PATRÓN DE DESCUBRIMIENTO

Información general	
Duración estimada en minutos:	120
Docente:	Carlos Andrés Florez Villarraga
Guía no.	03

Información de la Guía

La transición desde un entorno monolítico a otro basado en microservicios proporciona muchos beneficios desde el punto de vista del desarrollo y mantenimiento del software. Sin embargo, al mismo tiempo su adopción conlleva la **aparición de una serie de problemas** que necesitan ser tratados correctamente.

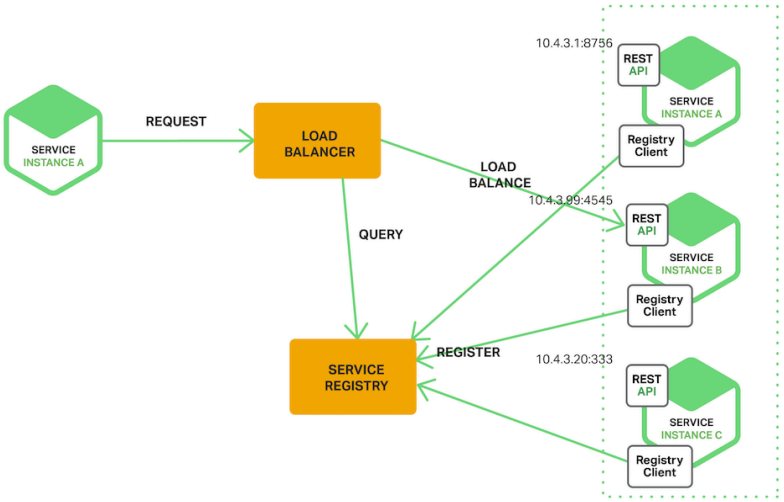
Uno de estos "nuevos" problemas es precisamente la **gestión de las direcciones** de todos los microservicios de los que consta nuestra aplicación. Dependiendo del número de microservicios, así como de las políticas de replicación de los mismos, el mantenimiento de las direcciones puede llegar a ser muy complejo.

Para lidiar con esta situación, en entornos distribuidos existe un concepto denominado Servicio de **registro y descubrimiento**. La aplicación de dicho concepto implica la creación de un servicio cuya única tarea es mantener el registro de todos los microservicios que son desplegados o eliminados. Podríamos considerarlo como una guía de teléfonos que contiene las direcciones de los microservicios. Para facilitar la implementación de dicho servicio, Spring Cloud nos ofrece el componente Spring Cloud Netflix Eureka.

Eureka es utilizado por Netflix para gestionar su propia infraestructura de servicios distribuidos en la nube, pero también está disponible para ser utilizado por otras organizaciones.

El objetivo principal de Eureka es proporcionar un registro centralizado de servicios y ayudar a los clientes a descubrir los servicios disponibles y a comunicarse con ellos de manera eficiente. Esto ayuda a simplificar el proceso de desarrollo y operación de aplicaciones distribuidas, ya que los servicios pueden registrarse en Eureka y los clientes pueden consultar la información de registro para encontrar y conectarse a los servicios necesarios, **la conexión se hace por nombres y no por direcciones ip fijas**.

Eureka también proporciona características adicionales como tolerancia a fallos y equilibrio de carga, lo que lo convierte en una herramienta importante para garantizar la disponibilidad y escalabilidad de los servicios en una arquitectura de microservicios distribuida.



¿Cómo funciona?

Cuando un microservicio registrado en Eureka arranca, envía un mensaje a Eureka indicándole que está disponible. El servidor Eureka almacenará la información de todos los microservicios registrados así como su estado. La comunicación entre cada microservicio y el servidor Eureka se realiza mediante heartbeats cada X segundos. Si Eureka no recibe un heartbeat de un tipo determinado pasados 3 intervalos, el microservicio será eliminado del registro.

Para más información:

- https://migueldoctor.medium.com/spring-cloud-series-crea-un-servicio-de-registro-y-descubrimiento-con-spring-cloud-netflix-eureka-4758615ad4cb#:~:text=C%C3%B3mo%20funciona%20Spring%20Cloud%20Netflix,solicitan%20registrarse%20en%20el%20Eureka.

