

Comandos Docker

Índice

Comandos	2
Docker	3
I	3
info	3
V	3
container	3
A	3
attach	3
B	3
C	3
create	4
I	4
inspect	4
L	4
logs	4
ls	4
R	5
run	5
restart	6
rm	6
P	6
prune	6
pause	6
S	7
start	7
stats	7
T	7
top	7
U	7
unpause	7
update	7
image	8
L	8
ls	8
images	8
rmi	8
start	9
stop	9

Volumes no Docker	10
Docker Volume	11
Algumas Explicações Sobre Volumes	11
Comandos	11
create	11
inspect	11
ls	11
rm	11
prune	11
Montando Volumes	11
Utilizando --mount type=bind	11
Utilizando --mount type=volume	12
Links para explicação sobre volumes no docker	12

Documentação Oficial

A documentação oficial da linha de comando do docker está disponível [aqui](#)

Comandos

Docker

docker → comando base para a linha de comando do docker. Se digitado sozinho aparecerá a ajuda do docker. Útil para saber se o docker está instalado na máquina.

service docker status → verifica se o docker está em execução.

service docker start → inicia daemon do docker.

I

info

docker info → Exibe informações gerais sobre o programa docker instalado. Por exemplo exibe a versão do kernel, quantidade de containers, quantidade de imagens, plugins instalados, entre outras informações.

V

docker version → Exibe informações sobre a versão do docker de uma maneira que seja fácil de ler.

container

Gerenciamento de containers

A

attach

docker container attach containerID → Possibilita conectar a um container em execução. Basta passarmos o container id como parâmetro.

TIP

Como sair do container sem encerrar

Se quiser sair do container e deixar ele ainda executando, utilize a combinação de teclas **Ctrl** + **P** + **Q** e não digite nada. Se digitar **exit** sairá do container e o encerrará.

B

docker build -t nomeRepository:tag /caminho/da/pasta/do/dockerfile → cria uma imagem do container a partir de um Dockerfile que deve estar indiciado no caminho no final do comando. Só se indica o caminho da pasta, não precisa indica o caminho completo do arquivo.

C

create

docker container create -ti nomeContainer → Apenas cria um container nomeContainer. Não cria e inicializa o container. O container é apenas criado.

I

inspect

docker container inspect containerID → exibe informações detalhadas do container com containerID. Exemplos de informações exibidas: sobre memória, cpu, etc.

docker container inspect containerID | grep -i cpu → Filtra resultados de informações das configurações de CPU do container.

docker container inspect containerID | grep -i mem → Filtra resultados de informações sobre a memória do container.

L

logs

docker container logs containerID → Exibe STDOUT, ou seja, a saída padrão do container com containerID.

docker container logs -f containerID → Exibe o STDOUT, a saída padrão do container com containerID de forma dinâmica, parecido com o **tail -f**. O terminal vai ficar travado exibindo os logs.

TIP

Como cancelar a exibição dos logs

Para sair utilize **Ctrl + C**.

ls

docker container ls → Lista containers em execução.

Table 1. Significado das Colunas

Coluna	Significado
CONTAINER ID	identificação única do container
IMAGE	a imagem que foi utilizada para a execução do container
COMMAND	o comando em execução
CREATED	quando o container foi criado
STATUS	o seu estado atual
PORT	a porta do container e do host que esse container utiliza

Coluna	Significado
NAMES	o nome do container

docker container ls -a → Lista todos os containers (em execução, finalizados e parados).

docker container ls -q → Exibe somente os CONTAINER ID dos containers em execução.

docker container ls -a -q → Exibe somente os CONTAINER ID de todos os containers (em execução, finalizados e parados).

