

Macoratti.net Docker - Uma introdução básica - IV



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



Hoje veremos como **mapear** alguns recursos dos **contêineres**. ([artigo anterior](#)).

Um contêiner deve possuir um isolamento controlado mas deve poder se comunicar com o exterior e com outros contêineres de forma segura para que possamos explorar os recursos das aplicações nos contêineres.



Mapeando recursos dos contêineres

Iniciaremos mostrando como mapear uma porta de um contêiner com o mundo exterior.

1- Mapeando portas

Vamos criar um contêiner usando a imagem **nginx** e mapear a porta do contêiner

Nota: *Nginx é um servidor HTTP e proxy reverso, bem como um servidor para proxy de email IMAP/POP3.*

Digite o comando abaixo no terminal:

docker container run -p 8080:80 nginx

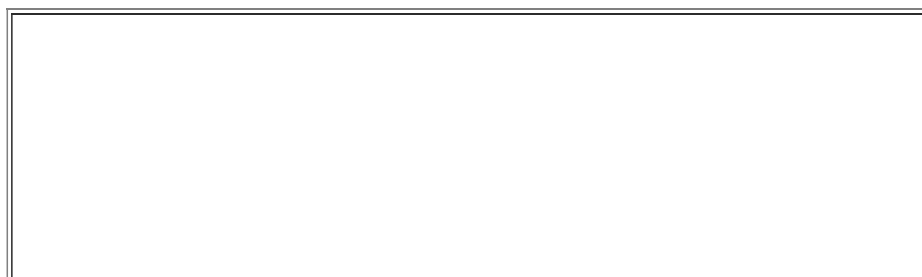
Onde:

docker container run -> comando para criar um contêiner
-p host:container -> compartilhando a porta 80 do contêiner com a porta 8080 do host
nginx -> imagem do **nginx**

```
root@linux: /home/macoratti
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti# docker container run -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
f17d81b4b692: Pull complete
d5c237920c39: Pull complete
a381f92f36de: Pull complete
Digest: sha256:b73f527d86e3461fd652f62cf47e7b375196063bbbd503e853af5be16597cb2e
Status: Downloaded newer image for nginx:latest
```

O Docker vai baixar a imagem **nginx** para o seu **host** e abrir ele na porta 8080 do seu **host**, ou seja, o **nginx** vai rodar na porta 80 dentro do contêiner mas estamos expondo para fora a porta 8080. Assim que estiver fora do contêiner vai acessar a porta 80 do contêiner a partir da porta 8080. (*ufa....*)

Para testar assim que ele finalizar a execução do comando acima, abra no seu navegador o endereço: <http://localhost:8080/> e veremos o seu retorno na imagem a baixo.



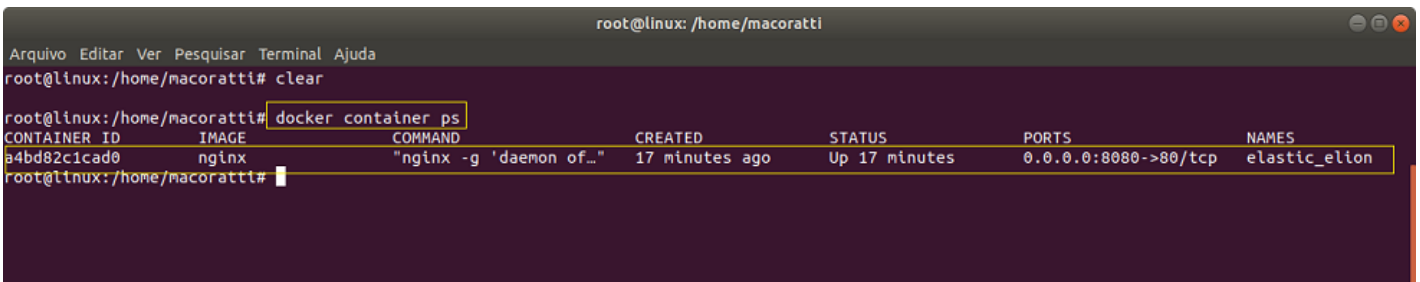


Observe que nosso console no terminal esta travado. Para executar o contêiner em segundo plano basta informar o parâmetro **-d** no comando:

docker container run -it -d -p 8080:80 nginx

Este comando vai criar o contêiner a partir da imagem **nginx** e compartilhar a porta do contêiner mas será executando em **background** liberando o terminal.

