ESPECIALIDAD: JAVA NIVEL: AVANZADO MODALIDAD: ONLINE

Curso:

ARQUITECTURA DE MICROSERVICIOS I

(Spring Boot, Spring Cloud, Docker, Design, Patterns, Buildings, Deployments)

Jean Ramal
Instructor

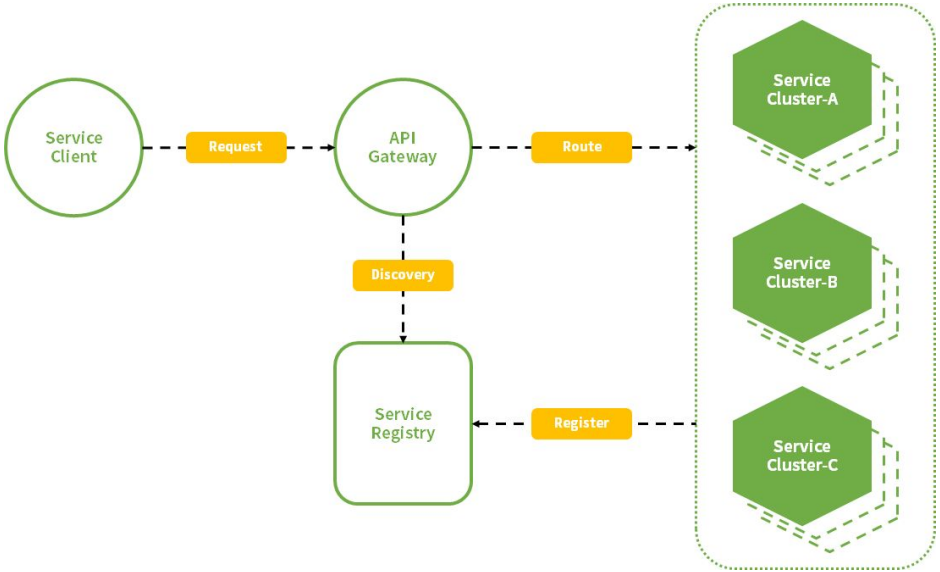
Agenda - Sesión 03

Implementación de la arquitectura de Microservicios

- Patrones a implementar: Api Gateway, Circuit Breaker, Database per Service, Application Metrics, Distributing Tracing y Health check Api.
- Spring Cloud Gateway, Resillience4J, Spring Boot Admin, Zipkin, Actuator, Docker.
- Implementar los patrones.

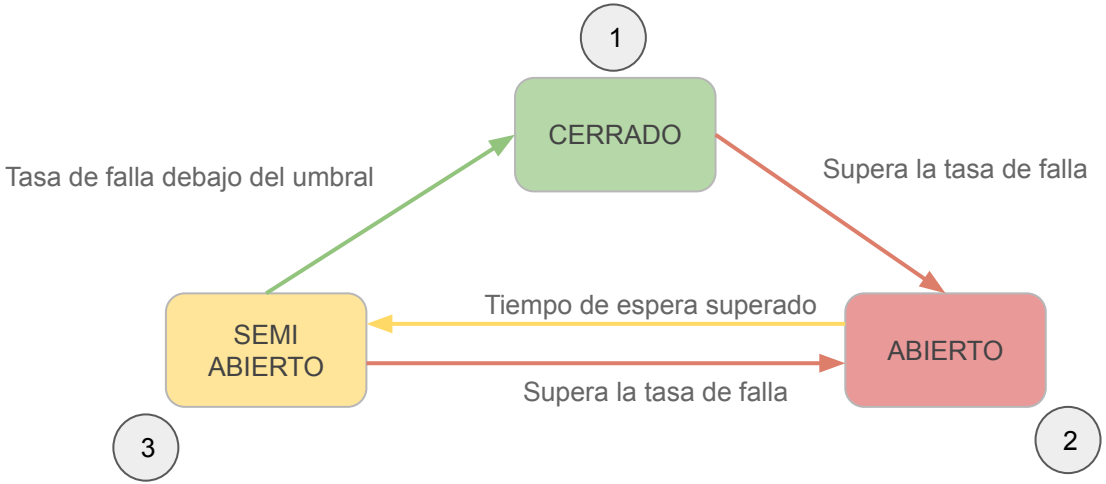
API Gateway

- Puerta de enlace, acceso centralizado
- Seguridad centralizada
- Enrutamiento dinamico
- balanceo de carga
- Manejo de filtros
- Permite extender funcionalidades
- Implementación en Spring con Zuul Netflix, **Spring Cloud Gateway**