- https://cloud.spring.io/spring-cloud-netflix/reference/html/

- https://docs.spring.io/spring-cloud-netflix/docs/current/reference/html/

EJERCICIO

Una biblioteca requiere un sistema de información que le permita gestionar la información de sus libros y préstamos, de cada libro se debe almacenar la información de su autor (o autores), isbn, nombre, género, año, unidades; de cada préstamo se debe registrar el código, la persona que pide el préstamo, el (o los) libros que presta, fecha del préstamo, fecha de devolución; por último, de la persona se requiere la cédula, nombre, email y teléfono.

Tome evidencia de los siguientes puntos:

1. Abra el proyecto de ejemplo de la biblioteca (`biblioteca-monolitico.zip`). Descárguelo desde el siguiente enlace:

   https://drive.google.com/file/d/1uKEoTaFUTcA_xO7er8abn536dlFD-9x4/view?usp=sharing

   Observe las clases, entidades, servicios, controladores, Dockerfile y docker-compose. Ejecute el proyecto y haga pruebas con Postman.

2. Abra el proyecto de ejemplo de la biblioteca (`biblioteca-microservicios.zip`). Descárguelo desde el siguiente enlace:

   https://drive.google.com/file/d/1y2MfJ3P3Zos-paRuuGDLXq0jq5mRWnYq/view?usp=sharing

   Observe las clases, entidades, servicios, controladores, Dockerfile y docker-compose. Ejecute el proyecto y haga pruebas con Postman.

   IMPORTANTE: Investigue acerca de RestTemplate.

3. En el proyecto de microservicios cree un nuevo módulo con el nombre **`eureka-server`**. En el `build.gradle` agregue la siguiente configuración:

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.0.2'
    id 'io.spring.dependency-management' version '1.1.0'
}

group = 'co.edu.eam'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
    maven { url 'https://artifactory-oss.prod.netflix.net/artifactory/maven-oss-candidates' }
}

ext {
    set('springCloudVersion', "2022.0.1")
}

dependencies {
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-server'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}

tasks.named('test') {
    useJUnitPlatform()
}
```

4. Cree un paquete con un nombre acorde al proyecto que estamos construyendo (`co.edu.eam.biblioteca`), y dentro del paquete agregue una clase con el nombre: `EurekaServiceApplication` con la siguiente configuración:

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaServiceApplication {

    public static void main(String[] args) {
```

```
        SpringApplication.run(EurekaServiceApplication.class, args);
    }

}
```

5. Cree, para este nuevo módulo, el archivo `application.properties` de la siguiente manera:

```
spring.application.name=eureka-server

server.port=8761
eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
```

Esto evita que el propio Eureka Server sea añadido en su directorio.

6. Ahora modifique el `build.gradle` de todos los microservicios (clientes, préstamos y libros) para agregar `netflix-eureka-client`. Así:

```
…

ext {
    set('springCloudVersion', "2022.0.1")
}

..

dependencies {
    …
    implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
    …
}

dependencyManagement {
    imports {
        mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
    }
}

…
```

Solo agregue lo que se menciona arriba, no debe borrar nada de lo que ya hay en dicho archivo.

7. En los archivo `application.properties` de los microservicios (clientes, libros y préstamos) agregue la siguiente propiedad respectivamente:

```
spring.application.name=cliente-service
```

```
spring.application.name=libro-service
```

```
spring.application.name=prestamo-service
```

Asegúrese que cada `application.properties` tenga un puerto diferente (`server.port`).

8. En los microservicios donde tenga la clase `RestTemplateConfig` debe agregar la anotación `@LoadBalanced` al método `restTemplate()` de dicha clase.

