

# Principais comandos

## docker run

Cria um novo contêiner, o inicia e executa o comando especificado (quando nada é especificado, executa o comando padrão).

```
docker run hello-world
```

Você verá uma saída como:

```
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
(...)
```

No comando anterior o terminal é direcionado para dentro do contêiner. Para iniciá-lo no modo *avulso* (`detached`) utilizaremos o parâmetro `-d`:

```
docker run -d hello-world
```

O terminal exibirá o `ID` de criação do contêiner, algo como:

```
bdcaba00ed90289490331613191d1ec2010dbc94330b532b4da8f7714e417d8e
```

Quando um nome (parâmetro `--name="nome"`) não é fornecido, o Docker criará o container com um nome aleatório, geralmente composto de 2 palavras (nome1\_nome2).

**Exemplo:** `inspiring_euclid`

**Dica:** Todos os comandos de manipulação de contêineres docker poder ser utilizados se referenciando o `nome do contêiner` ou o seu respectivo `ID`.

## docker ps

Lista os contêineres disponíveis na máquina. Por padrão, somente aqueles que estão atualmente executando serão exibidos. Para listar todos, utilize o parâmetro `-a`:

```
docker ps -a
```

No nosso exemplo, você verá algo como:

CONTAINER_ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
14a72c95d77b	hello-world	"/hello"	1 minute ago	Exited (0) 1 minutes ago		unruffled_bartik

## docker start

Inicia um contêiner que já tenha sido criado no host, seu acesso poderá ser realizado por `nome` ou por `ID`. Utilizando o exemplo anterior, vamos copiar o `ID` informado na coluna `CONTAINER_ID` para iniciá-lo novamente.

```
docker start 14a72c95d77b
```

Você verá como saída o `ID` do container indicando que ele foi iniciado.

```
14a72c95d77b
```

**Dica:** Um contêiner pode ser finalizado por ter terminado de executar a rotina para qual ele foi designado (Exemplo: processamento de um arquivo texto) ou por ter encontrado um erro no serviço (Exemplo: configuração incorreta, como um erro de sintaxe, no servidor web).

Quando se cria um contêiner, pode-se definir qual o política padrão de reinicialização quando ele é encerrado: `no`, `always`, `on-failure`, `unless-stopped`.

## docker stop

Encerra a execução de um contêiner. Mesma lógica do comando anterior.

```
docker stop 14a72c95d77b
```

**Dica:** A execução do container é paralisada, no entanto o container não é removido do host.

## docker restart

Reinicia a execução de um contêiner( `stop` e `start`). Mesma lógica dos comandos anteriores.

## docker rm

Remove um contêiner do host. Você pode utilizar `docker ps -a` para listar a relação de contêineres disponíveis, e removê-lo por `nome` ou `id`.

```
docker rm 14a72c95d77b
```

O comando `docker rm` somente remove containers que não estão em execução. Para remover um container em execução, use `docker rm -f`

Para conferir a remoção, vamos checar novamente os containers criados no host:

```
docker ps -a
```

## docker images

Conferir as imagens docker atualmente disponíveis no host:

```
docker images
```

Quando uma imagem docker não está disponível localmente, o docker tentará fazer o download da imagem docker em um repositório docker (Ex: [hub.docker.com](https://hub.docker.com)). Caso o repositório seja privado, será necessário efetuar o login neste repositório para ter acesso às imagens.

## docker pull

Faz o download da imagem docker do repositório para o host:

```
docker pull alpine
```

Você verá uma saída como:

```
Using default tag: latest
latest: Pulling from library/alpine
ec99f8b99825: Pull complete
Digest: sha256:b89d9c93e9ed3597455c90a0b88a8bbb5cb7188438f70953fede212a0c4394e0
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Quando você não especifica uma `tag`, o docker tomará como valor padrão a tag `latest`. O padrão de nomenclatura da imagem é `nome-da-imagem` : `nome-da-tag`.

Fazer o download de uma imagem docker em uma tag específica (tag: `3.20.1`):

```
docker pull alpine:3.20.1
```

Você verá uma saída como:

```
3.20.1: Pulling from library/alpine
Digest: sha256:b89d9c93e9ed3597455c90a0b88a8bbb5cb7188438f70953fed212a0c4394e0
Status: Downloaded newer image for alpine:3.20.1
docker.io/library/alpine:3.20.1
```

Depois de baixarmos as imagens, vamos conferir novamente as imagens disponíveis:

```
docker images
```

Saída:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	3.20.1	a606584aa9aa	12 days ago	7.8MB
alpine	latest	a606584aa9aa	12 days ago	7.8MB

## docker rmi

Remove a imagem do host. No entanto, só é possível remover as imagens que não está sendo utilizadas por nenhum contêiner no host.