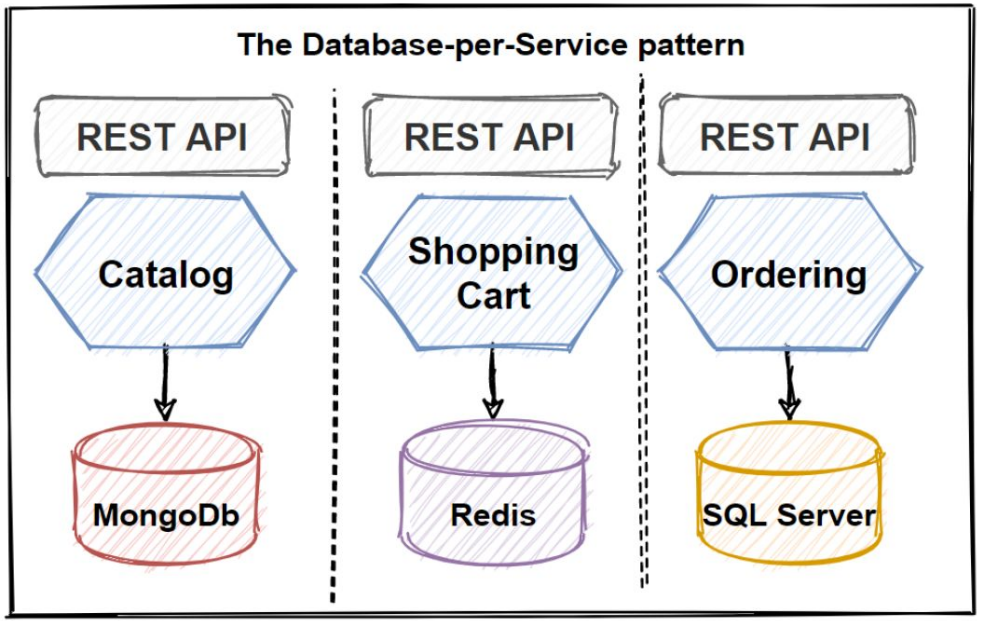
Circuit Breaker

- Implementar tolerancia a fallos y resiliencia
- Detectar fallas y encapsularlas para que no se repitan
- Brindar caminos alternos ante una falla, lentitud o indisponibilidad
- Implementación en Spring con Netflix Hystrix, **Resillence4J**, Sentinel, Spring Retry



