06

Praticando com o docker run

Transcrição

Agora que já conhecemos mais sobre *containers*, imagens e a diferença entre eles, já podemos fazer um *container* mais interessante, um pouco mais complexo. Então, vamos criar um *container* que segurará um site estático, para entendermos também como funciona a parte de redes do Docker. Para tal, vamos baixar a imagem **dockersamples/static-site**:

```
docker run dockersamples/static-site
```

Nas imagens que vimos anteriormente, as imagens *oficiais*, não precisamos colocar um *username* na hora de baixá-las. Esse *username* representa o usuário que toma conta da imagem, quem a criou. Como a imagem que vamos utilizar foi criada por outro(s) usuário(s), precisamos especificar o seu *username* para baixá-la.

Terminado o download da imagem, o *container* é executado, pois sabemos que os *containers* ficam no estado de *running* quando são criados. No caso dessa imagem, o *container* está executando um processo de um servidor web, que está disponibilizando o site estático para nós, então esse processo trava o terminal. Mas como evitamos que esse travamento aconteça?

Para tal, paramos o *container* que acabamos de criar e para impedir o travamento, nós executamos-o sem atrelar o nosso terminal ao terminal do *container*, fazendo isso através da *flag* `-d` :

```
docker run -d dockersamples/static-site
```

Assim, o *container* fica executando em segundo plano. Podemos verificar que o *container* realmente está rodando executando o comando `docker ps` :

```
alura@alura-estudio-03:~$ sudo docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED              STATUS
a6f2fab332db   dockersamples/static-site  "/bin/sh -c 'cd /u..."   About a minute ago   Up About a r
```

Mas como fazemos para acessar o site estático?

Acessando o site

Em nenhum momento dizemos onde está o site estático. Qual porta que utilizamos para acessá-lo? A **80**, conforme está na saída do `docker ps` ? Essa é a porta interna que o *container* está utilizando. Então, o que precisamos fazer é *linkar* essa porta interna do *container* a uma porta do nosso computador. Para fazer isso, precisamos adicionar mais uma *flag*, a `-P` , que fará com que o Docker atribua uma porta aleatória do mundo externo, que no caso é a nossa máquina, para poder se comunicar com o que está dentro do *container*:

```
docker run -d -P dockersamples/static-site
```

Agora, ao executar novamente o comando `docker ps` , na coluna ***PORTS***, vemos algo como:

```
PORTS
0.0.0.0:9001->80/tcp, 0.0.0.0:9000->443/tcp
```

No caso do mapeamento acima, vemos que a porta **9001** da nossa máquina faz referência à porta **80** do *container*, e a porta **9000** da nossa máquina faz referência à porta **443** do *container*. Uma outra maneira de ver as portas é utilizar o comando `docker port`, passando para ele o **id** do *container*:

```
alura@alura-estudio-03:~$ sudo docker port 989e4d7d3638
443/tcp -> 0.0.0.0:9000
80/tcp -> 0.0.0.0:9001
```

Então, se quisermos acessar a porta **80**, que é onde está o site estático, na nossa máquina, como o endereço `0.0.0.0` representa a nossa máquina local, podemos acessar o endereço `http://localhost:9001/` no navegador.

> *Caso você esteja utilizando o **Docker Toolbox**, como ele está rodando em cima de uma máquina virtual, o endereço `http://localhost:9001/` não funcionará, pois você deve acessar a porta através do **IP da máquina virtual**. Para descobrir o IP dessa máquina virtual, basta executar o comando `docker-machine ip`. Com o IP em mãos, basta acessá-lo no navegador, utilizando a porta que o Docker atribuiu, por exemplo `http://192.168.0.38:9001/`.*

Nomeando um container

Uma outra coisa interessante que é possível fazer quando estamos criando um *container* é que podemos dar um **nome** para o *container*, assim não ficamos dependendo os ids aleatórios que o Docker atribui, tornando mais fácil na hora de parar e remover o *container*, por exemplo. Para dar um nome para o *container*, utilizamos a *flag* `--name`:

```
docker run -d -P --name meu-site dockersamples/static-site
```

Assim o nome do nosso *container* será **meu-site**. Agora, para pará-lo, basta passar o seu nome para o comando `docker stop`:

```
docker stop meu-site
```

A mesma coisa seria para rodar o *container* novamente, ou para removê-lo, bastando apenas nós utilizarmos o seu nome.

Definindo uma porta específica

Uma outra coisa interessante para vermos é que, quando estamos criando um *container* e queremos *linkar* uma porta interna sua a uma porta do nosso computador, utilizamos a *flag* `-P`, para o Docker atribuir uma porta aleatória da nossa máquina, assim podemos nos comunicar com o que está dentro do *container*. Mas podemos definir essa porta, utilizando a *flag* `-p`, nesse modelo: `-p PORTA-MUNDO-EXTERNO:PORTA-CONTAINER`, por exemplo:

```
docker run -d -p 12345:80 dockersamples/static-site
```

Nesse exemplo, através da porta **12345** do nosso computador podemos acessar a porta **80** do *container*.

Atribuindo uma variável de ambiente

Além disso, podemos atribuir uma variável de ambiente no *container*. Por exemplo, a página do site estático pega o valor da variável de ambiente `AUTHOR` e o exibe junto à mensagem de *Hello*, então podemos modificar o valor dessa variável, através da *flag* `-e` :

```
docker run -d -P -e AUTHOR="Douglas Q" dockersamples/static-site
```

Quando abrirmos o site, a mensagem que será exibida é *Hello Douglas Q*.

Parando todos os containers de uma só vez

Por último, podemos ver apenas os ids dos *containers* que estão rodando, executando o comando `docker ps -q` . E com esse comando, podemos parar todos os *containers* de uma só vez. Para isso, podemos utilizar a interpolação de comandos, no padrão `$(comando)` , que executa o comando, captura sua saída e insere isso na linha de comando:

```
docker stop $(docker ps -q)
```

Então, o comando `docker ps -q` será executado e a sua saída,os ids dos *containers* que estão rodando, será inserida no comando `docker stop` , parando assim todos os *containers*.

Além disso, o comando `docker stop` demora um pouco para ser executado pois ele espera 10 segundos para parar o *container*. Podemos diminuir esse tempo através da *flag* `-t` , passando o tempo a ser aguardado, por exemplo:

```
docker stop -t 0 $(docker ps -q)
```

Com isso, exploramos um pouco mais o `docker run` .