**Criando e administrando *containers* Docker**

Como todos sabemos, o Docker utiliza a linha de comando para que você possa interagir com ele -- basicamente você utiliza o comando "docker".

Bom, agora que já iniciamos o Docker, vamos rodar nosso primeiro *container*.

Como é de costume quando alguém está aprendendo uma nova linguagem de programação, é bem comum fazer como o primeiro código um ***hello world*!**

Apesar de o Docker não ser uma linguagem de programação, vamos utilizar esse costume com o nosso primeiro exemplo de um *container* em execução.

O Docker possui uma imagem personalizada de *hello-world* e serve para que você possa testar a sua instalação e validar se tudo funciona conforme o esperado. :D

Para que possamos executar um *container*, utilizamos o parâmetro "run" do subcomando "container" do comando "docker". Simples, não? :D

root@linuxtips:~# docker container run hello-world

Unable to find image 'hello-world:latest' locally

latest: Pulling from library/hello-world

03f4658f8b78: Pull complete a3ed95caeb02: Pull complete

Digest: sha256:8be990ef2aeb16dbcb9271ddfe2610fa6658d13f6dfb8bc72074cc1ca36966a7

Status: Downloaded newer image for hello-world:latest

Hello from Docker.

This message shows that your installation appears to be working correctly.

1. The Docker client contacted the Docker daemon.

2. The Docker daemon pulled the "hello-world" image from the Docker Hub.

(amd64)

3. The Docker daemon created a new container from that image which runs the

executable that produces the output you are currently reading.

4. The Docker daemon streamed that output to the Docker client, which sent it

to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker Hub account:

https://hub.docker.com

For more examples and ideas, visit:

https://docs.docker.com/userguide/

root@linuxtips:~#

No exemplo anterior, estamos executando um *container* utilizando a imagem personalizada do *hello-world*.

Apesar de ser uma tarefa simples, quando você executou o comando "docker container run hello-world" foram necessárias quatro etapas para sua conclusão, vamos ver quais:

1. O comando "docker" se comunica com o *daemon* do Docker informando a ação desejada.
2. O *daemon* do Docker verifica se a imagem "hello-world" existe em seu *host*; caso ainda não, o Docker faz o *download* da imagem diretamente do Docker Hub.
3. O *daemon* do Docker cria um novo *container* utilizando a imagem que você acabou de baixar.
4. O *daemon* do Docker envia a saída para o comando "docker", que imprime a mensagem em seu terminal.

Viu? É simples como voar! :)

Muito bem, agora que nós já temos uma imagem em nosso *host*, como eu faço para visualizá-la?

Muito simples, basta digitar o seguinte comando:

root@linuxtips:~# docker image ls

REPOSITORY TAG IMAGE ID CREATED SIZE

hello-world latest 690ed74de00f 5 months 960 B

root@linuxtips:~#

Como você pode notar no código, a saída traz cinco colunas:

* **REPOSITORY** -- O nome da imagem.
* **TAG** -- A versão da imagem.
* **IMAGE ID** -- Identificação da imagem.
* **CREATED** -- Quando ela foi criada.
* **SIZE** -- Tamanho da imagem.

Quando executamos o comando "docker container run hello-world", ele criou o *container*, imprimiu a mensagem na tela e depois o *container* foi finalizado automaticamente, ou seja, ele executou sua tarefa, que era exibir a mensagem, e depois foi finalizado.

Para ter certeza de que ele realmente foi finalizado, digite:

root@linuxtips:~# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORT NAMES

root@linuxtips:~#

Com o "docker container ls", você consegue visualizar todos os *containers* em execução e ainda obter os detalhes sobre eles. A saída do "docker container ls" é dividida em sete colunas; vamos conhecer o que elas nos dizem:

* **CONTAINER ID** -- Identificação única do *container.*
* **IMAGE** -- A imagem que foi utilizada para a execução do *container.*
* **COMMAND** -- O comando em execução.
* **CREATED** -- Quando ele foi criado.
* **STATUS** -- O seu status atual.
* **PORTS** -- A porta do *container* e do *host* que esse *container* utiliza.
* **NAMES** -- O nome do *container.*

Uma opção interessante do "docker container ls" é o parâmetro "-a".

root@linuxtips:~# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

6e45cf509282 hello-world "/hello" 4 seconds Exited(0) tracted\_ardinghelli

root@linuxtips:~#

Com a opção "-a" você consegue visualizar não somente os *containers* em execução, como também *containers* que estão parados ou que foram finalizados.

**Legal, quero mais!**

Agora que vimos como criar um simples *container*, bem como visualizar as imagens e *containers* que estão em nosso *host*, vamos criar um novo, porém conhecendo três parâmetros que irão trazer maior flexibilidade no uso e na administração de nossos *containers*. Estou falando dos parâmetros "-t", "-i" e "-d".

* **-t** -- Disponibiliza um TTY (console) para o nosso *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 nos fornecem.