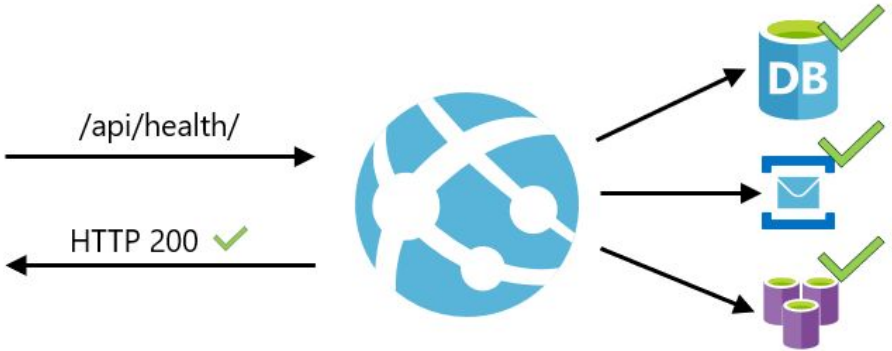
Database per Service

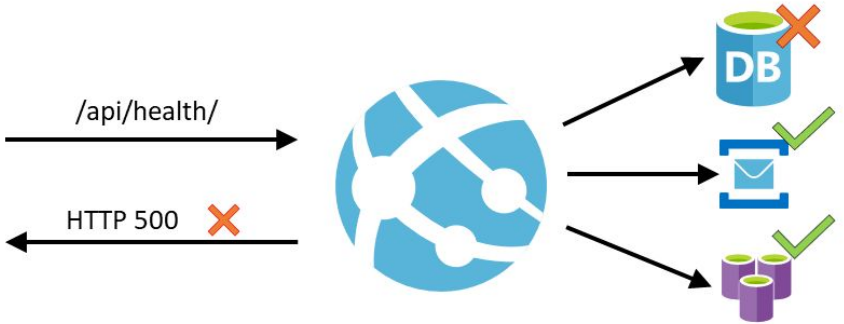
- Fuente de datos por servicio.
- Datos accedidos sólo a través del servicio.
- Implementación de las bases de datos en **Docker**.



Health Check Api

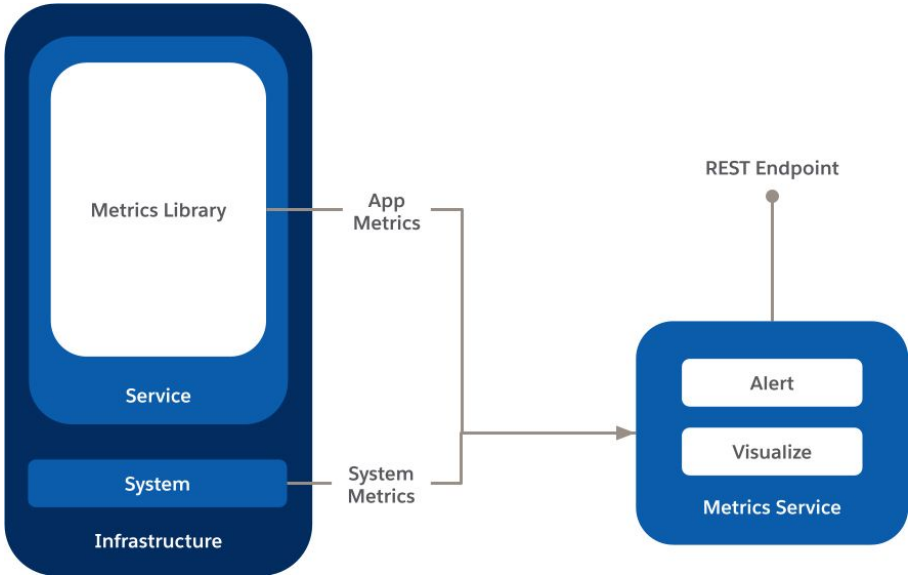
- Permitir consultar el estado del servicio.
- Permitir consultar el estado de la infraestructura.
- Implementación con Spring Boot Actuator.





Application Metrics

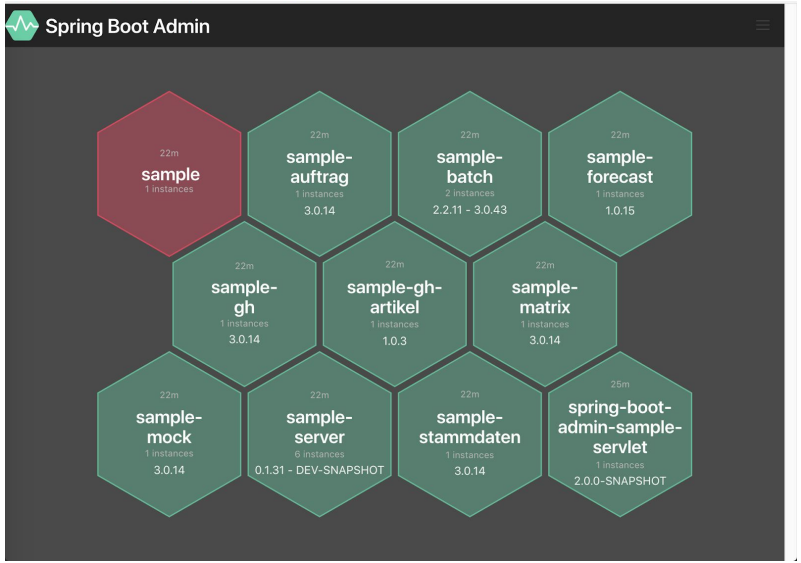
- Recopilar métricas agregadas de manera centralizada.
- Informes y alertas.
- Implementación con **Spring Boot Actuator** y **Spring Boot Admin**, Prometheus, Grafana, ELK.

