

# Microservicios – Arquitectura y Desarrollo

## LAB Microservicios y Netflix Eureka Registry Service

### Objetivos

- Mostrar al participante el procedimiento para usar Netflix Eureka como Registry Service con Microservicios

### Procedimiento

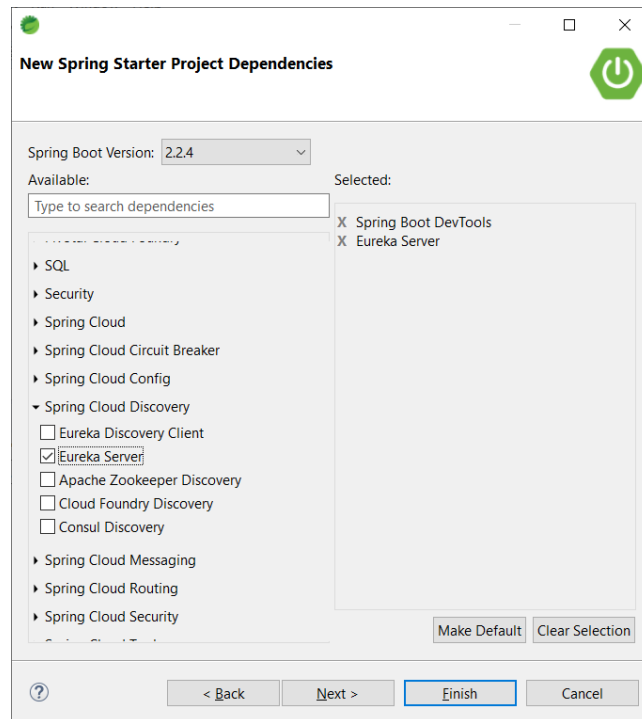
1. Crea el proyecto **springboot-servicio-eureka-server**.

The screenshot shows the 'New Spring Starter Project' dialog box. The fields are filled as follows:

- Service URL: <https://start.spring.io>
- Name: `springboot-servicio-eureka-server`
- ☒ Use default location
- Location: `C:\SpringCloud\workspace\springboot-servicio-eureka-server` (with a 'Browse' button)
- Type: `Maven`
- Packaging: `Jar`
- Java Version: `13`
- Language: `Java`
- Group: `com.cognos.app.eureka`
- Artifact: `springboot-servicio-eureka-server`
- Version: `0.0.1-SNAPSHOT`
- Description: `Spring Boot Registry Service based on Eureka`
- Package: `com.cognos.app.eureka`

At the bottom, there is a 'Working sets' section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below that is a 'Working sets:' dropdown menu and a 'Select...' button. At the very bottom, there are navigation buttons: '?', '< Back', 'Next >' (highlighted), 'Finish', and 'Cancel'.

2. Agrega las dependencias **Spring Boot DevTools** y **Eureka Server** al proyecto **springboot-servicio-eureka-server**.



3. Edita el archivo **application.properties** del proyecto.

```
application.properties
1
2 spring.application.name=servicio-eureka-server
3 server.port=8761
4
5 eureka.client.register-with-eureka=false
6 eureka.client.fetch-registry=false
7
```

4. Edita el **pom.xml** y agrega la dependencia de JAXB (Esta dependencia fue retirada en Java 11), la puedes copiar desde [https://cloud.spring.io/spring-cloud-static/Greenwich.SR5/single/spring-cloud.html#\\_jdk\\_11\\_support](https://cloud.spring.io/spring-cloud-static/Greenwich.SR5/single/spring-cloud.html#_jdk_11_support)

```
springboot-servicio-eureka-server/pom.xml
31     <scope>runtime</scope>
32     <optional>true</optional>
33 </dependency>
34 <dependency>
35     <groupId>org.springframework.boot</groupId>
36     <artifactId>spring-boot-starter-test</artifactId>
37     <scope>test</scope>
38     <exclusions>
39         <exclusion>
40             <groupId>org.junit.vintage</groupId>
41             <artifactId>junit-vintage-engine</artifactId>
42         </exclusion>
43     </exclusions>
44 </dependency>
45
46 <dependency>
47     <groupId>org.glassfish.jaxb</groupId>
48     <artifactId>jaxb-runtime</artifactId>
49 </dependency>
50
51 </dependencies>
52
53 <dependencyManagement>
54     <dependencies>
55         <dependency>
56             <groupId>org.springframework.cloud</groupId>
57             <artifactId>spring-cloud-dependencies</artifactId>
58             <version>${spring-cloud.version}</version>
59             <type>pom</type>
60             <scope>import</scope>
61         </dependency>
62     </dependencies>
63 </dependencyManagement>
64 </project>
```