Com isso temos dois modos de execução de nossos *containers*: modo interativo ou *daemonizando* o *container*.

**Modo interativo**

Na maior parte das vezes você vai subir um *container* a partir de uma imagem que já está pronta, toda ajustadinha. Porém, há alguns casos em que você precisa interagir com o seu *container* -- isso pode acontecer, por exemplo, na hora de montar a sua imagem personalizada.

Nesse caso, usar o modo interativo é a melhor opção. Para isso, basta passar os parâmetros "-ti" ao comando "docker container run".

***Daemonizando* o *container***

Utilizando o parâmetro "-d" do comando "docker container run", é possível *daemonizar* o *container*, fazendo com que o *container* seja executado como um processo *daemon*.

Isso é ideal quando nós já possuímos um *container* que não iremos acessar (via *shell*) para realizar ajustes. Imagine uma imagem já com a sua aplicação e tudo que precisa configurado; você irá subir o *container* e somente irá consumir o serviço entregue por sua aplicação. Se for uma aplicação *web*, basta acessar no *browser* passando o IP e a porta onde o serviço é disponibilizado no *container*. Sensacional, não?

Ou seja, se você quer subir um *container* para ser utilizado como uma máquina Linux convencional com *shell* e que necessita de alguma configuração ou ajuste, utilize o modo interativo, ou seja, os parâmetros "-ti".

Agora, se você já tem o *container* configurado, com sua aplicação e todas as dependências sanadas, não tem a necessidade de usar o modo interativo -- nesse caso utilizamos o parâmetro "-d", ou seja, o *container daemonizado*. Vamos acessar somente os serviços que ele provê, simples assim. :D

**Entendi, agora vamos praticar um pouco?**

Perfeito. Vamos iniciar um novo *container* utilizando dois desses novos parâmetros que aprendemos.

Para o nosso exemplo, vamos subir um *container* do Centos 7:

root@linuxtips:~# docker container run -ti centos:7

Unable to find image 'centos:7' locally

7: Pulling from library/centos

a3ed95caeb02: Pull complete 196355c4b639: Pull complete

Digest: sha256:3cdc0670fe9130ab3741b126cfac6d7720492dd2c1c8ae033dcd77d32855bab2

Status: Downloaded newer image for centos:7

[root@3c975fb7fbb5 /]#

Como a imagem não existia em nosso *host*, ele começou a baixar do Docker Hub, porém, caso a imagem já estivesse em nosso *host*, ele a utilizaria, não sendo necessário o *download*.

Perceba que mudou o seu *prompt* (variável $PS1), pois agora você já está dentro do *container*. Para provar que estamos dentro do nosso *container* Centos, execute o seguinte comando:

[root@3c975fb7fbb5 /]# cat /etc/redhat-release

CentOS Linux release 7.2.1511 (Core)

[root@3c975fb7fbb5 /]#

O arquivo "/etc/redhat-release" indica qual a versão do Centos que estamos utilizando, ou seja, estamos realmente em nosso *container* Centos 7. :D

**Tá, agora quero sair...**

Idealmente, no *container* vai haver apenas um processo rodando. No nosso caso, como estamos interagindo (opção "-ti"), é o processo do *bash*; logo, você não pode utilizar o comando "exit" para sair do console, pois dessa forma esse único processo para de rodar e seu *container* morre. Caso queira sair do *container* e mantê-lo em execução, é necessário sair com o seguinte atalho do teclado:

**mantenha o botão Ctrl pressionado + p + q**

Assim, você sairá do *container* e ele continuará em execução. Para confirmar se o *container* continua em execução, faça:

root@linuxtips:~# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3c975fb7fbb5 centos:7 "/bin/bash" 2 minutes Up 2 minutes angry\_wescoff

root@linuxtips:~#

**Posso voltar ao *container*?**

Deixamos o nosso *container* em execução e agora queremos acessá-lo novamente. Como podemos fazer?

Simples! Basta digitar o seguinte comando:

root@linuxtips:~# docker container attach <CONTAINER ID>

O parâmetro "attach" do comando "docker container" possibilita nos conectarmos a um *container* em execução. Para isso, basta passar como parâmetro o "CONTAINER ID", que você consegue através da saída do "docker ps", conforme mostramos no exemplo anterior.

**Continuando com a brincadeira...**

Existe a possibilidade de criar um *container,* porém não o executar imediatamente. Quando fazemos o uso do parâmetro "create" do comando "docker container", ele apenas cria o *container*, não o inicializando, conforme notamos no exemplo a seguir:

root@linuxtips:~# docker container create -ti ubuntu

Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

5a132a7e7af1: Pull complete

fd2731e4c50c: Pull complete

28a2f68d1120: Pull complete

a3ed95caeb02: Pull complete

Digest:sha256:4e85ebe01d056b43955250bbac22bdb8734271122e3c78d21e55ee235fc6802d

Status: Downloaded newer image for ubuntu:latest3e63e65db85a6e36950959dc6bdc00279e2208a335580c478e01723819de9467

root@linuxtips:~#

Perceba que quando você digita "docker container ls" ele não traz o *container* recém-criado, afinal a saída do "docker container ls" somente traz os *containers* em execução. Para visualizar o *container* recém-criado foi necessário utilizar o parâmetro "-a".

