Spring cloud Config

¿Qué es?

**1. Centralización de configuraciones:**

* **Spring Cloud Config** permite centralizar la configuración de todos los microservicios en un **solo lugar** (en tu caso, un repositorio Git). No tienes que ir a cada microservicio y modificar sus archivos application.yml individualmente.
* Todos los archivos .yml de tus microservicios (como api-gateway.yml, user-service.yml, etc.) se almacenan en un repositorio centralizado (como Git).

**2. Microservicios como clientes de configuración:**

* En lugar de configurar cada microservicio de forma independiente, los microservicios **consultan el Config Server** para obtener su configuración.
* En cada microservicio, solo necesitas configurar la **conexión al Config Server** (como un cliente de configuración). Esto se hace en su archivo application.yml:

Texto

El contenido generado por IA puede ser incorrecto.

**3. ¿Cómo gestionamos los cambios?**

* Si necesitas **cambiar una configuración** (por ejemplo, cambiar el puerto de un microservicio o la URL de la base de datos), **solo modificas el archivo correspondiente** en el repositorio Git central.
* Los microservicios automáticamente obtendrán los cambios sin necesidad de modificar su propia configuración, **solo necesitan conectar a ese repositorio**.

Esto significa que **no necesitas ir a cada microservicio y cambiar sus archivos de configuración manualmente**. Todo se gestiona desde el repositorio Git central.

**4. ¿Qué pasa cuando un microservicio se inicia?**

* Cuando un microservicio arranca, se conecta al **Config Server**, solicita su configuración específica (por ejemplo, el archivo api-gateway.yml), y la carga en su entorno.
* Si el microservicio ya está en ejecución y decides **actualizar su configuración** en el repositorio, puedes hacer que el microservicio recargue esa configuración sin reiniciarlo, usando el **refresh endpoint** de Spring Cloud (si está habilitado).

**Ventajas:**

* **Centralización:** Gestionas toda la configuración desde un solo lugar (el repositorio Git).
* **Consistencia:** Todos los microservicios obtienen la misma configuración, evitando inconsistencias.
* **Facilidad de mantenimiento:** Si necesitas cambiar una configuración (por ejemplo, cambiar el puerto de un microservicio), solo lo haces en un archivo y no en todos los microservicios.
* **Escalabilidad:** A medida que tu número de microservicios crece, no es necesario gestionar la configuración de cada uno de forma independiente.

**Ejemplo de flujo:**

1. Tienes un repositorio Git (por ejemplo, en GitHub) con los archivos de configuración para tus microservicios: api-gateway.yml, user-service.yml, etc.
2. Los microservicios se configuran para que apunten al Config Server (por ejemplo, http://localhost:8888).
3. Cuando un microservicio se inicia, consulta al Config Server para obtener su configuración, lo que le permite configurar el puerto, rutas, base de datos, etc., todo desde el repositorio Git.
4. Si necesitas cambiar la configuración (por ejemplo, cambiar el puerto de api-gateway), **solo lo haces en api-gateway.yml** dentro del repositorio Git.

**PRACTICA**

Creamos el proyecto: cloud-config:  
Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Luego asi nos quedaría:  
Texto

El contenido generado por IA puede ser incorrecto.

Luego configuraríamos en la clase principal

Texto

El contenido generado por IA puede ser incorrecto.

agregaríamos el github vacio que creamos y lo ponemos en el Application.yml Texto

El contenido generado por IA puede ser incorrecto.

Listo ahora arrancamos el proyecto y ya estaría

Texto

El contenido generado por IA puede ser incorrecto.

Ahora como paso siguiente es subir al github los yml de los microservicios de cada uno, donde estos estarán almacenados.

Ojo estamos trayendo los yml de los microservicios anteriores

Texto

El contenido generado por IA puede ser incorrecto.

Donde le cambiaremos el nombre por ejemplo el microservicio de servicio-consumidor:

El application.yml cambiaremos el nombre a esto ->servicion-consumidor.yml

Y asi para el resto….

Luego una vez que hayamos cambiado cada application de los microservicios. Lo subimos al github y quedari asi:  
Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Ahora lo siguiente es configurar cada microservicio yml:  
agregaríamos este pom:  
Texto

El contenido generado por IA puede ser incorrecto.

Este es el pom cliente

Y ahora configuraríamos cada application.yml de cada microservicio, asi:

Api-gateway  
Texto

El contenido generado por IA puede ser incorrecto.

Eureka-client

Texto

El contenido generado por IA puede ser incorrecto.

Servicio-consumidor:  
Texto

El contenido generado por IA puede ser incorrecto.

Luego arrancamos cada microservicio:

Primero el eureka, luego el config y por ultimo los microservicios  
Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Y no quedaría asi:  
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

No encendí el consumidor para ver que sucede xd

Veremos que ya detecta cada microservicio y todo desde el github, ahora ya no tendrás que configurar el application.yml de cada microservicio, solo con ir a github clonar el repositorio y subir tus cambios.

Y todo los yml de cada microservicio en el local, tendrán el mismo yml.

Osea lo que hace es que cada microservicio esta llamando al microservicio config (que este tiene su puerto general 8888), para realizar una petición de los yml. Una vez que hace esa petición, iran a su respectivo yml del github y arrancaran con el puerto que designaron en el github y eso es todo

EXTRA:

**Verifica que la URL que estás utilizando para acceder a la configuración de los microservicios esté correcta**:

* Si deseas ver la configuración de un microservicio en particular, accede a http://localhost:8888/<nombre-del-servicio>/<perfil>. Ejemplo:
  + Para **api-gateway**: http://localhost:8888/api-gateway/default
  + Para **eureka-client**: http://localhost:8888/eureka-client/default
  + Para **servicio-consumidor**: http://localhost:8888/servicio-consumidor/default

Texto

El contenido generado por IA puede ser incorrecto.