R

run

TIP

Como sair do container sem encerrar

Ao utilizar opções -it, se quiser sair do container e deixar ele executando ainda utilize a combinação de teclas **Ctrl** + **P** + **Q** e não digite nada. Se digitar **exit** sairá do container e o encerrará.

docker container run [opcoes] [imagem:tag]

Table 2. opcoes podem ser:

Opção	O que faz
-t	disponibiliza um TTY(console) para o container
-i	mantém o STDIN aberto mesmo que você não esteja conectado no container
-d	faz com que o container rode como um daemon, ou seja, sem a interatividade que os outros dois parâmetros fornecem.
--name	nomear um container
-m	limite de memória do container.

docker container run nomeContainer → Executa e cria o novo container nomeContainer.

docker container run hello-world → Executa o container hello-world. Se o docker hello-world não estiver sido criado no host, então ele pesquisa no docker-hub e depois baixa ele para depois executar.

docker container run echo “Ola Mundo” → Executa e cria um novo container do ubuntu, executa o comando echo “Ola Mundo” e depois encerra o container.

docker run nomeContainer comandoParaExecutar → Cria e executa um novo container nomeContainer e depois dentro do container irá executar o comandoParaExecuta.

docker container run -it IMAGE → Executa e cria o novo container IMAGE = image do container que se quer executar. E ainda por causa e -it , vai disponibilizar o terminal do container

compartilhado com o terminal do host, fazendo o container continuar em execução até que o container seja finalizado. Se a imagem já existir ela será usada.

docker container run -it --name NomeContainer IMAGE → Com a adição de **--name nomeContainer** ao comando **docker container run -it IMAGE** agora podemos dar um nome ao container criado que será nomeContainer, pois **--name** define o nome do container.

docker container run --cpu=0.5 --name nomeContainer IMAGE → Com a adição de **--cpu=0.5** ao comando **docker container run** estamos limitando o container com nomeContainer a utilizar 0,5 CPU, ou seja, metade de 1 core.

docker container run -ti -m nM IMAGE → Com a adição de **-m 512M** ao **docker container run** estamos limitando o container que usa a imagem IMAGE, a utilizar n quantide de Mb de memória do host.

docker container run -ti -m 512M --name nomeContainer IMAGE → Com a adição de **-m 512M** ao **docker container run** estamos limitando o container com nomeContainer, que usa a imagem IMAGE, a utilizar 512 Mb de memória do host.

restart

docker container restart containerID → reinicia um ou masi containers com o containerID.

rm

docker container rm containerID → Remove um container parado ,com a respectiva containerID, já existente. A imagem que originou o container ainda continua.

IMPORTANT

O que acontece quanto remove-se o container.

Ao remover o container só renovemos o container, a imagem que foi utilizada para a criação ainda continua no host.

docker container rm containerID -f → Remove um container mesmo em execução com a containerID respectiva. Remove o container forçadamente.

docker container rm \$(docker container ls -a -q) → Remove os containers que estão na lista do comando **docker container ls -a -q**.

P

prune

docker container prune → Rremove todos os container parados.

pause

docker container pause containerID → pausa todos os processos rm um ou mais containers com o container ID.

S

start

docker container stop containerId → Para um (ou mais) container ativos que tem o containerId como CONTAINER ID.

docker container start containerId → Inicia um (ou mais) container parado que tem o containerId como CONTAINER ID.

stats

docker container stats → Exibe informações(estatísticas) em tempo real referentes ao uso de recursos por todos os containers. Você terá informações de consumo de CPU, memória e rede.

TIP

Como sair

Para sair utilize **Ctrl** + **C**

docker container stats containerID → Exibe informações(estatísticas) em tempo real referentes ao uso de recursos de um ou mais container com containerID. Você terá informações de consumo de CPU, memória e rede. Exibir uma transmissão ao vivo de estatísticas de uso de recursos de container(s)

T

top

docker container top containerID → exibe informações sobre os processos em execução de um container com containerID. Colunas exibidas do top: UID, PID, C, STIME, TTY, TIME, COMMAND.

U

unpause

docker container unpause containerID → "despausa" um ou mais container com containerID. Utilizado para desfazer o docker container pause.

update

docker container update -m 256m --cpus=1 containerID → `docker container update` permite alterar configurações de um ou mais containers com containerID em containers em execução. No exemplo alteramos a memória e cpu.

WARNING

`docker container update` e `docker update` não são suportados por containers Windows.