Podemos verificar que o contêiner esta em execução digitando o comando : **docker container ps**



Para parar a execução do contêiner acima digite: **docker container stop elastic_elion**

Nota: *elastic_elion* é o nome que o Docker atribui de forma aleatória e bem criativa.

Vamos agora mapear um volume do contêiner.

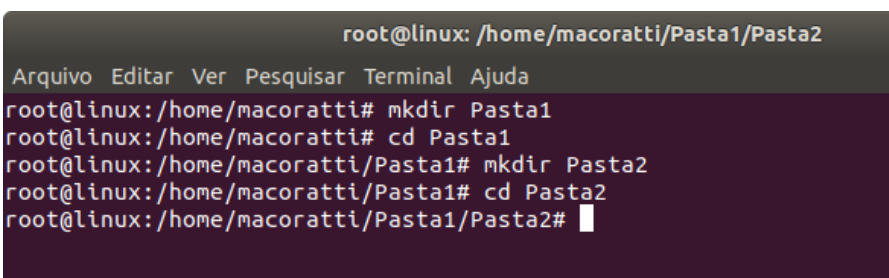
2- Mapeando volumes

Montar e mapear volumes é um assunto extenso por isso vou focar no básico.

Vamos mapear o volume de uma pasta de um computador host para a pasta em um contêiner.

Vou iniciar criando duas pastas no **host** usando o comando **mkdir** no terminal:

```
mkdir Pasta1
cd Pasta1
mkdir Pasta2
cd Pasta2
```



Criamos assim a pasta **Pasta1** e dentro dela a pasta **Pasta2** e entramos na pasta **Pasta2**.

Vamos agora criar um contêiner usando a imagem **nginx** a partir da pasta **Pasta2** :

```
docker container run -p 8080:80 -v $(pwd)/lixo:/usr/share/nginx/html nginx
```

Onde:

docker container run -> comando para criar um contêiner
-p host:container -> compartilhando a porta **80** do contêiner com a porta **8080** do host
-v \$(pwd)/lixo:/usr/share/nginx/html
nginx -> imagem do nginx

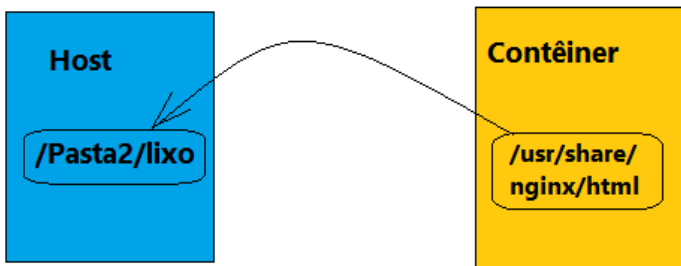
-V é o parâmetro para mapear volumes :

\$(pwd)/lixo -> a pasta atual **\$(pwd)/lixo** do **host** (*Observe que a pasta **lixo** não existe*)
/usr/share/nginx/html -> a pasta do contêiner

Como estamos executando a partir da **Pasta2** o docker vai tentar mapear a pasta **Pasta2/lixo** do host para a pasta **/usr/share/nginx/html** do contêiner.

Assim o nginx vai apontar para a pasta **Pasta2/lixo** do host e não mais para a pasta padrão.

Nota: O caminho **/usr/share/nginx/html** aponta para a pasta onde o **nginx** vai ler o arquivo **index** que ele carrega por padrão para exibir a página inicial. Assim ele vai tentar ler este arquivo na pasta **Pasta2/lixo** do host.



```
docker container run -p 8080:80 -v $(pwd)/lixo:/usr/share/nginx/html nginx
```

Note que a pasta **lixo** não existe. Fizemos isso para mostrar que vai ocorrer um erro.