5. Edita la clase **SpringbootServicioEurekaServerApplication**, habilita EurekaServer.

```
SpringbootServicioEurekaServerApplication.java
1 package com.cognos.app.eureka;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @EnableEurekaServer
8 @SpringBootApplication
9 public class SpringbootServicioEurekaServerApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(SpringbootServicioEurekaServerApplication.class, args);
13     }
14
15 }
16
```

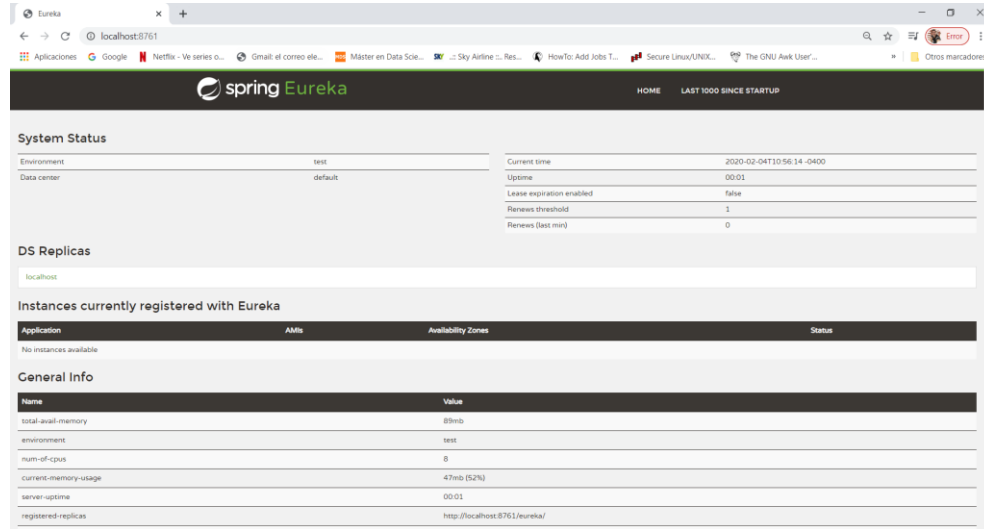
6. Ejecuta el proyecto Clic Derecho Luego Run As > Spring Boot App

```
Problems | Javadoc | Declaration | Console
springboot-servicio-eureka-server - SpringbootServicioEurekaServerApplication [Spring Boot App] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (4 feb. 2020 10:54:33)

:: Spring Boot :: (v2.2.4.RELEASE)

2020-02-04 10:54:36.734 INFO 38512 --- [ restartedMain] SpringbootServicioEurekaServerApplication : No active profile set, falling back to default
2020-02-04 10:54:37.692 WARN 38512 --- [ restartedMain] o.s.boot.actuate.endpoint.EndpointId : Endpoint ID 'service-registry' contains invalid characters
2020-02-04 10:54:37.825 INFO 38512 --- [ restartedMain] o.s.cloud.context.scope.GenericScope : BeanFactory id=f3456f2f-3159-3544-bef3-d66a809
2020-02-04 10:54:38.327 INFO 38512 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8761 (http)
2020-02-04 10:54:38.337 INFO 38512 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-02-04 10:54:38.337 INFO 38512 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.30]
2020-02-04 10:54:38.466 INFO 38512 --- [ restartedMain] o.a.c.c.f.Tomcat1 [localhost/] : Initializing Spring embedded WebApplicationContext
```