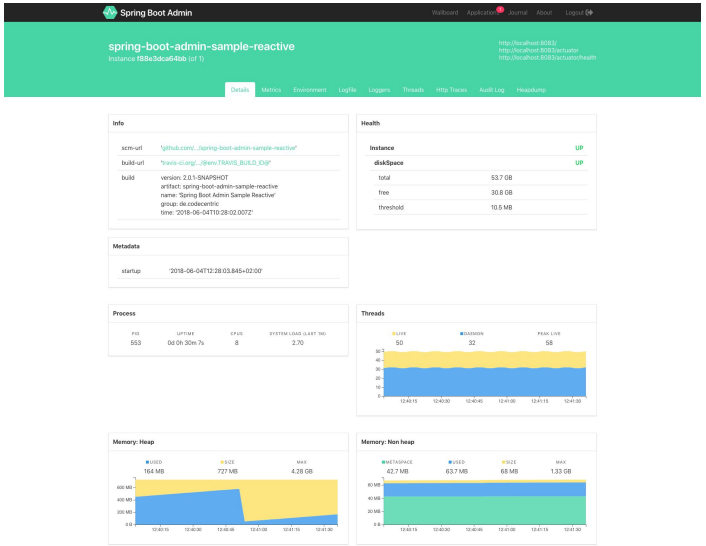


Spring Boot Admin

Es una aplicación web que permite gestionar y monitorear servicios de Spring Boot. Cada servicio es un cliente que se registra a este servidor (SBA).

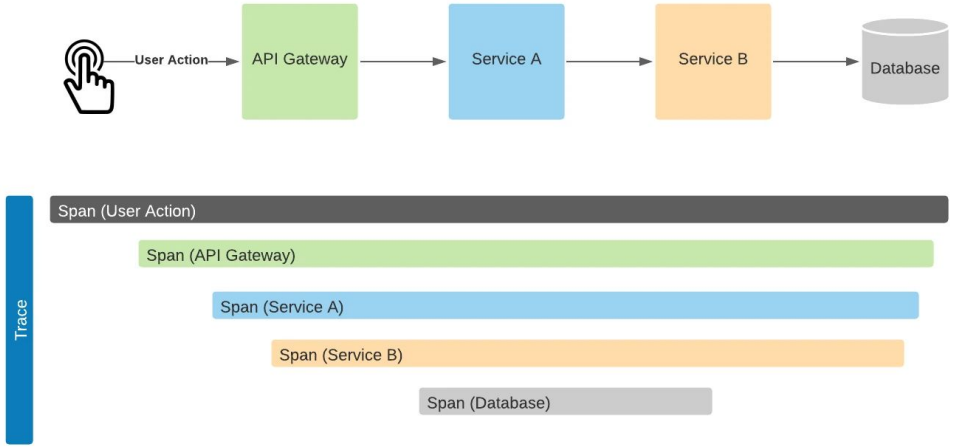
SBA hace uso de los endpoints de Spring Boot Actuator para obtener las métricas.





Distributed Tracing

- Identificar cada solicitud con un identificador único.
- Transferir el identificador a todos los servicios involucrados en la solicitud.
- Registrar los tiempos de cada solicitud en cada servicio involucrado.
- Centralizar la traza de cada solicitud.
- Implementación con **Spring Cloud Sleuth** y **Zipkin**, Jaeger, ELK, Opentelemetry, Opentracing.



Spring boot Sleuth

Es una solución de trazado distribuido para Spring Cloud.

