

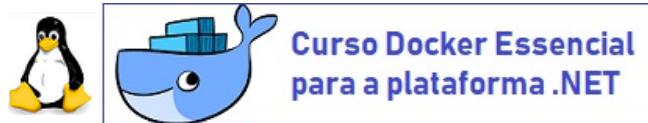


Neste artigo vou apresentar os conceitos básicos relativos ao Docker sob o ponto de vista de um desenvolvedor .NET.



Hoje vamos continuar criando **contêineres** a partir de **imagens** existentes no **Docker Hub** usando o modo interativo do Docker.

Lembrando que estou no ambiente **Linux** usando o **Ubuntu 18.04.1**.(artigo anterior)



Criando contêineres a partir de imagens existentes

Para criar um contêiner usamos o comando: **docker container run <nome_da_imagem> <comando>**

A novidade aqui é que podemos criar um contêiner a partir de uma imagem e executar um comando.

Vamos criar um contêiner digitando o comando abaixo em um terminal:

docker container run debian bash --version

Onde:

docker container run -> comando para criar um contêiner
debian -> nome da imagem (será obtida do docker hub)
bash --version -> comando a ser executado

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run debian bash --version
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
bc9ab73e5b14: Pull complete
Digest: sha256:802706fa62e75c96fff96ada0e8ca11f570895ae2e9ba4a9d409981750ca544c
Status: Downloaded newer image for debian:latest
GNU bash, version 4.4.12(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
root@linux:/home/macoratti#
```

Na execução do comando temos que :

- 1 - A imagem não foi localizada localmente e foi baixada do repositório;
- 2 - O comando **bash --version** foi executado;
- 3 - O execução do contêiner foi encerrada e retornamos ao prompt do linux;

Para ver todos processos dos contêineres digite: **docker container ps -a**

```

root@linux: /home/macoratti

Arquivo Editar Ver Pesquisar Terminal Ajuda

root@linux:/home/macoratti# clear

root@linux:/home/macoratti# docker container ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
9f0a290ef8e2       debian             "bash --version"   6 minutes ago       Exited (0) 6 minutes ago              lucid_kalam
d5495113f4b4       hello-world        "/hello"           24 hours ago        Exited (0) 24 hours ago              frosty_pare
root@linux:/home/macoratti#

```

Note que temos dois contêineres criados e com as seguintes informações:

CONTAINER_ID	O identificador do contêiner
IMAGE	o nome da imagem usada
COMMAND	o comando executado no momento
CREATED	a data de criação do contêiner
STATUS	o status do contêiner (tempo de execução)
PORTS	a porta do contêiner aberta
NAMES	o nome do contêiner atribuído pelo docker

Para visualizar as imagens digite o comando : **docker images**

```

root@linux: /home/macoratti

Arquivo Editar Ver Pesquisar Terminal Ajuda

root@linux:/home/macoratti# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
debian              latest             be2868bebaba       2 weeks ago        101MB
hello-world         latest             4ab4c602aa5e       8 weeks ago        1.84kB
root@linux:/home/macoratti#
root@linux:/home/macoratti#

```

Temos duas imagens com as seguintes informações:

REPOSITORY	o nome da imagem no repositório
TAG	a versão da imagem
IMAGE ID	o identificador da imagem
CREATED	a data de criação da imagem
SIZE	o tamanho da imagem

Se você executar novamente o comando, a execução vai ser mais rápida, pois a imagem **debian** agora existe localmente na sua máquina.

Agora vamos executar o comando: **docker container run --rm debian bash --version**

Neste comando temos o parâmetro **--rm** que vai remover o contêiner após sua execução. Assim após a execução o processo do contêiner na memória será removido.

Para mostrar isso vamos criar um contêiner e após sua execução removê-lo. Digite:

docker container run --rm microsoft/dotnet-samples

```

root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run --rm microsoft/dotnet-samples
Unable to find image 'microsoft/dotnet-samples:latest' locally
latest: Pulling from microsoft/dotnet-samples
802b00ed6f79: Pull complete
83bc4824d9c8: Pull complete
7d492e2a3505: Pull complete
ba4769a99356: Pull complete
fe3e6f11e347: Pull complete
0e47213f735e: Pull complete
Digest: sha256:61f0d271b94b1747225d7126b9e9125dc1d69ca61cf0ef1ca50d0cd889153281
Status: Downloaded newer image for microsoft/dotnet-samples:latest
Hello from .NET Core!

```

Nota: Aqui estou usando uma imagem do .NET Core.

Após a execução do comando se procurarmos pelo contêiner digitando: `docker ps -a`

Não veremos o processo usado para esse contêiner pois ele foi removido.

Podemos criar um contêiner e ter acesso iterativo ao contêiner. Vamos ver isso digitando o comando:

`docker container run -it debian bash`

Neste comando estamos usando os parâmetros `-it` onde `i` significa o modo interativo e o `t` para que possamos ter acesso ao terminal `bash`:

- `-i` é o parâmetro para interagir(vai manter o STDIN aberto).
- `-t` é para alocar um TTY(Talk to you) que é um terminal.