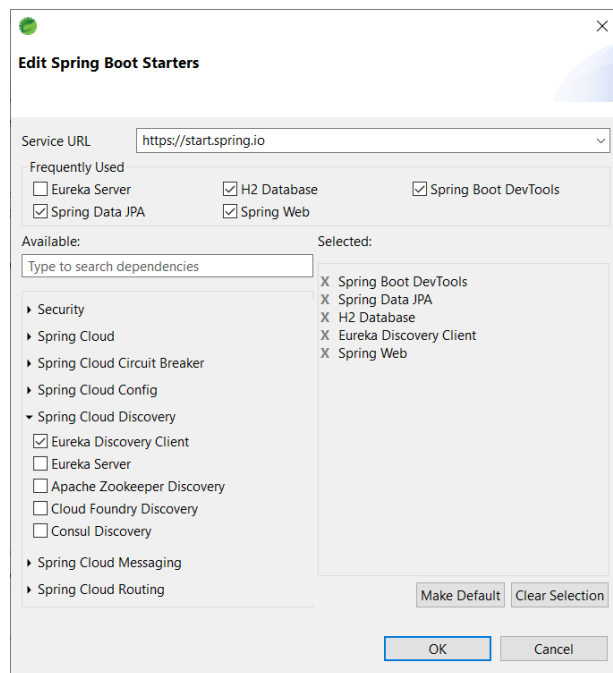
7. Abre el dashboard de Eureka Server desde <http://localhost:8761>



## Eureka Discovery Client y Microservicios

8. Agrega las dependencias “**Eureka Discovery Client**” a cada uno de los proyectos **springboot-servicio-productos** y **springboot-servicio-items**.

Sobre el proyecto Clic Derecho > Spring > Edit Starters



9. En cada proyecto habilita el servicio como cliente de Eureka. Edita la clase **SpringbootServicioItemApplication** y **SpringbootServicioProductoApplication** y agrega la anotación **@EnableEurekaClient**.

```
*SpringbootServicioItemsApplication.java
1 package com.cognos.app.items;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
6 import org.springframework.cloud.netflix.ribbon.RibbonClient;
7 import org.springframework.cloud.openfeign.EnableFeignClients;
8
9 @EnableEurekaClient
10 @RibbonClient(name="servicio-productos")
11 @EnableFeignClients
12 @SpringBootApplication
13 public class SpringbootServicioItemsApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(SpringbootServicioItemsApplication.class, args);
17     }
18 }
19
20
```

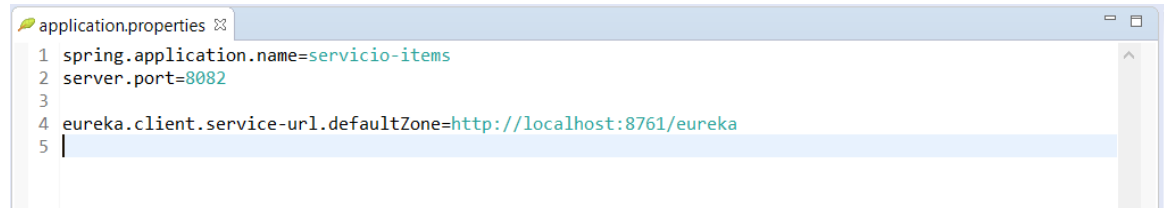
```
*SpringbootServicioProductosApplication.java
1 package com.cognos.app.productos;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
6
7 @EnableEurekaClient
8 @SpringBootApplication
9 public class SpringbootServicioProductosApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(SpringbootServicioProductosApplication.class, args);
13     }
14 }
15
16
```

