Workshop

Papel do Microgateway na arquitetura de APIs e Microsserviços

Requisitos

Github: https://github.com/wso2/product-microgateway

- Instalar e Configurar o JAVA: https://docs.wso2.com/display/MG300/Installation+Prerequisites#InstallationPrerequisites-MicrogatewayToolkit
- Instalar o Toolkit do Microgateway e o Runtime do Microgateway https://wso2.com/api-management/api-microgateway/
- Fazer o download ou clonar o repositório com os artefatos do Workshop https://github.com/fabiolgc/workshop-api-microgateway

Criar uma API no Microgateway

1. Criar o exemplo da API no Mocky.io
    a. Pode usar o JSON Gererator para gerar o payload de exemplos ou
    b. Usar o arquivo na pasta - customers.json

2. Iniciar um novo projeto com o Microgateway
    a. micro-gw init customer-api

3. Copiar o arquivo api-definition.yaml com a definição da API - OpenAPI
    a. Arquivo: workshop-api-microgateway/customer-api-definition-1.yaml
    b. Para: <projeto-mg>/api_definitions

4. Build do MG
    a. micro-gw build customer-api

5. Executar o MG pelo runtime - local
    a. gateway <nome-arquivo-saida> - Este nome do arquivo é gerado como saída comando build. O arquivo é gerado na pasta "target" do projeto.
    b. Utilizar o comando abaixo e o token para testar a API publicada no Microgateway.

        ```
        gateway ./customer-api/target/customer-api.balx
        ```

        Utilizar o Token abaixo para testes:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ik5UQXhabU14TkRNeVpEZzNNVFU
   xWkdNME16RXpPREpoWldJNE5ETmxaRFUxT0dGa05qRmlNUSJ9.eyJhdWQiOiJodHRwOl
   wvXC9vcmcud3NvMi5hcGltZ3RcL2dhdGV3YXkiLCJzdWIiOiJhZG1pbiIsImFwcGxpY2
   F0aW9uIjp7ImlkIjoyLCJuYW1lIjoiSldUX0FQUCIsInRpZXIiOiJVbmxpbWl0ZWQiLC
   Jvd25lciI6ImFkbWluIn0sInNjb3BlIjoiYW1fYXBwbGljYXRpb25fc2NvcGUgZGVmYX
   VsdCIsImlzcyI6Imh0dHBzOlwvXC9sb2NhbGhvc3Q6OTQ0M1wvb2F1dGgyXC90b2tlbi
   IsImtleXR5cGUiOiJQUk9EVUNUSU9OIiwic3Vic2NyaWJlZEFQSXMiOltdLCJjb25zdW
   1lcktleSI6Ilg5TGJ1bm9oODNLcDhLUFAxbFNfcXF5QnRjY2EiLCJleHAiOjM3MDMzOT
   IzNTMsImlhdCI6MTU1NTkwODcwNjk2MSwianRpIjoiMjI0MTMxYzQtM2Q2MS00MjZkLT
   gyNzktOWYyYzg5MWI4MmEzIn0=.b_0E0ohoWpmX5C-M1fSYTkT9X4FN--_n7-bEdhC3Y
   oEEk6v8So6gVsTe3gxC0VjdkwVyNPSFX6FFvJavsUvzTkq528mserS3ch-TFLYiquuze
   aKAPrnsFMh0Hop6CFMOOiYGInWKSKPgI-VOBtKb1pJLEa3HvIxT-69X9CyAkwajJVssm
   o0rvn95IJLoiNiqzH8r7PRRgV_iu305WAT3cymtejVWH9dhaXqENwu879EVNFF9udMRl
   G4l57qa2AaeyrEguAyVtibAsO0Hd-DFy5MW14S6XSkZsis8aHHYBlcBhpy2RqcP51xRo
   g12zOb-WcROy6uvhuCsv-hje_41WQ==
```

6. Parar o Microgateway en execução - comando "gateway" executado no passo anterior.

Executar sua API em Docker

1. Testar o MG através do runtime em Docker - o comando "gateway" que utilizamos acima é o runtime local.

   ```
   docker run -d -v
   /Users/fabio/Documents/Workshop/customer-api/target:/home/exec/ -p
   9095:9095 -p 9090:9090 -e project="customer-api"
   wso2/wso2micro-gw:latest
   ```

2. Verificar as imagens docker e tamanhos

   ```
   docker images "wso2/wso2micro-gw"
   ```

3. Parar o container do MG em execução

   ```
   docker ps
   docker stop <CONTAINER-ID>
   ```

Adicionar outros paths a sua API

1. Mais um Microgateway. Uma outra API. Vamos utilizar mais "paths" e "endpoints" na nossa API.
    a. Copiar os arquivos workshop-api-microgateway/customer-api-definition-2.yaml para a pasta "api_definitions" do projeto do microgateway.
    b. Arquivos Origem: customer-api-definition-2.yaml
    c. Copiar para Destino: <projeto-mg>/api_definitions

2. Desabilitar Segurança acima da configuração do endpoint - não é obrigatório ser acima, é apenas como referência.

    `x-wso2-disable-security: true`

    Exemplo:

```
x-wso2-disable-security: true
x-wso2-production-endpoints:
    urls:
      - https://restcountries.eu/rest/v2
```

Gerar um Microgateway com a API em uma imagem Docker

1. Gerar uma Microgateway em imagem Docker com o runtime + API juntos:
    a. Copiar os arquivos workshop-api-microgateway/deployment-config-1.toml e micro-gw.conf para a pasta "conf" do projeto do microgateway.
    b. Arquivos Origem:
        deployment-config-1.toml
        micro-gw.conf
    c. Copiar para Destino: <projeto-mg>/conf

2. Verifique o arquivo deployment-config.toml e altere as configurações de acordo com o seu perfil e repositório Docker.

    Atenção: Quando for utilizar o recurso do "push" para o Docker Registry, você deve inserir seu usuário e senha. No meu caso estou utilizando estas configurações em variáveis de ambiente, em caso de CI/CD se este arquivo for enviado ao Github, por exemplo, usuário e senha não são expostos.

```
[docker]
  [docker.dockerConfig]
    enable = false
    name = "customer-api"
    registry = 'fabiowso2'
    tag = 'v1'
    baseImage = 'wso2/wso2micro-gw:3.0.2'
    push = false
    username = "$env{DOCKER_USERNAME}" ou "adicionar seu usuário"
    password = "$env{DOCKER_PASSWORD}" ou "adicionar seu usuário"
  [docker.dockerCopyFiles]
    enable = false
    [[docker.dockerCopyFiles.files]]
        source = '/Users/fabio/Documents/Workshop/customer-api/conf/micro-gw.conf'
        target = '/home/ballerina/conf/micro-gw.conf'
        isBallerinaConf = true
```

3. Build do MG em Docker utilizando o arquivo com as configurações do Docker

```
micro-gw build customer-api --deployment-config
/Users/fabio/Documents/Workshop/workshop-api-microgateway/deployment
-config.toml
```

4. Build do MG em Docker utilizando o arquivo com as configurações do Docker

```
docker run -d -p 9090:9090 -p 9095:9095 fabiowso2/customer-api:v1
```

5. Parar o container do MG em execução

```
docker ps
docker stop <CONTAINER-ID>
```

Interceptors: Validar, Transformar, Alterar o Request e ou Response

1. Neste ponto vamos utilizar os "interceptors". Utilizaremos os interceptors para validação de request e customização do response.
    a. Copiar o arquivo workshop-api-microgateway/customer-api-definition-3.yaml pasta "api_definitions" do projeto do Microgateway.
    b. Copiar o arquivo workshop-api-microgateway/interceptor.bal pasta "interceptors" do projeto do Microgateway.

    c. Arquivo Origem: customer-api-definition-3.yaml
    d. Copiar para Destino: <projeto-mg>/api_definitions

    e. Arquivos Origem: interceptor.bal
    f. Copiar para Destino: <projeto-mg>/interceptors

2. Alterar a configuração do arquivo de definições da API e adicionar os interceptores de "request" e "response" no recurso "name" da API.

```
      x-wso2-request-interceptor: validateRequest
      x-wso2-response-interceptor: validateResponseLength
      x-wso2-disable-security: true
      x-wso2-production-endpoints:
        urls:
          - https://restcountries.eu/rest/v2
```

3. Configuração do objeto de "Endpoints". Esta configuração nos permite configurar os endpoints, segurança, tipos, etc. Podem ser reutilizados nas definições dos "paths" da sua definição de APIs. Veja no arquivo: customer-api-definition-4.yaml

   Configuração do endpoint:

   ```
   x-wso2-production-endpoints: "#/x-wso2-endpoints/countriesAll"
   ```

