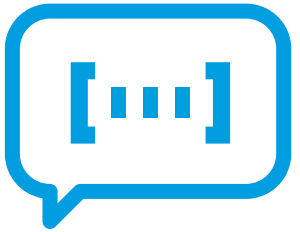
netmind
a bts company

Spring Boot

Implementacion microservicios con Spring cloud

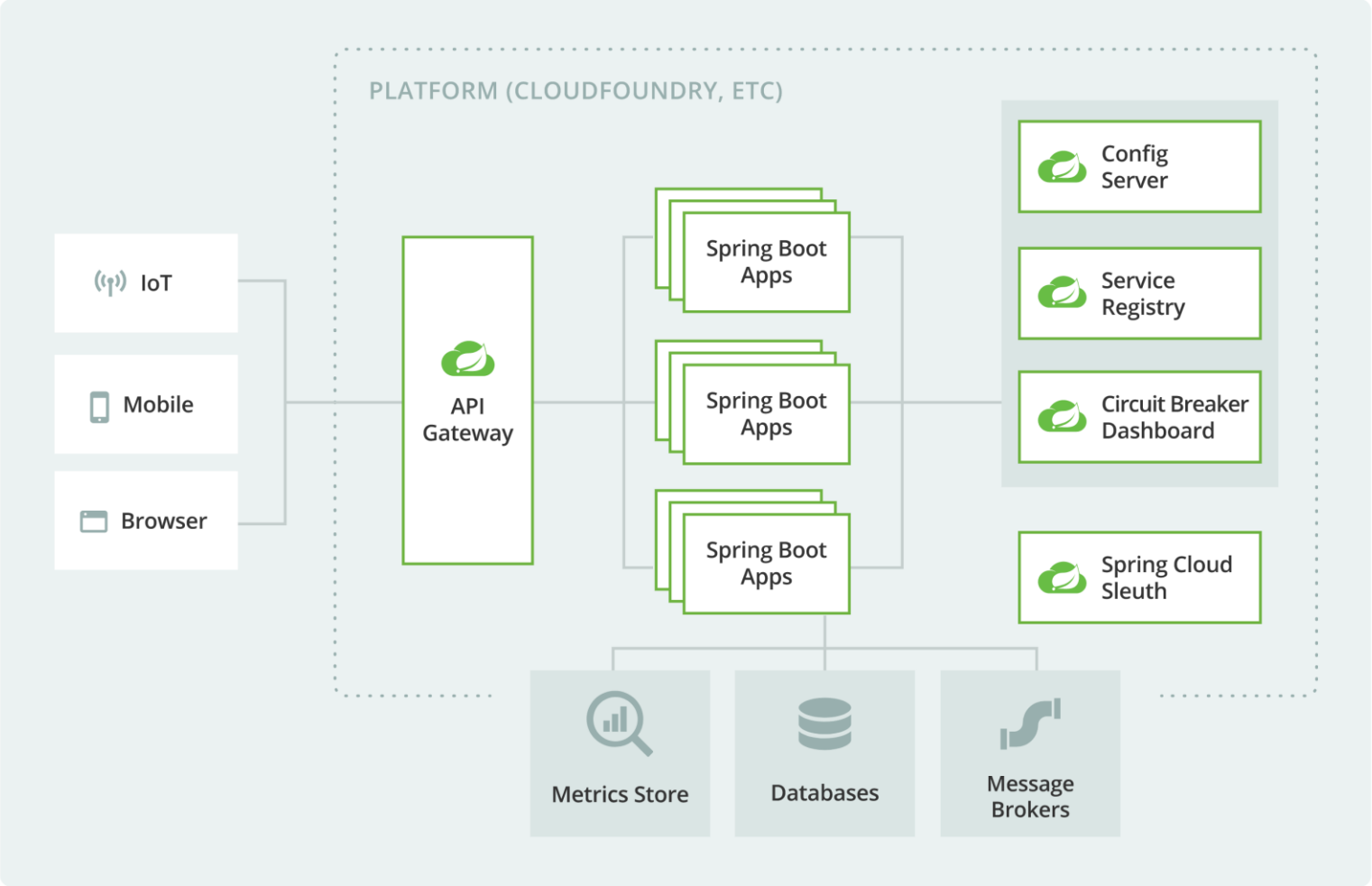
netmind
a bts company



Microservicios con Spring cloud

01

Ecosistema Spring Cloud



- **Componentes básicos** para la nube y microservicios.
- **Infraestructura de microservicios** como Service Discovery, un servidor de configuración y monitoreo.
- Varios otros **proyectos de código abierto** como Netflix OSS.
- **PaaS** como Cloud Foundry, AWS y Heroku.
- Utiliza **starters** estilo Spring Boot

Ecosistema Spring Cloud

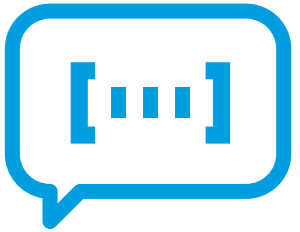
- **Service discovery:** Netflix Eureka (https://github.com/Netflix/eureka)
- **API gateway:** Spring Cloud Gateway (https://spring.io/projects/spring-cloud-gateway)
- **Cloud configuration:** Spring Cloud Config (https://spring.io/projects/spring-cloud-gateway)
- **Cloud load balancing:** Spring Cloud Load Balancer
- **Service client:** Spring Cloud Feign
- **Monitoring:** Spring Cloud Actuator
- **Circuit breakers:** Resilience4J, Sentinel, or Hystrix
- **Tracing:** Spring Cloud Sleuth, Zipkin
- **Testing:** Spring Cloud Contract