10. Edita el archivo **application.properties** del proyecto **springboot-servicio-productos**, indica la localización del servidor Eureka.

```
application.properties
1
2 spring.application.name=servicio-productos
3 server.port=8081
4
5 eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

11. Edita el archivo **application.properties** del proyecto **springboot-servicio-items**, indica la localización del servidor Eureka y elimina la siguiente línea:

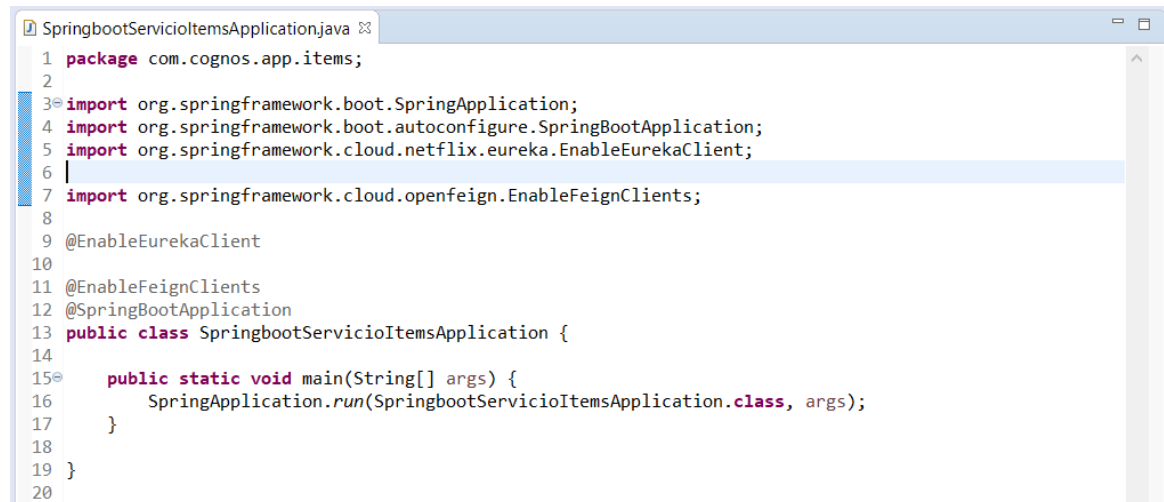
**servicio-productos.ribbon.listOfServers=localhost:8081,localhost:9081**

A screenshot of a code editor showing the 'application.properties' file. The file contains the following lines: 1 spring.application.name=servicio-items, 2 server.port=8082, 3, 4 eureka.client.service-url.defaultZone=http://localhost:8761/eureka, 5. The line number 5 is highlighted with a blue background.

```
1 spring.application.name=servicio-items
2 server.port=8082
3
4 eureka.client.service-url.defaultZone=http://localhost:8761/eureka
5
```

12. En el proyecto **springboot-servicio-items**, Edita el archivo **pom.xml** y retira la dependencia **spring-cloud-starter-netflix-ribbon**.

13. En el proyecto **springboot-servicio-items**, Edita la clase **SpringbootServicioItemsApplication** y elimina la siguiente línea: **@RibbonClient(name="servicio-productos")** y la línea **import org.springframework.cloud.netflix.ribbon.RibbonClient;**

A screenshot of a code editor showing the 'SpringbootServicioItemsApplication.java' file. The code includes package declaration, imports for SpringApplication, SpringBootApplication, EnableEurekaClient, and EnableFeignClients, followed by the class definition and main method. The line number 15 is highlighted with a blue background.

