
Criando Aplicações e Arquiteturas com Microservices e Kafka

Motivação e Abrangência:

Descrição Técnica:

Help:

Limpa pacotes: `mvn dependency:purge-local-repository`

Jar com main: `mvn clean package`

`-Dstart-class=com.fiap.shift.ms.microservicesdiscovery.MicroservicesdiscoveryApplication`

Main error no pom:

```
<properties>
  <!-- The main class to start by executing java -jar -->
  <start-class>com.mycorp.starter.HelloWorldApplication</start-class>
</properties>
```

`mvn install -DskipTests`

Execução Técnica:

Service Registry:

- 1) Crie um novo projeto no [SPRING INITIALIZR](#) adicionando “Eureka Server” como dependência.

SPRING INITIALIZR bootstrap your application now

Generate a with and Spring Boot

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Don't know what to look for? Want more options? [Switch to the full version.](#)

- 2) No projeto importado para o seu editor preferido, abra o arquivo pom.xml e verifique se a dependência do Eureka Server está correta.

```

28 <dependencies>
29 <dependency>
30 <groupId>org.springframework.cloud</groupId>
31 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
32 </dependency>
33

```

3) Para ativar o Eureka Server adicione o `@EnableEurekaServer` na Main da aplicação.

```

1 package com.fiap.shift.ms.microservicesdiscovery;
2
3 import org.springframework.boot.SpringApplication;
4
5 @EnableEurekaServer
6 @SpringBootApplication
7 public class MicroservicesdiscoveryApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(MicroservicesdiscoveryApplication.class, args);
11     }
12 }

```

4) Crie 2 arquivos no src/main/resources

a) application.yml

```

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
  server:
    wait-time-in-ms-when-sync-empty: 0

```

b) bootstrap.yml

```

1 server:
2   port: ${PORT:8761}
3 spring:
4   application:
5     name: ms-discovery
6

```

5) Execute:

- mvn clean install
- Acesse a pasta target
- java -jar <nomedomicroservicos.jar>
- <http://localhost:8761> Validando o Eureka Rodando.

Spring Cloud Config

6) Crie um novo projeto no [SPRING INITIALIZR](#) chamado (msconfigserver) adicionando “Config Server” e “Eureka Discovery” como dependência.

Project Metadata Artifact coordinates Group <input type="text" value="com.fiap.shift.ms"/> Artifact <input type="text" value="ms-config-server"/>	Dependencies Add Spring Boot Starters and dependencies to your application Search for dependencies <input type="text" value="Web, Security, JPA, Actuator, Devtools..."/> Selected Dependencies <div> <input checked="" type="checkbox"/> Config Server <input checked="" type="checkbox"/> Eureka Discovery </div>
---	---

- 7) No projeto importado para o seu editor preferido, abra o arquivo pom.xml e verifique se as dependências “Config Server” e “Eureka Discovery” está correta.

```
28 <dependencies>
29 <dependency>
30 <groupId>org.springframework.cloud</groupId>
31 <artifactId>spring-cloud-config-server</artifactId>
32 </dependency>
33 <dependency>
34 <groupId>org.springframework.cloud</groupId>
35 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
36 </dependency>
```

- 8) Modifique a main incluindo o suporte a Eureka Client e Config Server.

```
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.config.server.EnableConfigServer;
6 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
7 @EnableEurekaClient
8 @EnableConfigServer
9 @SpringBootApplication
10 public class MsConfigServerApplication {
11
```

- 9) Crie 2 arquivos no src/main/resources

- a) bootstrap.yml

```
1 server:
2   port: ${port:8881}
3 spring:
4   application:
5     name: ms-config-server
```

- b) application.yml

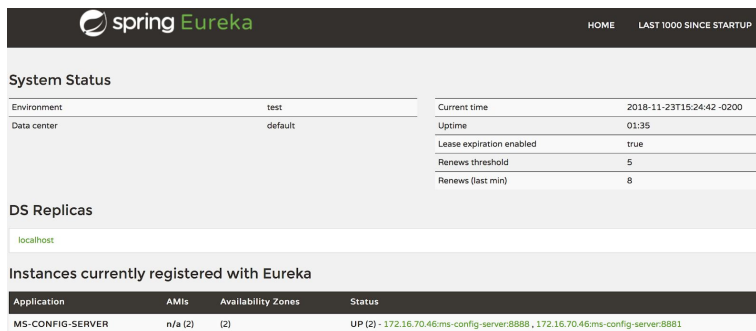
```
1 spring:
2   profiles:
3     active: native
4 ---
5 spring:
6   profiles: native
7   cloud:
8     config:
9       server:
10        native:
11          search-locations:
12            file://Users/cbajd/Base/FandS/shift/MSIKafka/V2build/ms-config-properties/{application}/{profile},
13            file://Users/cbajd/Base/FandS/shift/MSIKafka/V2build/ms-config-properties/global/{profile}
```

- 10) Execute:

- mvn clean install
- Acesse a pasta target
- java -jar <nomedomicroservicos.jar>
- <http://localhost:8881/ms-service/dev> Validando os dados retornados.

```
{ "name": "ms-service", "profiles": [ "dev" ], "label": null, "version": null, "state": null, "propertySources": [ ] }
```

- 11) Verifique no Eureka se o Config server está aparecendo.