```
paths:
  "/all":
    get:
```

```
    tags:
    - list
    summary: Get the list of available countries
    responses:
      '200':
        description: successful operation
      '400':
        description: invalid status value
    x-wso2-disable-security: true
    x-wso2-production-endpoints: "#/x-wso2-endpoints/countriesAll"
```

Configuração do endpoint:

```
x-wso2-production-endpoints: "#/x-wso2-endpoints/countriesName"
```

```
"/name/{name}":
    get:
      tags:
      - name
      summary: Search by country name
      description: Search by country name. It can be the native name or partial
  name
      operationId: getCountriesByName
      parameters:
      - name: name
        in: path
        description: native name or partial name
        required: true
        schema:
          type: string
      responses:
        '200':
          description: successful operation
        '400':
          description: Invalid name supplied
        '404':
          description: Country not found
      x-wso2-request-interceptor: validateRequest
      x-wso2-response-interceptor: validateResponseLength
      x-wso2-disable-security: true
      x-wso2-production-endpoints: "#/x-wso2-endpoints/countriesName"
```

Configuração do endpoint:

```
x-wso2-endpoints:
```

```
x-wso2-endpoints:
 - countriesAll:
    urls:
    - https://restcountries.eu/rest/v2/all
 - countriesName:
    urls:
    - https://restcountries.eu/rest/v2/name/
```

Referência para segurança do backend:
https://docs.wso2.com/display/MG300/Defining+a+Backend+Security+Scheme
https://swagger.io/docs/specification/authentication/

4. Substituindo Endpoints das APIs para abordagem Developer-First e processo automatizado de "deployment", por exemplo:

```
gateway -e
   countriesAll_prod_endpoint_0="https://restcountries.eu/rest/v2"
   /Users/fabio/Documents/Workshop/customer-api/target/customer-api.bal
   x
```

```
gateway -e
   countriesAll_prod_endpoint_0="https://restcountries.eu/rest/v2" -e
   countriesName_prod_endpoint_0="https://restcountries.eu/rest/v2"
 /Users/fabio/Documents/Workshop/customer-api/target/customer-api.balx
```

Logs e Troubleshooting

1. Executar o runtime com LOG:

```
gateway -e b7a.log.level=ALL ./customer-api/target/customer-api.balx
```

Referência: https://v1-0-0-beta.ballerina.io/learn/by-example/log-api.html

Deploy do Microgateway no Kubernetes

1. Gerar os artefatos para executar o Microgateway em Kubernetes
    a. Copiar os arquivos workshop-api-microgateway/deployment-config-2.toml a pasta "conf" do projeto do Microgateway. Você deve remover ou substituir por este novo arquivo o conteúdo do arquivo anterior que estava na pasta.
    b. Arquivos Origem:
        deployment-config-2.toml
    c. Copiar para Destino: <projeto-mg>/conf

    Atenção: Quando for utilizar o recurso do "push" para o Docker Registry, você deve inserir seu usuário e senha. No meu caso estou utilizando estas configurações em variáveis de ambiente, em caso de CI/CD se este arquivo for enviado ao Github, por exemplo, usuário e senha não são expostos.

```
micro-gw build customer-api --deployment-config
/Users/fabio/Documents/Workshop/customer-api/conf/deployment-config.
toml
```

2. Executar o "deploy" no kubernetes de acordo com a saída do comando "build".

```
kubectl apply -f
/Users/fabio/Documents/Workshop/customer-api/target/kubernetes/custo
mer-api
```

3. Verificar se o "deploy" no kubernetes está em execução.

```
kubectl get all
kubectl get pods
kubectl get svc
```

4. Remover o "deploy" do kubernetes.

```
kubectl delete -f
/Users/fabio/Documents/Workshop/customer-api/target/kubernetes/custo
mer-api
```

Anexo

Como importar certificado do API Manager para o Microgateway.

```
keytool -list -v -keystore
/Users/fabio/Documents/Product/wso2am-micro-gw-toolkit-3.0.2/lib/platform/
bre/security/ballerinaTruststore.p12
```

```
keytool -export -alias wso2carbon -file wso2carbon.crt -keystore
/Users/fabio/Documents/Workshop/wso2am-3.0.0/repository/resources/security
/wso2carbon.jks
```

```
keytool -import -trustcacerts -alias wso2carbon2 -file wso2carbon.crt
-keystore
/Users/fabio/Documents/Product/wso2am-micro-gw-toolkit-3.0.2/lib/platform/
bre/security/ballerinaTruststore.p12
```