Prof. Bernardo Copstein

Prof. Júlio Machado

Roteiro 6: Composição de Contêiners

OBSERVAÇÃO: o roteiro que segue referência o comando “docker compose” para execução das tarefas solicitadas. Caso você esteja utilizando o ambiente Linux nos laboratórios do prédio 32, utilize o comando equivalente “podman-compose”.

Introdução

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

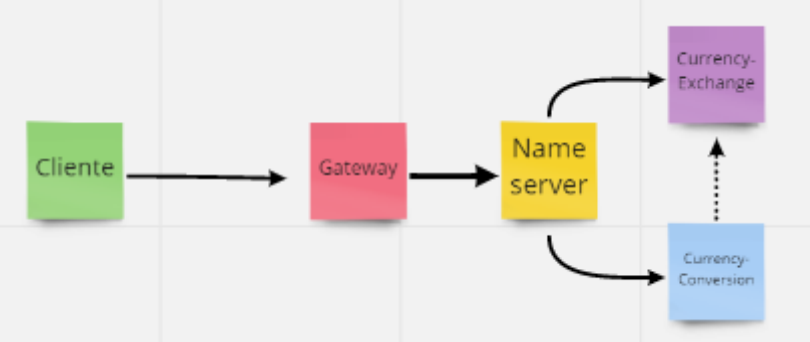

Figura 1 – Sistema desenvolvido até o momento

No roteiro anterior vimos como criar uma imagem e executar um container para cada um dos micros serviços criados. Neste roteiro, finalmente, vamos ver como disponibilizar todos os micros serviços de forma conjunta de maneira a dispor de uma aplicação.

O Docker Compose é uma aplicação que permite a definição e execução de aplicações multi-container. A definição da aplicação é feita através de um arquivo chamado “compose.yaml” (um arquivo YAML) e sua execução é disparada com um único comando.

Normalmente o Docker Compose é instalado junto com o Docker. Se ele não estiver instalado, procure a documentação de instalação junto as páginas do Docker. Teste se ele está corretamente instalado com o comando “docker compose version”.

ATENÇÃO: evite copiar e colar o código do word diretamente no VSCode para um arquivo “yaml” pois ele é sensível aos espaçamentos utilizados.

Passo 1: configurando o serviço de câmbio no “compose.yaml”

O docker compose usa o arquivo “compose.yaml” para descrever a aplicação que queremos executar. Então teremos de descrever cada um dos micros serviços que compõem nossa aplicação neste arquivo. A figura 1 apresenta a primeira versão do nosso “compose.yaml”. O ideal é que ele esteja localizado na raiz do projeto como um todo, isto é, em uma localização de onde seja possível visualizar os subdiretórios dos micros serviços que compreendem a aplicação.

```
services:
  currency-exchange:
    image: exchange
```

```
    ports:
      - "8000:8000"
    networks:
      - currency-network
    deploy:
      resources:
        limits:
          memory: 700m
networks:
  currency-network:
```

Figura 1 – Primeira versão do arquivo “compose.yaml”

Note que cada micros serviço deverá ser descrito na seção “services:”. A descrição de cada um começa com seu nome seguido de “:”. A questão do nome é muito importante e por isso mantenha os nomes que estamos usando até agora.

A configuração “image:” define que o serviço de câmbio “currency-exchange” executa a partir da imagem “exchange” que criamos manualmente no roteiro anterior

Segue-se a definição do mapeamento das portas em “ports:”. Estamos dizendo que a porta 8000 do “localhost” está sendo mapeada para a porta 8000 do container.

Também estamos dizendo que este micro serviço estará associado a uma rede chamada “currency-network” através da configuração “networks:”. A criação desta rede é comandada na seção “networks:” ao final.

Finalmente segue a definição do limite de memória reservado para este micro serviço. É na configuração de “deploy:” que muitos elementos de controle adicionais podem ser configurados para garantir o comportamento do contêiner que será criado.

Para executar o arquivo “compose.yaml” execute o seguinte comando a partir de uma janela de “shell”:

```
docker compose up
```

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

Para encerrar a execução da imagem use “ctrl-c”.