Executando o comando no terminal e abrindo o navegador em : <http://localhost:8080> teremos:



Isso ocorreu porque a página **index** não foi encontrada visto que a pasta **lixo** não existe.

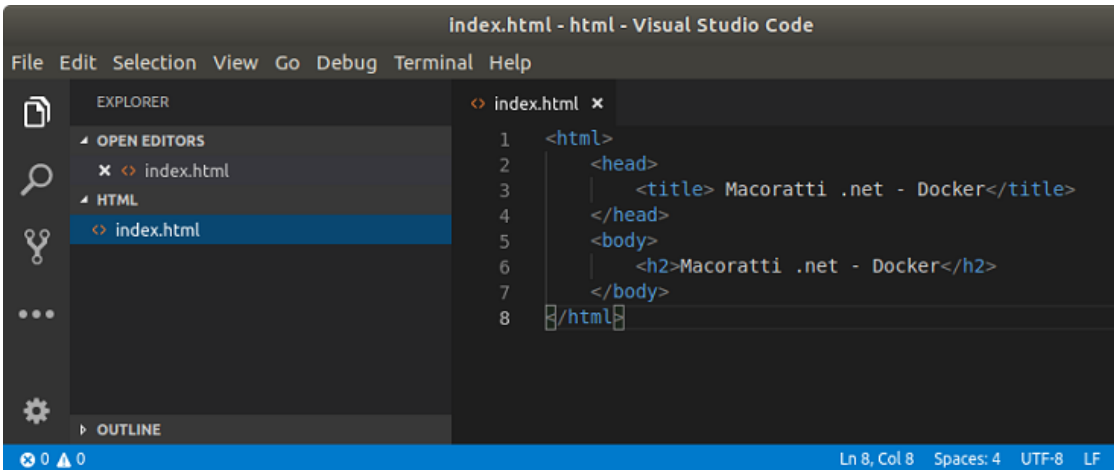
Vamos então criar um arquivo **index.html** válido em uma pasta existente e refazer o mapeamento.

Vamos criar uma pasta chamada **html** dentro da pasta **Pasta2** :

```
mkdir html  
cd html  
code .
```

```
root@linux: /home/macoratti/Pasta1/Pasta2/html
Arquivo Editar Ver Pesquisar Terminal Ajuda
root@linux:/home/macoratti/Pasta1/Pasta2# mkdir html
root@linux:/home/macoratti/Pasta1/Pasta2# cd html
root@linux:/home/macoratti/Pasta1/Pasta2/html# code .
```

Abra o [Visual Studio Code](#)(ou outro editor de sua preferência) e crie um arquivo **index.html** na pasta **html** digitando os comandos abaixo:



```
index.html - html - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXPLORER
  OPEN EDITORS
    x index.html
  HTML
    index.html
  OUTLINE
  0 0
Ln 8, Col 8 Spaces: 4 UTF-8 LF
```

```
1 <html>
2   <head>
3     <title> Macoratti .net - Docker</title>
4   </head>
5   <body>
6     <h2>Macoratti .net - Docker</h2>
7   </body>
8 </html>
```

Agora temos o arquivo **index.html** dentro da pasta **/Pasta1/Pasta2/html/** no host.

Vamos refazer o mapeamento :

docker container run -p 8080:80 -v \$(pwd)/html:/usr/share/nginx/html nginx

Agora estamos mapeando para a pasta **html** dentro da pasta atual que é a pasta **Pasta2**.

Após executar o comando abrindo o navegador em <http://localhost:8080> iremos obter:



Vemos assim o conteúdo do arquivo **index.html** que criamos na pasta **html**, e, que graças ao mapeamento, estamos acessando a partir do contêiner.

Vimos assim como mapear **portas e volumes** de um contêiner.

Na [próxima aula](#) vamos ver como tratar e **criar imagens**.