9. En el microservicio de préstamo puede modificar las rutas que están quemadas con "http://localhost:XXXX/api/etc." por el nombre acorde al servicio que requiere. Por ejemplo:

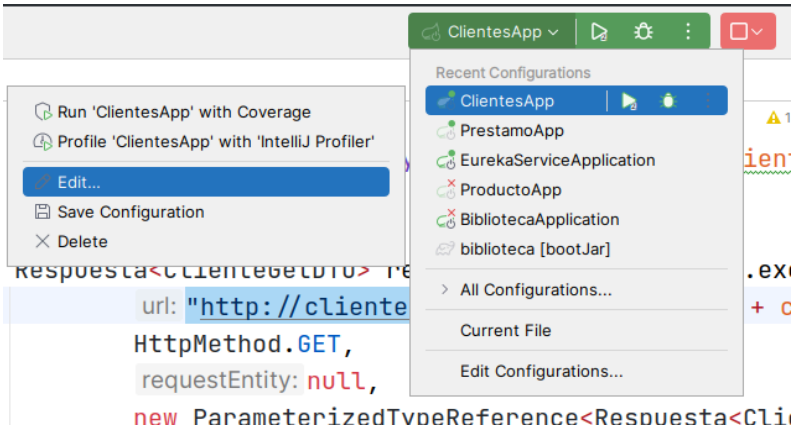
    - En lugar de usar la ruta con el nombre "http://localhost:XXXX/api/cliente" use "http://cliente-service/api/cliente".
    - En lugar de usar "http:///localhost:XXXX/api/libro" use "http://libro-service/api/libro".

10. Ejecute el proyecto de Eureka Server, luego ejecute el servicio de Clientes, Libros y Préstamos.

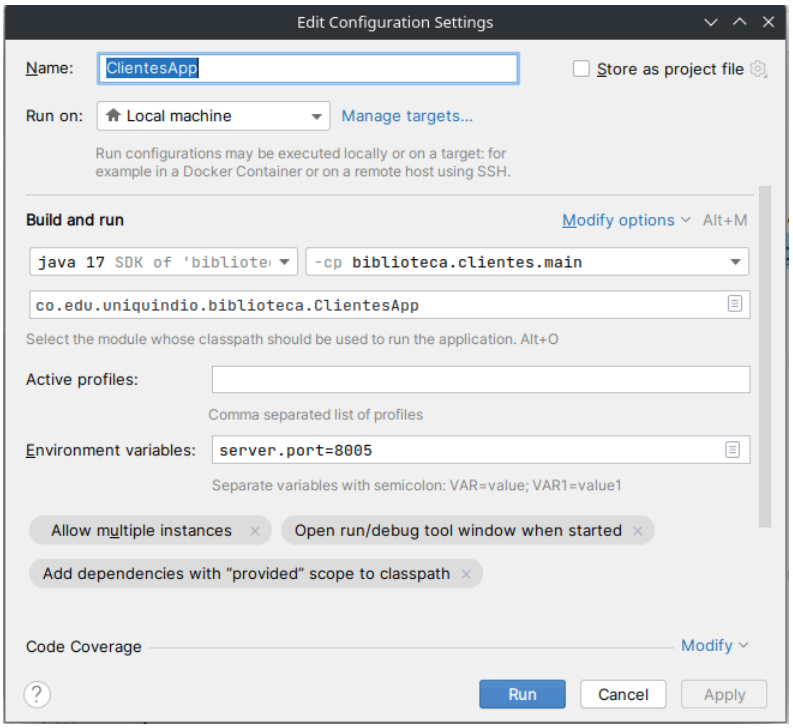
11. Ingrese a la página http://localhost:8761/ y observe los microservicios que se han registrado en el servidor de eureka.

12. Desde Postman (o la aplicación que desee) haga una prueba para crear un préstamo (verifique si se están llamando correctamente los microservicios de clientes y libros).

13. Dado que Eureka implementa un balanceador de carga por defecto, pruebe agregar otra instancia del microservicio de clientes, para esto, vaya a la parte superior donde se muestra en ejecución `ClientesApp` de click en los tres puntos y luego en Edit, así:



Luego, vaya a la parte que dice "Modify options" y habilite la opción de "Allow multiple instances". Luego, en la parte de Environment Variables agregue: `server.port=8005` (o un puerto que esté libre). Debe verse así:



Una vez hecho esto, de click en Run.

14. Observe que una vez ejecutada otra instancia del microservicio, ahora deben salir dos endpoints asociados a la aplicación "`CLIENTE-SERVICE`". Esto implica que, al hacer una petición a dicha aplicación hay dos procesos que pueden dar respuesta.