Para finalizar, digite o comando:

```
docker compose down
```

Passo 2: configurando o serviço de name server no “compose.yaml”

Para inserir o “name server” na aplicação, modifique o arquivo “compose.yaml” de acordo com a Figura 2. O trecho alterado encontra-se destacado na cor vermelha.

```
services:
  currency-exchange:
    image: exchange
    ports:
      - "8000:8000"
    networks:
      - currency-network
```

```
    deploy:
      resources:
        limits:
          memory: 700m
  naming-server:
    image: nameserver
    ports:
      - "8761:8761"
    networks:
      - currency-network
    deploy:
      resources:
        limits:
          memory: 700m
networks:
  currency-network:
```

Figura 2 – Segunda versão do arquivo “compose.yaml”

Se rodarmos o “docker compose” com arquivo assim, porém, o serviço de câmbio não vai conseguir se registrar no “name server”. Isso porque nas propriedades do serviço de câmbio indicamos que o “name server” está localizado em “http://localhost:8761/eureka”. Ocorre que cada micro serviço está executando em container próprio de maneira que “localhost” passa a ser o container local. A solução para isso é usar o próprio “nameserver” para encontrar a localização correta. Então iremos alterar a propriedade no próprio “compose.yaml” como indicado na Figura 3 (veja o trecho em vermelho).

```
services:
  currency-exchange:
    image: exchange
    ports:
      - "8000:8000"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  naming-server:
    image: nameserver
    ports:
      - "8761:8761"
    networks:
      - currency-network
    deploy:
      resources:
        limits:
          memory: 700m
networks:
  currency-network:
```

Figura 3 – Terceira versão do arquivo “compose.yaml”

A configuração “depends_on:” expressa dependências de inicialização e encerramento entre serviços. No nosso caso, o serviço “currency-exchange” deve ser colocado em atividade depois do serviço “naming-server”.

A configuração “environment:” permite a definição de pares “chave=valor” contendo parâmetros configuráveis de ambiente para o contêiner que será executado.

Para verificar se as configurações estão adequadas, execute o “docker compose up” novamente e abra a seguinte URL no seu navegador:

http://localhost:8761/

Procure observar se o micro serviço está corretamente registrado no “name server”.

Encerre completamente a aplicação antes de seguir para o próximo passo.

Passo 3: ativando o conversor de moedas e o “gateway”

Ativar o conversor de moedas neste momento é bastante simples. Basta acrescentar o trecho de configuração na Figura 4 ao arquivo “compose.yaml”. Note que neste não temos nenhuma novidade em relação ao serviço “currency-exchange”.

```
services:
  currency-exchange:
    image: exchange
    ports:
      - "8000:8000"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  currency-conversion:
    image: conversion
    ports:
      - "8100:8100"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  naming-server:
    image: nameserver
    ports:
      - "8761:8761"
    networks:
      - currency-network
    deploy:
      resources:
        limits:
          memory: 700m
networks:
  currency-network:
```

Figura 4 – Quarta versão do arquivo “compose.yaml”

Em relação ao “gateway”, vale o mesmo. Basta acrescentar a seção correspondente no arquivo “compose.yaml” de acordo com a Figura 5.

```
services:
  currency-exchange:
    image: exchange
    ports:
      - "8000:8000"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  currency-conversion:
    image: conversion
    ports:
      - "8100:8100"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  api-gateway:
    image: gateway
    ports:
      - "8765:8765"
    networks:
      - currency-network
    depends_on:
      - naming-server
    environment:
      - eureka.client.serviceUrl.defaultZone=http://naming-
server:8761/eureka
    deploy:
      resources:
        limits:
          memory: 700m
  naming-server:
    image: nameserver
    ports:
      - "8761:8761"
    networks:
      - currency-network
    deploy:
      resources:
        limits:
          memory: 700m
networks:
  currency-network:
```

Figura 5 – Quinta versão do arquivo “compose.yaml”

Para verificar se as configurações estão adequadas, execute o “docker compose up” novamente e abra a seguinte URL no seu navegador:

http://localhost:8761/

Procure observar se os micros serviços estão corretamente registrados no “name server”.

Teste com as URLs que seguem:

http://localhost:8765/currency-exchange/from/USD/to/INR

http://localhost:8765/currency-conversion/from/USD/to/INR/quantity/10

http://localhost:8765/currency-conversion-feign/from/USD/to/INR/quantity/10

Por fim, encerre completamente a aplicação.