Permite identificar la petición completa de cada microservicios como un todo en cada llamada individual.

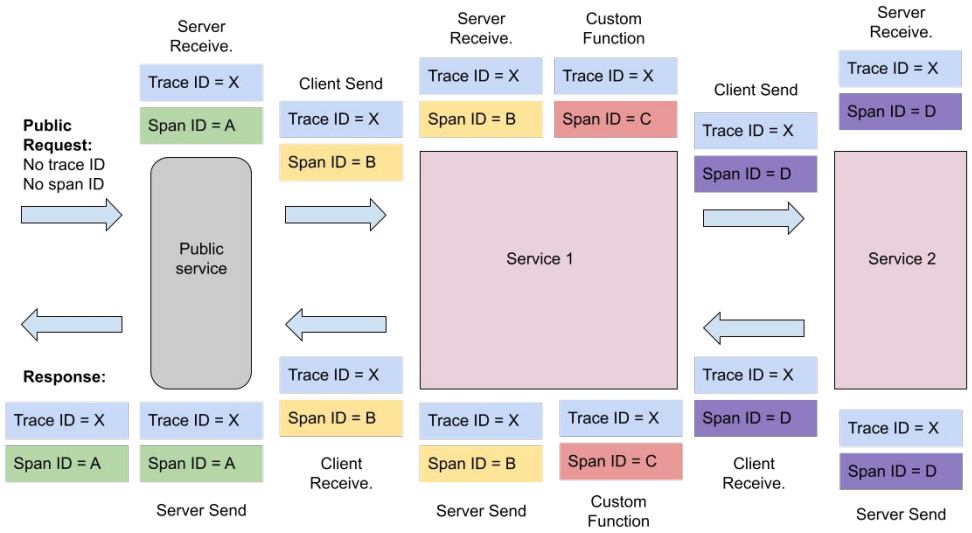
Identificadores:

TraceId: Identificador de toda la petición.

SpanId: Identificador de la petición en un solo microservicios.

Annotations: Permiten calcular los tiempos en cada petición: CS: Client Sent, SR: Server Received, SS: Server Sent, CR: Client Received.

Tags: Etiquetas para personalizar las peticiones.



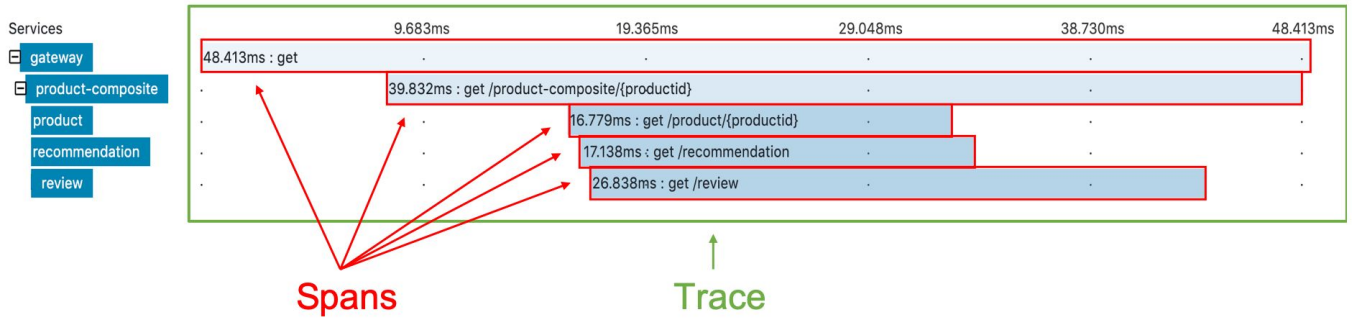
Spring boot Sleuth

Se agregar el TraceId y SpanId al Slf4j MDC:

```
2017-01-10 22:36:38.254  INFO
  [Microservicio,4e30f7340b3fb631,4e30f7340b3fb631,false] 12516
  --- [nio-8080-exec-1] c.b.spring.session.SleuthController : Hello Sleuth
```

Tiene la siguiente estructura:

```
[nombre de la aplicación, traceId, spanId, exportación]
```