```
docker rmi alpine
```

Para ver a lista de todas as imagens baixadas para o host utilize `docker images`

## docker exec

Executa um comando dentro de um contêiner que está em execução.

- Primeiro executaremos um contêiner criado a partir de uma imagem do Ubuntu, em modo `detached`:

```
docker run --name ubuntu_bash -d -it ubuntu
```

- Vamos conferir se o contêiner está em execução sem problemas:

```
docker ps -a
```

Saída:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
759d52725ffe	ubuntu	"/bin/bash"	7 seconds ago	Up 6 seconds		ubuntu_bash

- Agora executaremos o comando `ls -la` dentro do contêiner:

```
docker exec ubuntu_bash ls -la
```

Saída:

```
total 56
drwxr-xr-x  1 root root 4096 Jul  3 12:50 .
drwxr-xr-x  1 root root 4096 Jul  3 12:50 ..
-rwxr-xr-x  1 root root    0 Jul  3 12:50 .dockerenv
lrwxrwxrwx  1 root root    7 Apr 22 13:08 bin -> usr/bin
drwxr-xr-x  2 root root 4096 Apr 22 13:08 boot
drwxr-xr-x  5 root root 360 Jul  3 12:50 dev
drwxr-xr-x  1 root root 4096 Jul  3 12:50 etc
drwxr-xr-x  3 root root 4096 Jun  5 02:06 home
lrwxrwxrwx  1 root root    7 Apr 22 13:08 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Apr 22 13:08 lib64 -> usr/lib64
drwxr-xr-x  2 root root 4096 Jun  5 02:02 media
drwxr-xr-x  2 root root 4096 Jun  5 02:02 mnt
drwxr-xr-x  2 root root 4096 Jun  5 02:02 opt
dr-xr-xr-x 399 root root    0 Jul  3 12:50 proc
drwx----- 2 root root 4096 Jun  5 02:05 root
drwxr-xr-x  4 root root 4096 Jun  5 02:06 run
lrwxrwxrwx  1 root root    8 Apr 22 13:08 sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 Jun  5 02:02 srv
dr-xr-xr-x 13 root root    0 Jul  3 12:50 sys
drwxrwxrwt  2 root root 4096 Jun  5 02:05 tmp
drwxr-xr-x 12 root root 4096 Jun  5 02:02 usr
drwxr-xr-x 11 root root 4096 Jun  5 02:05 var
```

- Para logar dentro do contêiner e executar comandos diretamente de lá (`exit` para sair):

```
docker exec -it ubuntu_bash bash
```

Saída:

```
root@759d52725ffe:/#
```

Note que você está dentro do terminal do contêiner, com o usuário `root`.

Saia do contêiner:

```
root@759d52725ffe:/# exit
exit
```

## docker stats

Exibe o consumo de recursos dos containers em execução no host:

