Criando imagens

O Docker pode construir automaticamente imagens lendo as instruções de um arquivo `Dockerfile`.

Dockerfile

O arquivo `Dockerfile` funciona como uma receita de bolo contendo as instruções de configuração para a configuração e execução do seu contêiner.

Neste exemplo, criaremos uma imagem de servidor web que exibirá uma página personalizada.

Servidor web

Crie uma pasta para armazenar os arquivos necessários para nossa imagem:

```
# acesando o home do usuário
cd ~

# criando a pasta
mkdir web

# entrando na pasta criada
cd web
```

Crie um arquivo chanado `index.html` com o seguinte conteúdo:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Página em manutenção</title>
</head>
<body>
<article>
    <h1>Página de Teste!</h1>
    <div>
        <p>Esta é uma página de teste.</p>
    </div>
</article>
```

```
</body>
</html>
```

Crie um arquivo chamado `Dockerfile` com o seguinte conteúdo:

```
# imagem base
FROM  ubuntu:oracular

# atualizando os pacotes e instalando o pacote nginx (servidor web)
RUN apt update && apt install -y nginx

# expondo a porta onde será executado o serviço no container
EXPOSE 80

# copiando o arquivo local para pasta dentro do container
COPY index.html /var/www/html

# comando que é executado, por padrao, quando o container se inicia
CMD ["nginx", "-g", "daemon off;"]
```

Criando a imagem

Dentro da pasta contendo o `Dockerfile`, execute o comando de build:

```
docker build -t nginx-treinamento .
```

Em caso de sucesso, você verá algo como:

```
(...)

 => => writing image
sha256:4ba7dcc4bf31d3a97cee54016702f9565b5c203f62d624136b7cdc3e0cf9090d
                0.0s
 => => naming to docker.io/library/nginx-treinamento
```

A imagem docker foi criada com o nome `docker.io/library/nginx-treinamento` ou simplesmente `nginx-treinamento:latest`.

Mostrar as imagens locais disponíveis:

```
docker images
```

Saída:

```
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
nginx-treinamento   latest    4ba7dcc4bf31   About a minute ago   123MB
```

- `nginx-treinamento`: nome da imagem
- `latest`: tag da imagem

> Quando listamos as imagens, o nome do repositório padrão `docker.io/library/` é suprimido.

Executando a imagem

Execute o seguinte comando:

```
docker run \
    -it \
    -d \
    --name="web" \
    -p 9595:80 \
    --restart="always" \
    nginx-treinamento
```

> A porta `9595` é onde o serviço ficará acessível no host, e a porta `80` é onde o serviço está disponível dentro do contêiner.

> **Opcional**: se a porta utilizada no nosso exemplo já tiver sendo utilizado por outro serviço, você pode executar o contêiner com o mapeamento para outra porta (Exemplo: `9090`) altere a linha `-p` para:

```
    -p 9090:80
```

Listando os contêiners ativos:

```
docker ps
```

Você verá algo como:

CONTAINER_ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
14a72c95d77b	nginx-treinamento	"nginx -g 'daemon ..."	1 minute ago	Up	0.0.0.0:9595->80

Para testar, acesse pelo navegador o endereço: http://localhost:9595/.

Você deverá ver algo como:



Página de Teste!

Esta é uma página de teste.

Página de teste acessada pelo navegador

Vamos remover este contêiner:

```
docker rm -f web
```

Saída:

```
web
```

Acessando os arquivos dentro do contêiner

Vamos recriar este mesmo contêiner do exemplo anterior, no entanto, desta vez vamos adicionar uma montagem de volume, para que possamos realizar a edição da página inicial que criamos e visualizar o reflexo destas alterações na página no navegador.

Criando novamente o contêiner:

```
docker run \
  -it \
  -d \
  --name="web" \
  -v ~/web:/var/www/html \
  -p 9595:80 \
  --restart="always" \
  nginx-treinamento
```

> A pasta `~/web` refere-se à localização da pasta dentro do host, a pasta `/var/www/html` refere-se à localização da pasta dentro do contêiner.

Acesse o navegador novamente.

Vamos editar o arquivo `index.html` localizado na pasta `~/web` e conferir se as alterações se refletem no navegador web.

Alterar esse trecho de:

```
<p>Esta é uma página de teste.</p>
```

Para:

```
<p>Esta é uma página de teste, agora com alterações</p>
```

Acesse o navegador novamente e atualize (atalho `F5`).

Se tudo ocorreu como esperado, você irá visualizar:



Página de teste com alterações

A partir de agora, você está pronto para realizar as alterações desejadas nos seus arquivos na sua área de desenvolvimento.

Removendo o contêiner

Após finalizar os testes, remova o contêiner:

```
docker rm -f web
```

Depois disso, caso deseje, remover os arquivos utilizados para nosso teste, apagar a pasta criada:

```
rm -rf ~/web
```

Referências

https://docs.docker.com/reference/dockerfile/

Revision #11
Created 2024-07-02 19:47:41 UTC by Éverton de Oliveira Paiva
Updated 2024-07-05 11:17:18 UTC by Éverton de Oliveira Paiva