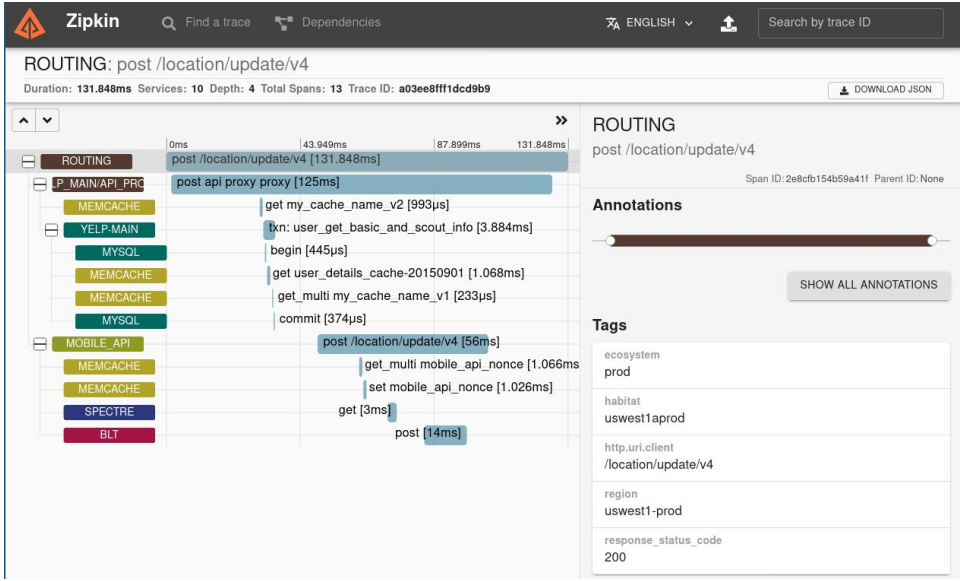


Zipkin

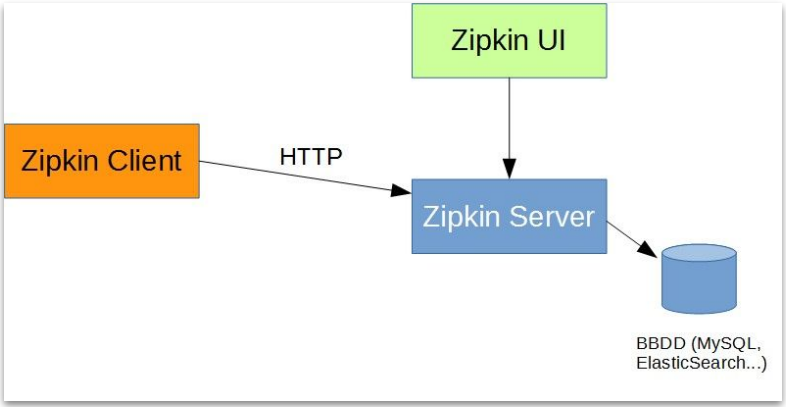
Permite recopilar los datos de la trazabilidad y de la temporización.

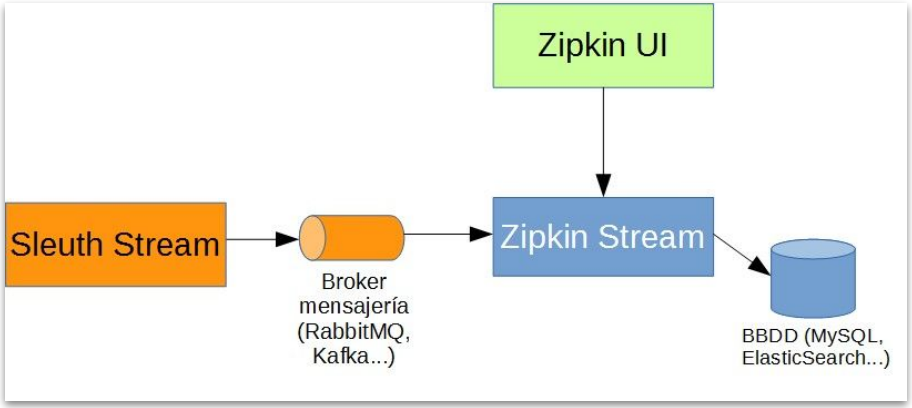
Aplicación web que permite realizar la búsqueda centralizada de estos datos almacenados.

