04

Utilizando o Docker

Transcrição

Atenção: Vamos utilizar o Mysql na versão 5.7.19 que é compatível com a aplicação web.

A ***Alura Sports*** nos contratou para que implementássemos um novo projeto que eles estão desenvolvendo para cadastrar produtos de artigos esportivos. No primeiro dia de reunião, eles nos passaram os arquivos do projeto para que fizéssemos a implementação deles em nossa máquina local.

Temos o diretório denominado "Projeto" contendo a pasta "loja", com os arquivos do projeto. De imediato, veremos que ele foi feito utilizando-se a linguagem **PHP**.

Além disso, por se tratar de cadastramento de artigos esportivos, devemos ter a comunicação desta aplicação com o banco de dados, para que as informações possam ser guardadas ali. Nossa aplicação utiliza o **MySQL** como banco de dados.

Legal, mas qual será o ambiente utilizado pela equipe de desenvolvimento da empresa? Será que a configuração deste ambiente é a mesma da máquina que estamos usando, e do ambiente de produção (Google Cloud)?

Questões deste tipo, como vimos no curso de Docker, podem ocasionar em conflitos relacionados a ambientes que possam ter sido configurados de formas diferentes.

Para evitarmos este tipo de problema, colocaremos a aplicação em containers com a ajuda do Docker. Criaremos um deles para o banco de dados, e outro para a aplicação web.

Como precisaremos criar estes containers estabelecendo a comunicação entre eles, utilizaremos o arquivo **Docker Compose**, que nos facilitará muito.

Abriremos o terminal e criaremos o diretório para nosso projeto, em que consolidaremos os arquivos. Vamos digitar:

```
mkdir Area\ de\ Trabalho/
cd Area\ de\ Trabalho/

mkdir Projeto/
cd Projeto/
```

Se precisamos ter este arquivo do Docker Compose, também teremos que utilizar um editor de texto - fique à vontade para escolher qualquer um de sua preferência. Durante o curso, usarei o Atom (https://atom.io/) e, então, incluiremos o seguinte comando no terminal:

```
atom docker-compose.yaml
```

Sendo que `yaml` é a linguagem utilizada pelo Docker Compose. Desta forma, pedimos para que o Atom abra este arquivo para que possamos configurá-lo. Na primeira linha de `docker-compose.yaml`, devemos informar a versão com que iremos trabalhar para a criação dos containers, e defini-los.

Atenção: a linguagem YAML exige uma indentação correta para funcionar bem. O Atom é um editor que possui indentação automática, o que nos auxilia muito neste aspecto.

Por ora, focaremos no container referente ao banco de dados. Primeiro, vamos dar um nome para sua identificação. Nossa aplicação web terá que se comunicar com este container, então, dentre os arquivos que possuímos, está `conecta.php`.

Ele nos mostra que esta aplicação web está se conectando ao banco de dados com as configurações `"db", "root", "", "loja"`. No primeiro campo, há o endereço IP do banco de dados, que no caso está sendo resolvido por `db`, nome pelo qual será procurado pela aplicação web para a realização do cadastro, da persistência dos dados.

O nome do nosso serviço do banco de dados será justamente este, e passaremos também as informações que serão vinculadas a este container.

Além disso, é preciso informar a imagem utilizada pelo `db`, que é do **MySQL na 5.7.19**. Ao criarmos o container do banco de dados, passaremos as mesmas informações que a aplicação web está utilizando para realizar a comunicação com ele. Ela se conectará ao banco de dados por meio do usuário `root`, com senha vazia, e as informações relativas aos produtos serão armazenadas no banco `loja`.

Já podemos configurar estas informações para quando o nosso container do `db`, ou seja, do banco de dados, for criado. Faremos estas configurações nas **variáveis de ambiente**, não deixando de informar sobre o banco de dados `loja`, deixando tudo como está na aplicação web.

Digitaremos:

```
version: "3.0"
services:
  db:
    image: mysql:5.7.19
    environment:
      - MYSQL_DATABASE=loja
      - MYSQL_USER=root
      - MYSQL_ALLOW_EMPTY_PASSWORD=yes
```

Maravilha! Os containers, no entanto, possuem uma vida passageira, como visto no curso de Docker. Seria aquele conceito de **efemeridade**, e isto quer dizer que se por acaso o container parar de funcionar, todas as informações que haviam sido cadastradas serão perdidas, e não queremos isto.

Por isso, no container, que é onde as informações são salvas, é necessário **mapearmos o volume** em algum lugar da máquina local, para que sempre tenhamos algum tipo de backup.

Vamos voltar ao terminal e criar um diretório chamado `volume_mysql`, em que entraremos em seguida:

```
mkdir volume_mysql
cd volume_mysql
pwd
```

Com o comando `pwd` acima, solicitamos a impressão do caminho completo, que é:

```
/home/alura/Área de Trabalho/Projeto/volume_mysql
```

Faremos o mapeamento voltando ao `docker-compose.yaml` :

```
version: "3.0"
services:
  db:
    image: mysql:5.7.19
    environment:
      - MYSQL_DATABASE=loja
      - MYSQL_USER=root
      - MYSQL_ALLOW_EMPTY_PASSWORD=yes
    volumes:
      - /home/alura/Área de Trabalho/Projeto/volume_mysql:/var/lib/mysql
```

No container, o SQL, as informações são salvas por padrão no diretório `var/lib/mysql` . Através deste mapeamento de volumes, colocamos tudo que é salvo neste diretório em `volume_mysql` , que se encontra na máquina local, garantindo que tenhamos as informações salvas mesmo se o container deixar de funcionar.

Finalizamos as configurações no nosso banco de dados! Falta fazermos o mesmo com o container da aplicação web. Vamos lá?