Prof. Bernardo Copstein

Prof. Júlio Machado

Roteiro 5: Conteiner

OBSERVAÇÃO: o roteiro que segue faz referência ao comando “docker” para execução das tarefas solicitadas. Verifique se você possui o "Docker” instalado no ambiente que estiver utilizando.

Introdução

A figura 1 mostra a situação da nossa arquitetura até o momento.

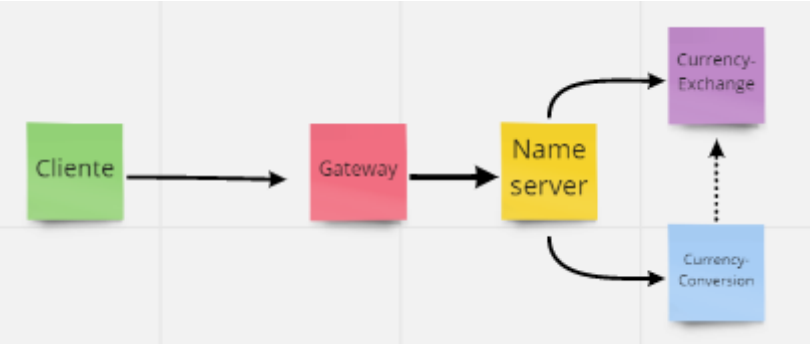

Figura 1 – Sistema desenvolvido até o momento

Até agora disponibilizamos nossa aplicação de conversão de moedas subindo cada um dos micros serviços, tanto os de infraestrutura (“gateway”, “name server”) como os de negócio, um por vez manualmente. Além disso a máquina onde estamos executando estes micros serviços possui todo o ambiente necessário para a execução deles (Java, Maven, Spring-boot, banco de dados, etc.).

Para podermos executar estes micros serviços em qualquer ambiente (servidores locais, nuvem, etc.) precisamos garantir que os recursos necessários para a execução de cada um deles estejam presentes. A melhor maneira de fazer isso é usar containers (no caso Docker containers). Criando um container a partir de uma imagem Docker básica e acrescentando nela o que mais for necessário, criamos o ambiente perfeito para a execução de cada micros serviço. Desta forma podemos executar eles em qualquer lugar, sem necessidade de configurações adicionais, desde que esse “local” tenha suporte para containers Docker (o que é o caso de todos os serviços de nuvem hoje em dia).

Outra vantagem é que podemos usar containers criados por terceiros que contenham micros serviços que possam nos interessar. Dessa forma fica fácil de agregar esse micro serviço na nossa aplicação.

Passo 1: criando um executável Java

Até agora usamos um “plug-in” do “Spring-boot” para executar nossos micros serviços. Agora vamos precisar criar um executável Java que possa ser disparado dentro do ambiente de execução da nossa aplicação baseada em micros serviços.

Inicialmente vamos criar um arquivo executável Java para o micro serviço de câmbio. Para tanto abra uma janela de "shell" na pasta que contém o arquivo "POM.xml" deste projeto. O nome original da pasta é "currency-exchange-service".

Para criar um arquivo ".jar" compile o programa usando o comando:

```
mvn clean package
```

Este comando irá apenas gerar um arquivo de terminação ".jar" contendo o "executável" do programa. Após a compilação, você irá encontrar este ".jar" na pasta "target" do projeto. Se você seguiu a nomenclatura usada nos roteiros anteriores, provavelmente o arquivo se chama "currency-exchange-service-0.0.1-SNAPSHOT.jar".

Para testar se o arquivo foi criado corretamente, vamos executar ele de forma manual. Troque o diretório corrente para a pasta "target" e então execute o arquivo ".jar" com o comando:

```
java -jar currency-exchange-service-0.0.1-SNAPSHOT.jar
```

Se tudo correu bem o micro serviço irá executar e ficará escutando na porta 8000 e você poderá enviar requisições HTTP para ele como vem fazendo até agora. Utilize uma requisição GET com a seguinte URL:

http://localhost:8000/currency-exchange/from/USD/to/INR

ATENÇÃO: caso ocorra um erro na execução, verifique se não existe um arquivo fonte de teste na pasta "src/test". Em caso positivo simplesmente delete este arquivo e recompile o programa. Caso contrário o comando "package" do maven só deixará o programa ser executado se os testes forem bem-sucedidos. Como eles não foram trabalhados, podem estar falhando.

Passo 2: criando uma imagem para o micro serviço de câmbio

Para podermos executar os diversos micros serviços que integram nossa aplicação, cada um com suas necessidades (banco de dados, arquivos etc.), será necessário criar um container Docker para cada um. Desta forma cada um irá executar em seu próprio container incluindo todos os recursos necessários. Para poder criar um container, entretanto, primeiro precisamos criar uma imagem com todos os recursos necessários.