```
1 package com.cognos.app.items;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
6
7 import org.springframework.cloud.openfeign.EnableFeignClients;
8
9 @EnableEurekaClient
10
11 @EnableFeignClients
12 @SpringBootApplication
13 public class SpringbootServicioItemsApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(SpringbootServicioItemsApplication.class, args);
17     }
18 }
19
20
```

14. Inicia los proyectos primero **springboot-servicio-eureka-server** y luego los proyectos **springboot-servicio-productos** y **springboot-servicio-items**.

15. Abre el dashboard de Eureka y verifica que los Microservicios se hayan registrado.

The screenshot shows the Spring Eureka dashboard at localhost:8761. The 'Instances currently registered with Eureka' section is highlighted with a red box. It contains a table with the following data:

Application	AMIs	Availability Zones	Status
SERVICIO-ITEMS	n/a (1)	(1)	UP (1) - LAPTOP-P45M6H5U servicio-items.8082
SERVICIO-PRODUCTOS	n/a (1)	(1)	UP (1) - LAPTOP-P45M6H5U servicio-productos.8081

Below the table, the 'General Info' section shows various system metrics:

Name	Value
total-avail-memory	87mb
environment	test
num-of-cpus	8
current-memory-usage	56mb (64%)
server-up-time	00:01

16. Inicia otra instancia del proyecto **springboot-servicio-productos** en el **puerto 9001**

Clic Derecho, **Run As > Run Configuration.**

The screenshot shows the Eclipse Run Configurations dialog for the project 'springboot-servicio-productos - SpringbootServicioProductosApplication'. The 'Spring Boot' tab is selected. The 'Program arguments' field is empty. The 'VM arguments' field contains '-Dserver.port=9001'. The 'Working directory' is set to 'Default' with the value '\${workspace\_loc:springboot-servicio-productos}'. The 'Run' button is highlighted.

Clic en Run.

17. Verifica en el dashboard de Eureka que haya dos instancias de **SERVICIO-PRODUCTOS**.

The screenshot shows the Spring Eureka web interface in a browser. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. Below the header, there are three main sections:

- System Status:** A table showing environment details.
 

Environment	test	Current time	2020-02-04T11:47:19 -0400
Data center	default	Uptime	00:07
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	8
- DS Replicas:** A section with a search bar containing 'localhost'.
- Instances currently registered with Eureka:** A table listing registered services.
 

Application	AMIs	Availability Zones	Status
SERVICIO-ITEMS	n/a (1)	(1)	UP (1) - LAPTOP-P45M6H5U servicio-items.8082
SERVICIO-PRODUCTOS	n/a (2)	(2)	UP (2) - LAPTOP-P45M6H5U servicio-productos.9001, LAPTOP-P45M6H5U servicio-productos.8081
- General Info:** A table showing instance details.
 

Name	Value
total-avail-memory	87mb
environment	test
num-of-cpus	8
current-memory-usage	39mb (44%)
server-uptime	00:07

18. Consume el servicio **SERVICIO-ITEMS** con Postman y verifica que se balancea la petición entre las dos instancias del servicio **SERVICIO-PRODUCTOS**.

The screenshot shows the Postman interface. A GET request is configured to 'http://localhost:8082/listar'. The response is a JSON array of three product objects, indicating load balancing across multiple instances.

```

[
  {
    "producto": {
      "id": 1,
      "nombre": "Cap",
      "precio": 800.0,
      "created": "2020-02-04T00:00:00.000+0000",
      "port": 9001
    },
    "cantidad": 1,
    "total": 800.0
  },
  {
    "producto": {
      "id": 2,
      "nombre": "T-Shirt",
      "precio": 500.0,
      "created": "2020-02-04T00:00:00.000+0000",
      "port": 9001
    },
    "cantidad": 1,
    "total": 500.0
  },
  {
    "producto": {
      "id": 3,

```

### Nota:

Entre el servidor Eureka y las instancias de los servicios hay una señal ("heart-beat") cada 30 s. Eureka verifica que este activo el servicio (Up), si no responde en tres señalizaciones (90 s), se considera que el servicio esta caído ("down").

19. Para terminar este lab elimina de **Run Configurations** el argumento **-Dserver.port=9001**, para que la próxima no se inicie por defecto alguna instancia en el puerto 9001.



## Escalando Microservicios

En esta parte del LAB mostraremos como:

- Escalar los Microservicios de tal manera que tengamos varias instancias ejecutándose simultáneamente sin colisionar.
- Dinamizar los puertos asignados a las instancias de Microservicios de tal manera que se asignen puerto de forma aleatoria.

20. En el proyecto springboot-servicio-productos edita el archivo application.properties.