Existen 2 formas de recopilar la información: Por **http** o **broker de mensajería**.

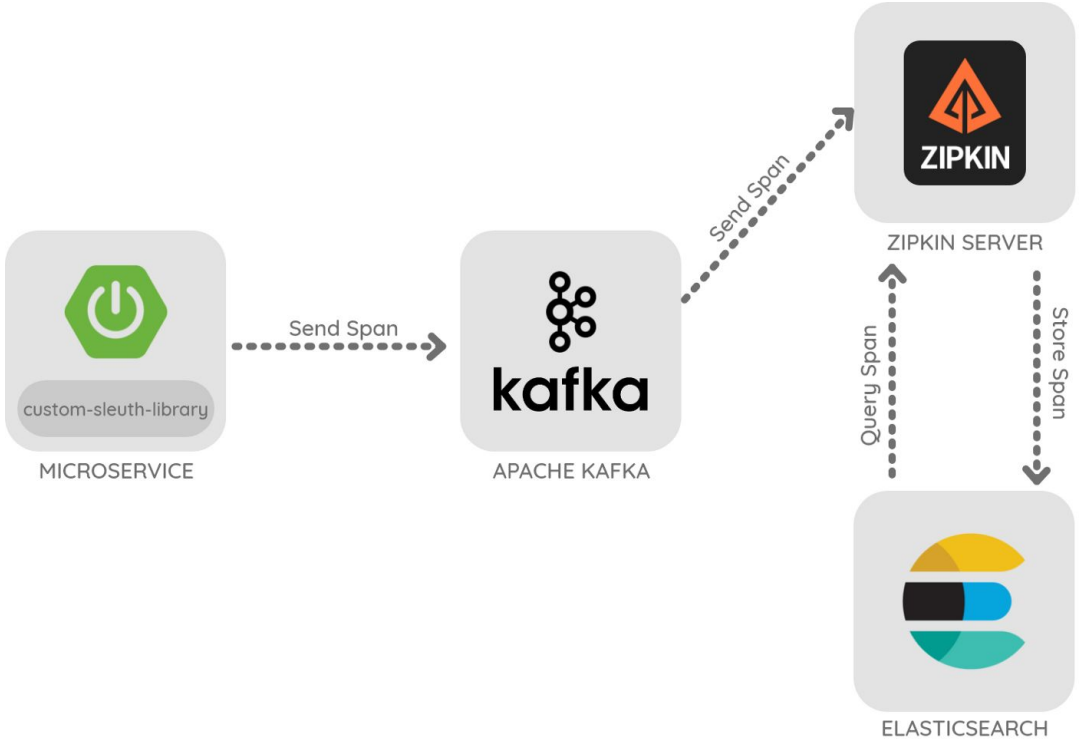


Zipkin


Zipkin UI
Zipkin Client
HTTP
Zipkin Server
BBDD (MySQL, ElasticSearch...)


Zipkin UI
Sleuth Stream
Broker mensajería (RabbitMQ, Kafka...)
Zipkin Stream
BBDD (MySQL, ElasticSearch...)

Distributed Tracing con Zipkin y Sleuth



Recursos de Arquitectura de Microservicios

https://spring.io/projects/spring-cloud-gateway
https://spring.io/projects/spring-cloud-circuitbreaker
https://www.docker.com
https://microservices.io/patterns/observability/application-metrics.html
https://github.com/codecentric/spring-boot-admin
https://microservices.io/patterns/observability/distributed-tracing.html
https://spring.io/projects/spring-cloud-sleuth
https://zipkin.io
https://microservices.io/patterns/observability/health-check-api.html

¡GRACIAS!

Correo: jean.ramal@gmail.com

Cel: 994641153

Link: https://www.linkedin.com/in/jeanramal/