ~~~

~~~ docker rmi $(docker images -a -q) ~~~

~~~ docker rmi -f $(docker images | grep “^” | awk “{print $3}”) ~~~

~~~ docker stop $(docker ps -a -q) && docker rm $(docker ps -a -q) ~~~

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

(dangling) ~~~ docker volume prune –filter “label!=keep” ~~~

~~~ docker network prune ~~~

~~~ docker system prune ~~~

Docker 17.06.1+ . : ~~~ docker system prune –volumes ~~~

Docker Swarm

Docker Swarm ~~~ curl -ssl https://get.docker.com | bash ~~~ . .: *Docker 1.12.0+ , . . Docker Swarm Docker Engine (Swarm mode).*

Swarm ~~~ docker swarm init –advertise-addr 192.168.10.1 ~~~

(worker) Swarm ~~~ docker swarm join-token worker ~~~

(manager) Swarm ~~~ docker swarm join-token manager ~~~

~~~ docker service ls ~~~

~~~ docker node ls ~~~

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

Swarm ~~~ docker service ps ~~~

~~~ docker service scale vote=3 ~~~

~~~ 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 ~~~

#

~~~ docker exec nginx env ~~~

docker-compose Dockerfile ~~~ docker cli docker-compose.yml Dockerfile ——————————————————————— –name sample container_name: sample -

-e BASE=asdf environment: ENV BASE=asdf - BASE=asdf

–volume `pwd`/src/:/app/src/ volumes: - - ./src/:/app/src/

–publish 8080:3000 ports: - - “8080:3000”

ubuntu:18.04 image: ubuntu:18.04 FROM ubuntu:18.04

sleep 1000 command: sleep 1000 CMD sleep 1000 ~~~

P.S. https://github.com/eon01/DockerCheatSheet