O Docker usa um arquivo chamado "Dockerfile" para armazenar descrições de imagens. A partir dessa descrição podemos criar a imagem desejada com o comando "docker build". Para o caso do serviço de câmbio siga os passos que seguem:

a) Crie um arquivo texto nomeado "Dockerfile" com o conteúdo que segue e armazene o mesmo na pasta raiz do projeto do micro serviço de câmbio:

```
FROM eclipse-temurin:17-jdk-alpine
COPY target/*.jar app.jar
EXPOSE 8000
ENTRYPOINT [ "java", "-jar", "/app.jar" ]
```

A primeira linha diz que nossa imagem será criada a partir de uma imagem já existente no Docker Hub chamada "eclipse-temurin". Como existem várias versões para esta imagem, indicamos após os ":" que versão dela devemos usar. No caso estamos indicando a versão "17-jdk-alpine" com Linux Alpine e Openjdk 17. Sinta-se à vontade para consultar o Docker Hub e verificar a existência de imagens mais recentes se for o caso. A segunda linha copia o arquivo executável do micro serviço de câmbio para dentro da imagem. A terceira linha indica que a porta 8000 do container

deverá ser exposta. Finalmente, a quarta linha indica que o primeiro comando que deve ser executado quando o container for criado é: java -jar app.jar.

b) Crie a imagem propriamente dita usando o comando que segue. Não se esqueça que ao disparar tal comando seu "shell" deve estar aberto na mesma pasta onde se encontra o "Dockerfile", isto é, na raiz do projeto. Para criar a imagem use o comando:

```
docker build -t exchange:latest .
```

Esse comando cria uma imagem chamada exchange (opção -t (tag)) que irá sobrepor qualquer outra com este nome que já exista (latest). Você deve repetir este comando toda a vez que o programa for recompilado para gerar uma nova imagem. Você também pode usar qualquer tag que achar adequada para gerar múltiplas versões de sua imagem Docker.

c) Crie e execute o container a partir da imagem, disponibilizando o micro serviço, usando o comando que segue:

```
docker run --rm -p 8000:8000 exchange
```

A opção "--rm" indica que o container deve ser destruído quando o programa for interrompido. A opção "-p porta_externa:porta_interna" indica como deve ser feito o mapeamento entre as portas da máquina hospedeira (onde o container está rodando) e as portas do container. No caso tudo o que for enviado para a porta 8000 de "localhost" será mapeado para a porta 8000 do container. Para encerrar o micro serviço basta usar "ctrl-c" na tela de "shell".

d) Pronto, seu micro serviço está executando, escutando na porta 8000. Teste se tudo está funcionando de acordo enviado uma requisição "HTTP" através do "Postman" ou outro software equivalente. Se tudo correu bem o micro serviço irá executar e ficará escutando na porta 8000 e você poderá enviar requisições HTTP para ele como vem fazendo até agora. Utilize uma requisição GET com a seguinte URL:

http://localhost:8000/currency-exchange/from/USD/to/INR

Passo 3: criando uma imagem para cada um dos demais micros serviços

Para que possamos colocar toda a nossa aplicação baseada em micros serviços a executar em conjunto, será necessário criar uma imagem para cada um dos diferentes micros serviços. Use a tabela que segue para definir os nomes de cada um e as portas a serem expostas pelo contêiner:

Micro serviço	**Nome da imagem**	**Porta**
Câmbio	exchange	8000
Conversão de moeda	conversion	8100
Name server	nameserver	8761
API Gateway	gateway	8765

Para observar se as respectivas imagens estão disponíveis no repositório local de imagens do Docker, execute o seguinte comando:

```
docker images
```

Você deverá observar uma saída semelhante a seguinte imagem:

```
REPOSITORY                          TAG             IMAGE ID      CREATED         SIZE
localhost/gateway                   latest          7095fc911c35  23 minutes ago  409 MB
localhost/nameserver                latest          19eb270bb4ce  25 minutes ago  412 MB
localhost/conversion                latest          7fe91b2123ae  32 minutes ago  407 MB
localhost/exchange                  latest          eca03ac1e234  50 minutes ago  428 MB
docker.io/library/eclipse-temurin   17-jdk-alpine   16f6b40d1b0e  8 days ago      360 MB
```

Verifique se os containers executam corretamente. Não se preocupe que a comunicação entre o serviço de conversão e o serviço de câmbio não irá funcionar. No próximo roteiro iremos verificar como fazer para todos os containers compartilharem a mesma rede.