```
application.properties
1 spring.application.name=servicio-productos
2 server.port=${PORT:0}
3
4 eureka.instance.instance-id=${spring.application.name}:${spring.application.instance_id:${random.value}}
5
6 eureka.client.service-url.defaultZone=http://localhost:8761/eureka
7
8
```

21. Inicia tres instancias del servicio SERVICIO-PRODUCTOS. Verifica en el dashboard de Eureka.

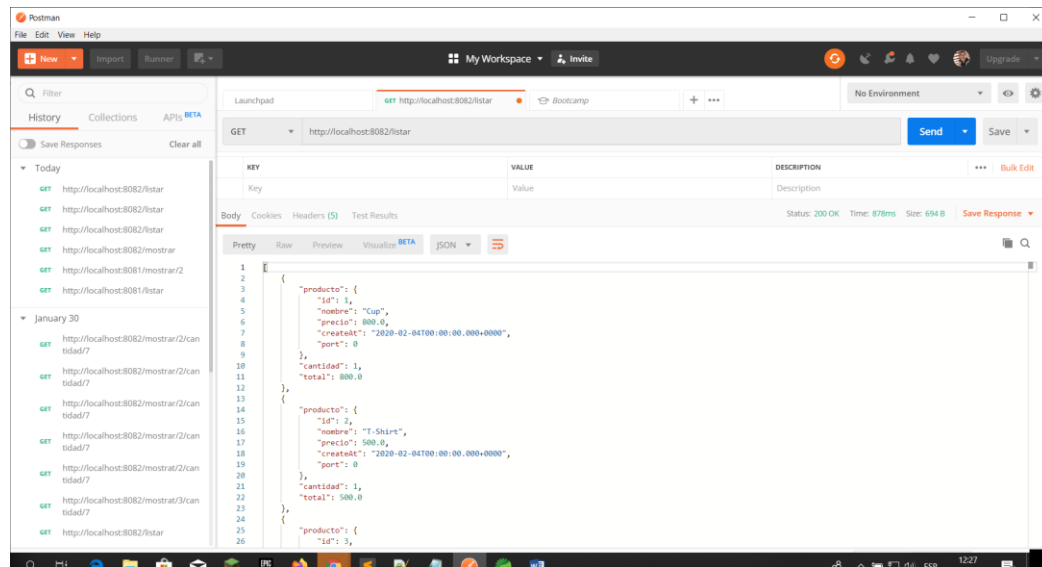
The screenshot shows the Eureka dashboard in a web browser. The 'Instances currently registered with Eureka' section contains a table with the following data:

Application	AMIs	Availability Zones	Status
SERVICIO-ITEMS	n/a (1)	(1)	UP (1) - LAPTOP-P45M6H5U-servicio-items 8082
SERVICIO-PRODUCTOS	n/a (3)	(3)	UP (3) - servicio-productos:9c3de0aa758a6c36c026f3bdf61de9c1, servicio-productos:7b6fc901c895d9b26da4b75fe14b682, servicio-productos:3bb2c576ecd10f6f719d3b57cc56fa5

The row for 'SERVICIO-PRODUCTOS' is highlighted with a red rectangle. Below this, the 'General Info' section shows system metrics:

Name	Value
total-avail-memory	87mb
environment	test
num-of-cpus	8
current-memory-usage	49mb (56%)
server-up-time	00:02

22. Consume el servicio **SERVICIO-ITEM** usando Postman y verifica que la respuesta es exitosa.



**Nota:** En cada item se muestra el valor cero para el campo **port**, esto es porque ahora el puerto se asigna de manera aleatoria. (application.properties)