```
docker stats
```

Saída (exemplo):

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
e95bd53f6904	apache-ecampus	0.00%	98.92MiB / 512MiB	19.32%	22.6MB / 48.3MB	1.86MB / 12.3kB	8
77aa8c8f5d9b	pgadmin4	0.03%	202.9MiB / 31.18GiB	0.64%	53MB / 14MB	5.07MB / 22.3MB	20
c3f5ff58225d	lumen-pm	0.01%	52.88MiB / 31.18GiB	0.17%	318kB / 0B	5.41MB / 8.19kB	8
7659f7c19173	void-mail	6.08%	18.43MiB / 512MiB	3.60%	8.5MB / 9MB	6.45MB / 0B	2
4edcb809e7f8	apache-void	0.01%	11.64MiB / 512MiB	2.27%	13.1MB / 349kB	188kB / 8.19kB	7
39294708453c	apache-diploma	0.00%	2.672MiB / 512MiB	0.52%	348MB / 41.2MB	1.4MB / 9.72MB	2
c0187b82297c	apache-relatorio	0.00%	4.535MiB / 512MiB	0.89%	308MB / 7.45MB	3.44MB / 0B	2
ca7dd8afcbe3	laravel-mangaba	0.01%	124.9MiB / 31.18GiB	0.39%	318kB / 0B	10.9MB / 13.3MB	8
6e79d13253aa	laravel-assinador	0.00%	152.2MiB / 31.18GiB	0.48%	1.95MB / 1.71MB	3.42MB / 9.5MB	8
f782b4d255c6	rebrow-assinador	0.01%	22.86MiB / 31.18GiB	0.07%	318kB / 0B	4.89MB / 324kB	1
38aa0954d19c	web-ckan	0.01%	183.2MiB / 31.18GiB	0.57%	22.8MB / 22.6MB	36.9kB / 4.31MB	11
80e596827889	apache-moodle3	0.01%	10.5MiB / 512MiB	2.05%	13.1MB / 352kB	12.3kB / 4.1kB	6
670a68bc71ff	postgres-ecampus	0.26%	168.2MiB / 31.18GiB	0.53%	50MB / 654MB	23.6MB / 505MB	7

e9265ae24b14	conector-ufvjm	0.25%	187.9MiB / 31.18GiB	0.59%	9.31MB / 8.72MB	16.4kB / 8.36MB	43
7aff38a1598f	phpmyadmin	0.01%	41.53MiB / 31.18GiB	0.13%	318kB / 0B	422kB / 1.98MB	7
fb98fc81448a	postgres-mangaba	0.01%	18.11MiB / 31.18GiB	0.06%	318kB / 0B	1.45MB / 14.4MB	7
27739272e682	apache-ldapadmin	0.01%	13.84MiB / 31.18GiB	0.04%	670kB / 289kB	8.19kB /	12.3kB 8
30bf52a8d11a	solr-ckan	0.11%	335.9MiB / 31.18GiB	1.05%	4.08MB / 3.75MB	45.1kB / 68.8MB	48
8481f13f87b7	postgres-pressiga	0.67%	43.35MiB / 31.18GiB	0.14%	4.14MB / 3.74MB	4.86MB /	71.7MB 7
6d1e7ba2556d	maria-pm	0.06%	97.93MiB / 31.18GiB	0.31%	319kB / 0B	11.1MB / 2.17MB	31
35aac4c5e8ca	redis-ckan	0.20%	3.551MiB / 31.18GiB	0.01%	340kB / 20.8kB	520kB / 0B	5
ade1ecfeb5e8	postgres-conector	0.01%	20.16MiB / 31.18GiB	0.06%	9.03MB / 8.98MB	4.83MB / 23MB	7
6e0fe48184b3	datapusher-ckan	0.02%	25.97MiB / 31.18GiB	0.08%	319kB / 0B	16.4kB / 111kB	1
15e396ca1835	postgres-ckan	0.00%	27.46MiB / 31.18GiB	0.09%	17.7MB / 17.3MB	8.09MB / 43.1MB	12
2fafc1c89a6c	redis-assinador	0.18%	7.848MiB / 31.18GiB	0.02%	319kB / 0B	4.1kB / 0B	5
5daf93ede22b	postgres-moodle	0.00%	21.84MiB / 31.18GiB	0.07%	318kB / 0B	5.52MB / 20.1MB	7
dc3216f22764	mailhog	0.00%	14.86MiB / 31.18GiB	0.05%	940kB / 574kB	2.86MB / 1.89MB	18
7237c9d8ab05	cache-ecampus	0.03%	8.301MiB / 31.18GiB	0.03%	2.37MB / 925kB	16.4kB / 0B	10
3819a952c8e6	ldap-ufvjm	0.03%	724.1MiB / 31.18GiB	2.27%	2.59MB / 1.19MB	586kB / 445MB	4
e05b811b58bc	postgres-assinador	0.01%	28.07MiB / 31.18GiB	0.09%	2.03MB / 1.63MB	12.3MB /	43.5MB 7

Para liberar a saída do terminal novamente use o atalho `Ctrl + C`.

## docker logs

Exibe no terminal os logs do contêiner (parâmetro `-f` pode ser utilizado para travar a tela exibindo os logs, `Ctrl + C` para sair):

```
docker logs ubuntu_bash
```

Remover o container criado para realizar os testes:

```
docker rm -f ubuntu_bash
```

Saída:

```
ubuntu_bash
```

## Referências

[docker command CLI](#)

[List of Docker Commands with Examples](#)

---

Revision #18

Created 2 July 2024 19:45:36 by Éverton de Oliveira Paiva

Updated 4 July 2024 17:34:47 by Éverton de Oliveira Paiva