04

Rodando a aplicação via Docker

Transcrição

[00:00] Oi, bem-vindos de volta ao curso de Spring Boot na Alura. Agora que aprendemos como definir o arquivo Docker file no nosso projeto com Spring Boot e criar a imagem Docker no terminal usando o comando `docker build`, já estamos com tudo pronto e já podemos criar o nosso contêiner, rodar nossa aplicação utilizando um contêiner Docker.

[00:20] E é justamente isso que vamos ver nesse vídeo. É bem simples. Para rodar um contêiner tem o comando `docker run` seguido do nome da imagem, que no nosso caso é `alura/forum`, então `docker run alura/forum`. Bem simples. Então `docker run` seguido do nome da imagem que ele vai se basear para criar o contêiner.

[00:40] Esse tinha sido o nome da tag que utilizamos na hora de gerar a imagem Docker da nossa aplicação com Spring Boot.

[00:46] Se tudo der certo ele vai começar a inicializar. E já teve um problema. Vou dar um "Ctrl + C" para interromper o processo. O problema foi a questão do *profile*.

[00:57] Nós não definimos em nenhum momento para o Docker, para nossa aplicação dentro do container qual era o *profile* ativo do momento. Então ele cai naquele problema, ele não encontra um *profile* ativo e usa o *default*. Vou limpar a tela.

[01:11] Então na verdade, vamos rodar esse comando de outra maneira. Eu preciso dizer para ele algumas outras informações, como qual é o *profile* ativo.

Nós vamos simular aquele ambiente de produção, então vou passar *profile* prod.

[01:21] Só que lembra que no *profile* de produção eu preciso passar aquelas variáveis de ambiente, do DataSource, do JSON Web Token, que vimos na última aula.

[01:30] Eu tenho o comando copiado, eu vou só mostrar ele para vocês, para não perder muito tempo. Esse é o comando que vamos executar.

```
FROM openjdk:8-jdk-alpine
RUN addgroup -S spring && adduser -S spring -G spring
USER spring:spring
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

COPIAR CÓDIGO

docker run -p 8080:8080 -e FORUM_DATABASE_URL='jdbc:h2:mem:alura-forum' -e FORUM_DATABASE_USERNAME='sa' -e FORUM_DATABASE_PASSWORD='' -e FORUM_JWT_SECRET='123456' alura/forum

[01:43] Esse é o comando: `docker run -p 8080:8080` , porque o container vai rodar a nossa aplicação na porta 8080 e eu quero expô-la na minha máquina, no *host* na porta 8080, porque conseguimos testar do nosso browser a aplicação que está rodando dentro do contêiner.

[02:05] E para passar uma variável de ambiente nós usamos o parâmetro `-e` , seguido do nome da variável e igual ao valor entre aspas. Então `docker run –p 8080:8080 –e FORUM_DATABASE_URL=’jdbc:h2:mem:alura-forum’` .

[02:23] Então são aquelas mesmas variáveis que vimos na última aula. A URL do banco, o *login*, a senha, o *secret* do JSON Web Token. São esses os

parâmetros que vamos passar para o Docker na hora de rodar o comando para ele criar o contêiner.

[02:38] E além desses está faltando também mais um parâmetro, que é o do *profile*. Nós podemos passar com o `-e` também: `-e SPRING_PROFILES_ACTIVE='prod'`.

[02:55] Então esse é o comando que vamos executar no terminal. Vamos pedir para o Docker rodar, expondo a porta 8080 do contêiner na mesma porta no nosso *host*; e as variáveis de ambiente. Qual é o *profile* e as variáveis que o *profile* de produção precisa, que estão configurados no `application.properties` de produção.

[13:12] Vamos dar um "Ctrl + C", volto no terminal e vou rodar esse comando dessa maneira. E vamos ver se ele vai conseguir subir o contêiner, vai ler essas variáveis de ambiente, vai expor na porta 8080.

[03:26] Só cuidado, porque como vamos usar a porta 8080 certifique-se de que no Eclipse, na sua IDE não está rodando o projeto, porque senão vai dar conflito de portas. Então no meu Eclipse está parado, não está rodando nada.

[03:39] Ele começou a subir o contêiner e agora já pegou o *profile* de produção. Ele leu a variável corretamente do *profile*.

[03:48] Se ele tiver lido também essas outras variáveis não vai dar nenhuma *exception*, ele vai conseguir passar essas variáveis para o `application.properties` de produção; vai rodar o projeto; vai subir tudo e vai expor na porta 8080.

[04:01] E tem aquele endpoint `/topicos`, que nós conseguimos testar até do próprio navegador, não precisa ser via Postman.

[04:10] É só abrir o navegador e entrar `localhost:8080/topicos`; lembra que tinha essa URL `/topicos` via método GET como `permitAll`. Então ele não te

autenticação nem autorização. Entrou nessa URL, ele vai devolver o JSON com todos os tópicos que estão cadastrados no banco de dados.

[04:29] Ele pegou o Data source: `jdbc:h2:mem:alura-forum` . Então ele leu as variáveis certas que passamos no comando na hora de subir o contêiner.

[04:39] Ele vai carregar, inicializar, vai subir tudo, configurar conforme está na nossa aplicação, criar a imagem.

[04:48] Só vai dar problema por causa do Spring Boot Admin, que eu não estou rodando. Nós poderíamos criar também um contêiner para o Spring Boot Admin, enfim; não vai ser o foco, só para não complicar muito, focar só na parte do Docker da aplicação em si.

[05:04] Ele vai inicializar. Só está demorando bastante porque o meu computador está um pouco lento, mas se a máquina for um pouco mais rápida ele vai rodar mais rápido. Então não vai demorar tanto assim.

[05:18] Ele já começou a criar as tabelas, fazer os scripts do Hibernate e tudo mais.

[05:25] Então já vou até abrindo o navegador e vamos entrar no endereço "localhost:8080/topicos", que é aquela URL para carregar todos os tópicos. E ele já está se comunicando com o contêiner.

[05:43] Rodou, fez o *select* do curso, do Join corretamente. E funcionou, abriu no meu navegador, carregou tudo corretamente.

[05:55] Então no meu navegador, no *host* na porta 8080 ele está apontando para a 8080 do contêiner. Então funcionou tudo corretamente.

[06:06] Por hoje esse era o objetivo desse vídeo, mostrar como rodar um contêiner Docker baseado numa imagem da nossa aplicação com Spring Boot e

como passar as variáveis de ambiente, caso você queira passar o *profile* e queira passar variáveis que vão ser carregadas no *profile* ou pela aplicação.

[06:23] Então vejo vocês no próximo vídeo. Um abraço e até lá.