```

root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run -it debian bash
root@ff6ec7c41cde:/# ls -a
. . . .dockerenv bin boot dev etc home lib lib64 media mnt opt proc root run/sbin srv sys tmp usr var
root@ff6ec7c41cde:/# touch docker.txt
root@ff6ec7c41cde:/# ls docker.txt
docker.txt
root@ff6ec7c41cde:/# exit
exit
root@linux: /home/macoratti#

```

Na execução entramos no terminal do contêiner e executamos os comandos:

- 1- `ls -a` -> exibe os arquivos
- 2- `touch docker.txt` -> cria o arquivo docker.txt
- 3- `ls docker.txt` -> exibe o arquivo
- 4- `exit` -> sai do contêiner

Nota: Para sair de dentro do container, digite `exit`. Isso vai matar o container e ele não vai existir mais. O mesmo vai acontecer se você fizer um `CTRL+C`.

Assim criamos um arquivo texto no contêiner criado.

Se você executar novamente o último comando e procurar pelo arquivo `docker.txt` não vai encontrar...

Por quê ???

Porque um comando `docker container run` sempre cria um novo contêiner.

Vamos agora criar um contêiner e dar um nome a ele, pois como você já percebeu, o Docker dá nome aleatório aos contêineres, mas esses nomes não são fáceis de lembrar.

Digite no terminal: `docker container run --name macdeb -it debian bash`

Aqui estamos usando o parâmetro `--name` e dando o nome de `macdeb` para o contêiner:

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run --name macdeb -it debian bash
root@b0a6555a1089:/# exit
exit
root@linux:/home/macoratti#
```

Se tentarmos executar novamente o mesmo comando teremos um erro pois já existe um contêiner com o nome `macdeb`:

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run --name macdeb -it debian bash
docker: Error response from daemon: Conflict. The container name "/macdeb" is already
in use by container "b0a6555a1089b5a151032a823ff5236a6bc05835de20920a7d2bcbb81502fa43".
You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
root@linux:/home/macoratti#
```

Visualizando todos os contêineres: `docker container ps -a`

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b0a6555a1089	debian	"bash"	6 minutes ago	Exited (0) 5 minutes ago		macdeb
ff0ec7c41cde	debian	"bash"	26 minutes ago	Exited (0) 11 minutes ago		festive_to
rvalds						
9f0a290ef8e2	debian	"bash --version"	About an hour ago	Exited (0) About an hour ago		lucid_kala
d5495113f4b4	hello-world	"/hello"	25 hours ago	Exited (0) 25 hours ago		frosty_par

```
root@linux:/home/macoratti#
```

Vemos o nosso contêiner criado com o nome `macdeb` atribuído.

Sabendo o nome do contêiner podemos reusar o contêiner.

Vamos primeiro exibir os contêineres digitando: `docker container ls -a`

Este comando também lista todos os contêineres:

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b0a6555a1089	debian	"bash"	6 minutes ago	Exited (0) 5 minutes ago		macdeb
ff0ec7c41cde	debian	"bash"	26 minutes ago	Exited (0) 11 minutes ago		festive_to
rvalds						
9f0a290ef8e2	debian	"bash --version"	About an hour ago	Exited (0) About an hour ago		lucid_kala
d5495113f4b4	hello-world	"/hello"	25 hours ago	Exited (0) 25 hours ago		frosty_par

```
root@linux:/home/macoratti#
```

Para reutilizar o contêiner `macdeb` digite:

docker container start -ai macdeb

Neste comando estamos usando o método **start** para iniciar um contêiner e os parâmetros **-ai** onde **a** vai anexar(*atachar*) o terminal e **i** indica o modo interativo:

```

root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container start -ai macdeb 1
root@b0a6555a1089:/# touch docker.txt a
root@b0a6555a1089:/# ls
bin boot dev docker.txt etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@b0a6555a1089:/# exit b
exit
root@linux:/home/macoratti#
root@linux:/home/macoratti# docker container start -ai macdeb 2
root@b0a6555a1089:/# ls a
bin boot dev docker.txt etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@b0a6555a1089:/#

```

Na execução fizemos o seguinte:

1- Reutilizamos o contêiner **macdeb**

- a - no contêiner criamos o arquivo texto **docker.txt** e listamos os arquivos
- b- saímos do contêiner(**exit**)

2- Reutilizamos novamente o contêiner **macdeb**

- a- Digitamos **ls** e vemos que o arquivo **docker.txt** criado no contêiner esta presente.

Com isso podemos reutilizar contêineres já existentes. Assim é importante dar um nome ao contêiner para poder reutilizá-lo.

Para concluir vou deixar uma lista dos principais comandos relacionados aos contêineres:

Comando	Descrição
docker container run	Cria e inicia um contêiner
docker container ps	Exibe os processos dos contêineres em execução. Para lista todos inclua -a.
docker container rm	Remove um contêiner. Apagar todos: docker container rm \$(docker ps -a -q)
docker container create	Cria um novo contêiner com base em uma imagem
docker container stop	Para a execução de um contêiner
docker container logs	Exibe a saída gerada pelo contêiner
docker container exec	Executa um comando em um contêiner ou inicia uma sessão interativa
docker container rename	Renomeia um contêiner existente
docker container start	Inicia a execução de um contêiner existente

Nota: Você pode até omitir a palavra **container** dos comandos mas a sintaxe atual recomenda usá-la.

Os parâmetros mais utilizados na execução de um container são:

Parâmetro	Descrição
-d	Execução do container em background(segundo plano)
-i	Modo interativo. Mantém o STDIN aberto mesmo sem console anexado
-t	Aloca uma pseudo TTY
--rm	Automaticamente remove o container após finalização. (Não funciona com -d)
--name	Da um nome ao container
-v	Realiza o mapeamento de um volume
-p	Faz o mapeamento de porta
-m	Limta o uso de memória RAM
-c	Realiza o balanceamento do uso de CPU

Na [próxima aula](#) veremos como mapear **portas, diretórios e volumes** em contêineres.