Opções disponíveis para docker container update:

- `--blkio-weight`

- `--cpu-period`
- `--cpu-quota`
- `--cpu-rt-period`
- `--cpu-rt-runtime`
- `--cpu-shares` , `-c`
- `--cpus`
- `--cpuset-cpus`
- `--cpuset-mems`
- `--kernel-memory`
- `--memory` , `-m`
- `--memory-reservation`
- `--memory-swap`
- `--pids-limit`
- `--restart`

image

L

ls

docker image ls → listar imagens existentes

Table 3. Significado das colunas

Coluna	Significado
REPOSITORY	nome da imagem
TAG	versão da imagem
IMAGE ID	identificação da imagem
CREATED	quando a imagem foi criada
SIZE	tamanho da imagem

images

docker images → Lista imagens de containers.

rmi

docker rmi REPOSITORY:Tag → Remove uma ou mais imagens, também pode usar a combinação `name(repository):tag` com o `docker rmi`.

start

docker start containerId → Inicia um (ou mais) container parado que tem o containerId como CONTAINER ID.

docker start -a -i containerId → inicia um container parado que tem o containerId como CONTAINER ID, e ao mesmo tempo uso um attach(-a) no container com intereção do terminal(-i). Inicio o container e ao mesmo e já entro dentro dele em execução com um terminal atrelado.

TIP

Caso queira derrubar o container

Se quiser sair se derrubar o container usar `Ctrl + P + Q`.

stop

docker stop containerId → Para um (ou mais) container ativos que tem o containerId como CONTAINER ID.

Volumes no Docker

Docker Volume

Algumas Explicações Sobre Volumes

Comandos

create

docker volume create volumeName → cria um volume volumeName no host local.

IMPORTANT

Sempre que um volume é criado ele cria um diretório com o mesmo nome dentro de `/var/lib/docker/volumes`

inspect

docker volume inspect volumeName → exibe informações detalhadas de um ou mais volumes.

docker volume inspect --format '{{ .Mountpoint }}' volumeName → exibe a localização do se volume com volumeName no host.

ls

docker volume ls → lista volumes dos container no host

rm

docker volume rm volumeName → remove um ou mais volume(s) com volumeName especificado

prune

docker volume prune → remove todos os volumes que não estão sendo usados.

Montando Volumes

Utilizando --mout type=bind

IMPORTANT

Não é muito aconselhevel utilizar o `--mout type=bind` e suas opções ao montar volumes. é uma forma mais antiga e não muito elegante.

docker container run -ti --mount type=bind,src=/pastaLocalNoHost,dst=/pastaDestinoContainer imagemContainer → Cria um container utilizando a imagem imagemContainer e compartilhar a pasta pastaLocalNoHost que está no host com o container, onde essa pasta corresponderá a pastaDestinoContainer.

Explicando o comando

--mount

anexa um ponto de montagem de arquivos ao container.

type=bind

indica que o source(src) será um diretório.

src=localPastaOrigem

indica o local no host que queremos que seja montado e compartilhado com o container

dst=localPastaDestino

indica o local que o diretório indicado em **src=localPastaOrigem** será montado dentro do container.

NOTE

No lugar de **src=/pastaLocalNoHost,dst=/pastaDestinoContainer** ao invés de referenciar uma pasta também podemos referenciar um arquivo no **src** e **dst**.

Utilizando --mout type=volume

IMPORTANT

O **--mout type=volume** é a maneira mais elegante atual de se criar volumes.

docker container run -ti --mount type=volume,src=volumeName,dst=/pastaDestinoContainer imagemContainer → Cria um container utilizando a imagem **imagemContainer** e compartilhar o volume criado com nome **volumeName** que está no host com o container, onde essa o conteúdo desse volume corresponderá a **pastaDestinoContainer**.

Explicando o comando

--mount

anexa um ponto de montagem de arquivos ao container.

type=volume

indica que o tipo é volume e que o source(src) será um volume.

src=volumeName

indica qual o volume, no caso **volumeName**, que queremos que seja montado e compartilhado com o container

dst=localPastaDestino

indica o local que o diretório indicado em **src=localPastaOrigem** será montado dentro do container.

Links para explicação sobre volumes no docker

<http://stack.desenvolvedor.expert/appendix/docker/armazenamento.html>