System Status		System Status	
Environment	test	Current time	2018-11-23T15:24:42 -0200
Data center	default	Uptime	01:35
		Lease expiration enabled	true
		Renews threshold	5
		Renews (last min)	8

DS Replicas			
localhost			

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
MS-CONFIG-SERVER	n/a (2)	(2)	UP (2) - 172.16.70.46:ms-config-server:8888, 172.16.70.46:ms-config-server:8881

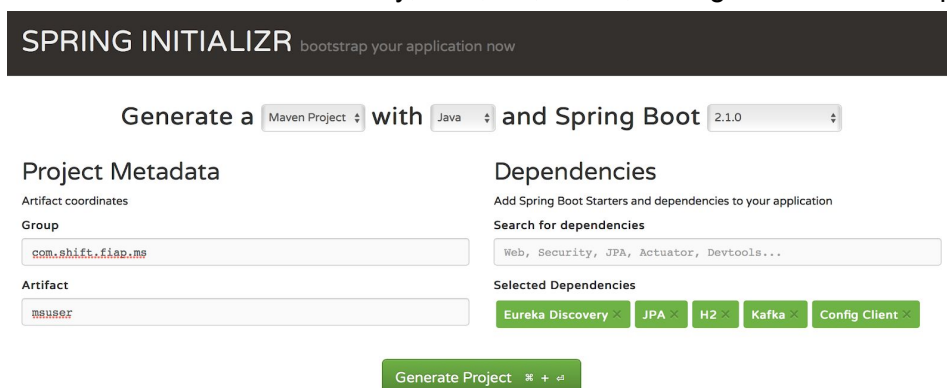
Message Broker (Kafka e Zookeeper)

12) Instale o zookeeper e o kafka.

- Acesse <https://kafka.apache.org/downloads> executando o download do binario. (ex: kafka_2.11-2.1.0.tgz)
- Descompacte com o comando: `tar -xzf <nomedoarquivo>`
- Acesse o diretório descompactado.
- Execute o comando `bin/zookeeper-server-start.sh config/zookeeper.properties` para iniciar o zookeeper.
- Em outro terminal: Execute o comando `bin/kafka-server-start.sh config/server.properties` para iniciar o kafka.

Microservices <Qualquer>

13) Crie um novo projeto no [SPRING INITIALIZR](#) chamado (O SEU MICROSERVICES) adicionando “Eureka Discovery, JPA, H2, Kafka, Config Client” como dependência.



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.1.0

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Eureka Discovery
JPA
H2
Kafka
Config Client

Generate Project

14) Confirme que todas as dependências estão no pom.

15) Insira o `@EnableEurekaClient` na main do microservices.

```

3 import org.springframework.boot.SpringApplication;
6 @EnableEurekaClient
7 @SpringBootApplication
8 public class MsuserApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(MsuserApplication.class, args);
12     }
13 }

```

16) Crie um arquivo bootstrap.yml em src/main/resources com a seguinte estrutura

```

1  server:
2    port: 8081
3  spring:
4    application:
5      name: ms-user
6    cloud:
7      config:
8        discovery:
9          enabled: true
10         service-id: ms-config-server

```

17) A configuração dos demais itens como kafka, H2 e etc deve ficar no arquivo ms-user.yml na pasta de configuracoes ms-config-properties/ms-user

```

1  spring:
2    h2:
3      console:
4        enabled: true
5        path: /h2-console
6    datasource:
7      url: jdbc:h2:mem:testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
8      username: sa
9      password:
10     kafka:
11       bootstrap-servers: localhost:9092
12       topic:
13         userCreated: USER_CREATED_TOPIC
14     security:
15       basic:
16         enabled: false

```

18) Crie os Microservices que considere necessários.

Gateway (Zuul)

19) Crie um novo projeto no [SPRING INITIALIZR](#) chamado (msgateway) adicionando “Eureka Discovery, Config Client e Zuul” como dependência.

SPRING INITIALIZR bootstrap your application now

Generate a with and Spring Boot

Project Metadata <small>Artifact coordinates</small> Group <input type="text" value="com.fiap.sift.ms"/> Artifact <input type="text" value="msgateway"/>	Dependencies <small>Add Spring Boot Starters and dependencies to your application</small> Search for dependencies <input type="text" value="Web, Security, JPA, Actuator, Devtools..."/> Selected Dependencies <input type="button" value="Eureka Discovery"/> <input type="button" value="Config Client"/> <input type="button" value="Zuul"/>
--	--

20) Confirme a configuração do POM

```

28 <dependencies>
29 <dependency>
30 <groupId>org.springframework.cloud</groupId>
31 <artifactId>spring-cloud-starter-config</artifactId>
32 </dependency>
33 <dependency>
34 <groupId>org.springframework.cloud</groupId>
35 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
36 </dependency>
37 <dependency>
38 <groupId>org.springframework.cloud</groupId>
39 <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
40 </dependency>
41
42 <dependency>
43 <groupId>org.springframework.boot</groupId>
44 <artifactId>spring-boot-starter-test</artifactId>
45 <scope>test</scope>
46 </dependency>
47 </dependencies>

```

21) Configure o zuul como client do Cloud Config, criando o arquivo bootstrap.yml no src/main/resources

```

1  server:
2    port: 8765
3  spring:
4    application:
5      name: ms-gateway
6    cloud:
7      config:
8        discovery:
9          enabled: true
10         service-id: ms-config-server

```

22) Inclua na main as seguintes notations

```

import org.springframework.boot.SpringApplication;
@EnableEurekaClient
@EnableZuulProxy
@SpringBootApplication
public class MsgatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(MsgatewayApplication.class, args);
    }
}

```

23) Execute:

- a) mvn clean install
- b) Acesse a pasta target
- c) java -jar <nomedomicroservicos.jar>

Conclusão das Atividades: