Transcrição

> Nessa aula vamos usar um exemplo escrito Node.js. O código desse projeto pode ser baixado aqui (https://s3.amazonaws.com/caelum-online-public/646-docker/03/projetos/volume-exemplo.zip).

Já vimos que o que escrevemos no volume (pasta **/var/www** do *container*) aparece na pasta configurada da nossa máquina local, que no vídeo anterior foi o Desktop. Mas podemos pensar o contrário, ou seja, tudo o que escrevemos no Desktop será acessível na pasta **/var/www** do *container*.

Isso nos dá a possibilidade de implementar localmente um código de uma linguagem que não está instalada na nossa máquina, e colocá-lo para compilar e rodar dentro do *container*. Se o *container* possui Node, Java, PHP, seja qual for a linguagem, não precisamos tê-los instalados na nossa máquina, **nosso ambiente de desenvolvimento pode ser dentro do *container***.

É isso que faremos, pegaremos um código nosso, que está na nossa máquina, e colocaremos para rodar dentro do *container*, utilizando essa técnica com volumes.

Rodando código em um container

Para isso, vamos usar um exemplo escrito **Node.js**, que pode ser baixado aqui (https://s3.amazonaws.com/caelum-online-public/646-docker/03/projetos/volume-exemplo.zip). Até podemos executar esse código na nossa máquina, mas temos que instalar o Node na versão certa em que o desenvolvedor implementou o código.

Agora, como fazemos para criar um *container*, que irá pegar e rodar esse código Node que está na nossa máquina? Vamos utilizar os volumes. Então, vamos começar a montar o comando.

Primeiramente, como vamos rodar um código em Node.js, precisamos utilizar a sua imagem:

```
docker run node
```

COPIAR CÓDIGO

Além disso, precisamos criar um volume, que faça referência à pasta do código no nosso Desktop:

```
docker run -v "C:\Users\Alura\Desktop\volume-exemplo:/var/www" node
```

COPIAR CÓDIGO

Agora, para iniciar o seu servidor, executamos o comando `npm start`. Para executar um comando dentro do *container*, podemos iniciá-lo no modo interativo ou passá-lo no final do `docker run`:

```
docker run -v "C:\Users\Alura\Desktop\volume-exemplo:/var/www" node npm start
```

COPIAR CÓDIGO

Por fim, esse servidor roda na porta **3000**, então precisamos *linkar* essa porta a uma porta do nosso computador, no caso a **8080**. O comando ficará assim:

```
docker run -p 8080:3000 -v "C:\Users\Alura\Desktop\volume-exemplo:/var/www" node npm start
```

COPIAR CÓDIGO

Executado o comando, recebemos um erro. Nele podemos verificar a seguinte linha:

```
npm ERR! enoent ENOENT: no such file or directory, open '/package.json'
```

COPIAR CÓDIGO

Isto é, o **package.json** não foi encontrado, mas ele está dentro da pasta do código. O que acontece é que o *container* não inicia já dentro da pasta **/var/www**, e sim em uma pasta determinada pelo próprio *container*. Por exemplo, se a imagem é baseada no Ubuntu, o *container* iniciar no *root*.

Então devemos especificar que o comando `npm start` deve ser executado dentro da pasta **/var/www**. Para isso, vamos passar a *flag* `-w` (*Working Directory*), para dizer em qual diretório o comando deve ser executado, a pasta **/var/www**:

```
docker run -p 8080:3000 -v "C:\Users\Alura\Desktop\volume-exemplo:/var/www" -w "/var/www" node npm
```

COPIAR CÓDIGO

Agora, ao acessar a porta **8080** no navegador, vemos uma página exibindo a mensagem **Eu amo Docker!**. E para testar que está mesmo funcionando, podemos editar o arquivo **index.html** localmente, salvá-lo e ao recarregar a página no navegador, a nova mensagem é exibida! Ou seja, podemos criar um ambiente de desenvolvimento todo baseado em *containers*, o que ainda facilita o trabalho da nossa equipe, já que se todos utilizarem o *container*, todos terão o mesmo ambiente de desenvolvimento.

Melhorando o comando

Por fim, sabemos que podemos executar o Docker de qualquer local da nossa máquina, então podemos executar o comando que fizemos dentro da pasta do nosso projeto. Fazendo isso, podemos melhorar esse comando com o auxílio da interpolação

de comandos, já que o comando `pwd` retorna o nosso diretório atual:

```
alura@alura-estudio-03:~$ cd Desktop/volume-exemplo/
alura@alura-estudio-03:~/Desktop/volume-exemplo$ pwd
/home/alura/Desktop/volume-exemplo
```

COPIAR CÓDIGO

Assim, ao invés de passar o diretório físico para dentro do comando `docker run`, podemos utilizar a interpolação de comandos, e interpolar o comando `pwd`, assim a sua saída será capturada e inserida dentro do `docker run`:

```
alura@alura-estudio-03:~$ cd Desktop/volume-exemplo/
alura@alura-estudio-03:~/Desktop/volume-exemplo$ pwd
/home/alura/Desktop/volume-exemplo
alura@alura-estudio-03:~/Desktop/volume-exemplo$ docker run -p 8080:3000 -v "$(pwd):/var/www" -w "/
```

COPIAR CÓDIGO

Assim, vimos como rodar um código local, que está na nossa máquina, dentro de um *container*, utilizando a tecnologia dos volumes, *linkando* a nossa pasta local com uma pasta do *container*, criando assim um ambiente de desenvolvimento todo baseado em *containers*.