Application	AMIs	Availability Zones	Status
CLIENTE-SERVICE	n/a (2)	(2)	UP (2) - 192.168.1.12:cliente-service:8005 , 192.168.1.12:cliente-service:8001

15. Pruebe que todo funcione correctamente.

16. Modifique los puertos de los microservicios para que sean asignados aleatoriamente. En los archivos `application.properties` de los microservicios de libros y clientes agregue lo siguiente:

```
server.port=0
eureka.instance.instance-id=${spring.application.name}:${random.value}
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

Recuerde borrar la propiedad `server.port` que tenía antes en dicho archivo. Se recomienda dejar asignado un puerto fijo al microservicio de préstamos ya que por ahora será el punto de acceso a los demás microservicios.

17. Detenga todos los microservicios y ejecute de nuevo todo. Recuerde compilar los proyectos y ejecutarlos desde el `docker-compose.yml`. Verifique que todo funciona sin problemas. Con el ajuste anterior, ya no es necesario asignar la propiedad `server.port`

manual para crear varias instancias de un mismo microservicio como se hizo en el punto 12.

18. Investigue cómo se puede evidenciar que si se tienen múltiples instancias de un microservicio en ejecución, el balanceador de carga si está haciendo su trabajo correctamente.

Docker-compose con eureka

```
version: '3'

volumes:
 mysql_db_1:
 mysql_db_2:
 mongo_db:

services:

 mongo_database:
   image: mongo
   environment:
     - MONGO_INITDB_ROOT_USERNAME=root
     - MONGO_INITDB_ROOT_PASSWORD=example
   volumes:
     - mongo_db:/data/db
   ports:
     - "27017:27017"

 mongo-express:
   image: mongo-express
   restart: always
   depends_on:
     - mongo_database
   ports:
     - "8081:8081"
   environment:
     - ME_CONFIG_MONGODB_ADMINUSERNAME=root
     - ME_CONFIG_MONGODB_ADMINPASSWORD=example
     - ME_CONFIG_MONGODB_URL=mongodb://root:example@mongo_database:27017/

 mysql_database_1:
   image: mysql:8.0.34
   volumes:
     - mysql_db_1:/var/lib/mysql
   environment:
     - MYSQL_USER=user
     - MYSQL_PASSWORD=root1234
     - MYSQL_ROOT_PASSWORD=root1234

 mysql_database_2:
   image: mysql:8.0.34
   volumes:
     - mysql_db_2:/var/lib/mysql
   environment:
     - MYSQL_USER=user
     - MYSQL_PASSWORD=root1234
     - MYSQL_ROOT_PASSWORD=root1234

 eureka-server:
   image: bib_eureka_server
   build: ./eureka-server
   restart: always
   ports:
```

```
      - "8761:8761"

  clientes:
    image: bib_clientes
    build: ./clientes
    restart: always
    depends_on:
      - mongo_database
      - eureka-server
    ports:
      - "8082:8081"
    environment:
      spring.data.mongodb.uri:
mongodb://root:example@mongo_database:27017/clientes?authSource=admin
      eureka.client.service-url.defaultZone: http://eureka-server:8761/eureka

  libros:
    image: bib_libros
    build: ./libros
    restart: always
    depends_on:
      - mysql_database_1
      - eureka-server
    ports:
      - "8084:8083"
    environment:
      spring.datasource.url:
jdbc:mysql://mysql_database_1:3306/libros?createDatabaseIfNotExist=true
      spring.datasource.username: root
      spring.datasource.password: root1234
      eureka.client.service-url.defaultZone: http://eureka-server:8761/eureka

  prestamos:
    image: bib_prestamos
    build: ./prestamos
    restart: always
    depends_on:
      - mysql_database_2
      - eureka-server
    ports:
      - "8083:8082"
    environment:
      spring.datasource.url:
jdbc:mysql://mysql_database_2:3306/prestamos?createDatabaseIfNotExist=true
      spring.datasource.username: root
      spring.datasource.password: root1234
      eureka.client.service-url.defaultZone: http://eureka-server:8761/eureka
```