root@linuxtips:~# docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3e63e65db85a ubuntu "/bin/bash" 18 seconds ago Created elo\_visves

root@linuxtips:~#

Para que o nosso *container* recém-criado seja executado, basta utilizar o "docker container start [CONTAINER ID]", conforme segue:

root@linuxtips:~# docker container start [CONTAINER ID]

root@linuxtips:~# docker container attach [CONTAINER ID]

root@b422f04df14c:/#

Verificando se estamos realmente utilizando o *container* do Ubuntu:

root@b422f04df14c:/# cat /etc/issue

Ubuntu 18.04 LTS \n \l

root@b422f04df14c:/#

Lembrando que para sair do *container* e mantê-lo em execução é necessário utilizar o atalho: **Ctrl + p + q**.

**Subindo e matando *containers*...**

Caso eu queira parar um *container* em execução, basta utilizar o parâmetro "stop" seguido do "CONTAINER ID":

# docker container stop [CONTAINER ID]

Verificando se o *container* continua em execução:

# docker container ls

Lembrando que para visualizar os *containers* que não estão em execução é necessário utilizar o parâmetro "-a".

Para colocar novamente em execução um *container* que está parado, é necessário utilizar o parâmetro "start" do comando "docker container" seguido do "CONTAINER ID":

# docker container start [CONTAINER ID]

Da mesma forma como podemos utilizar o *stop/start* para desligar/iniciar um *container*, podemos também fazer o uso do "restart", como notamos a seguir:

# docker container restart [CONTAINER ID]

Para pausar um *container*, execute:

# docker container pause [CONTAINER ID]

E verifique o status do *container*:

root@linuxtips:~# docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

b34f4987bdce ubuntu "/bin/bash" 12 seconds ago Up 11 seconds (Paused) drunk\_turi

root@linuxtips:~#

Para "despausar" o *container*:

# docker container unpause [CONTAINER ID]

**Visualizando o consumo de recursos pelo *container*...**

Caso você queira visualizar informações referentes ao consumo de recursos pelo *container*, também é bastante simples: basta utilizar o parâmetro "stats" para verificar o consumo de CPU, memória e rede pelo *container* em tempo real.

# docker container stats [CONTAINER ID]

CONTAINER CPU% MEM USAGE/LIMIT MEM % NET I/O BLOCK I/O PIDS

b34f4987bdce 0.00% 503.8kB/2.094GB 0.02% 648B/648B 0B/0B 2

Para sair, pressione **Ctrl + C.**

Para visualizar todos os *containers* de uma só vez, basta não especificar o [CONTAINER ID],conforme segue:

# docker container stats

Agora, se você quer visualizar quais processos estão em execução em determinado *container*, utilize o parâmetro "top". Com ele você consegue informações sobre os processos em execução, como, por exemplo, UID e o PID do processo.

# docker container top [CONTAINER ID]

UID PID PPID C STIME TTY TIME COMMAND

root 10656 4303 0 20:24 pts/3 00:00:00 /bin/bash

Para verificar os *logs* de um determinado *container*, utilize o parâmetro "logs", simples assim. :D

# docker container logs [CONTAINER ID]

Lembre-se: ele exibe o STDOUT, a saída padrão. Ou seja, normalmente você irá visualizar o histórico de mensagens que aparecerem em primeiro plano durante a execução do *container*.

Para exibir os *logs* de forma dinâmica, ou seja, conforme aparecem novas mensagens ele atualiza a saída no terminal utilizamos a opção "-f"

# docker container logs -f [CONTAINER ID]

. Com isso seu terminal ficará travado, apenas escutando o *log*, e qualquer nova entrada ele exibirá na tela. Saída parecida com o "tail -f" no Linux. Lembre-se, utilize o ctrl+c para cancelar a exibição dos logs.

**Cansei de brincar de *container*, quero removê-lo!**

Bem, remover um *container* é mais simples ainda do que sua criação. Quando removemos um *container*, a imagem que foi utilizada para a sua criação permanece no *host*; somente o *container* é apagado.

root@linuxtips:~# docker container rm b34f4987bdce

Failed to remove container (b34f4987bdce): Error response from daemon:

Conflict, You cannot remove a running container. Stop the container

before attempting removal or use -f

root@linuxtips:~#

Perceba que, quando você tentou remover o *container*, ele retornou um erro dizendo que falhou em remover, pois o *container* estava em execução. Ele inclusive recomenda que você pare o *container* antes de removê-lo ou então utilize a opção "-f", forçando assim sua remoção.

root@linuxtips:~# docker container rm -f b34f4987bdce

b34f4987bdce

root@linuxtips:~#

Para confirmar a remoção do *container*, utilize o comando "docker container ls -a".