Otros intereresantes:

- **Clientes API:** Feign e Hystrix
- **API Gateway:** Netflix Zuul

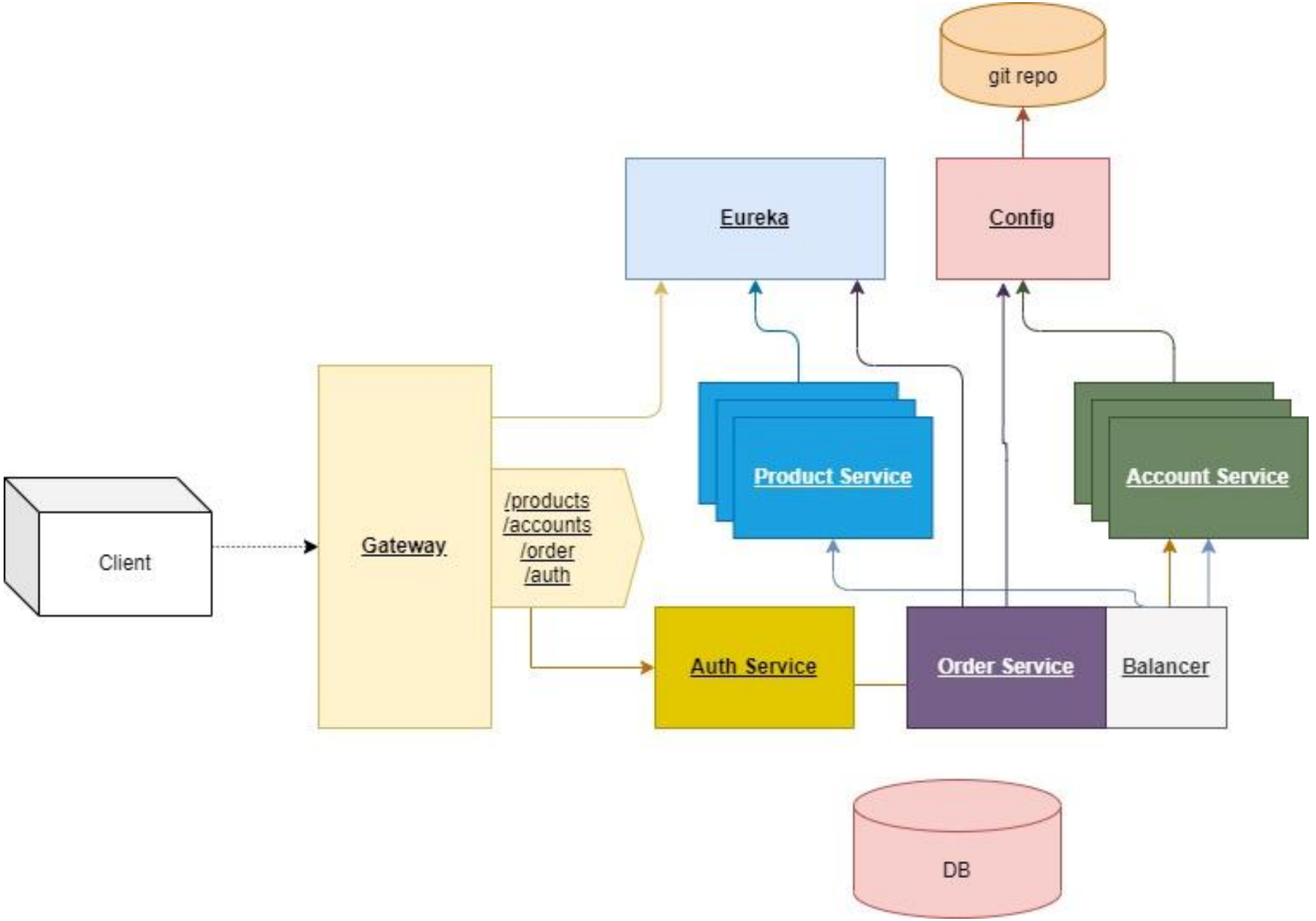
netmind
a bts company

02



Implementando un MSA sencillo

Implementando un sistema de microservicios sencillo



Config service

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-config-server</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

```
spring.application.name: spring-cloud-config-server
server.port: 8888
spring.cloud.config.server.git.default-label: master
spring.cloud.config.server.git.uri: “<path_to_git-localconfig-repo>"
```

```
@SpringBootApplication
@EnableConfigServer
public class SpringCloudConfigServerApplication {
        ...
}
```

El repo git debe tener archivos con nombre:

- `<service-name>.properties`
- `<service-name>-<profile>.properties`

Ver configuración:

- `http://localhost:8888/<service-name>/default`
- `http://localhost:8888/<service-name>/<profile>`

Cliente del Config service

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
```

```
spring.config.import: configserver:http://localhost:8888 # config server
spring.profiles.active: dev #for activate profile
management.endpoints.web.exposure.include: "*“ # for refresh
```

```
@Component
@ConfigurationProperties(“<service-name>")
@Getter @Setter
public class Configuration {
    // properties
}

...

@Autowired
private Configuration configuration;
```

🔗 **Get actuator env**

```
http://localhost:9100/actuator/env
```

🔗 **Refresh**

```
POST http://localhost:9100/actuator/refresh
```

Eureka

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

```
spring.application.name: netflix-eureka-naming-server
server.port: 8761

spring.config.import: configserver:http://localhost:8888
eureka.client.register-with-eureka: false
eureka.client.fetch-registry: false
```

```
@SpringBootApplication
@EnableEurekaServer
public class NetflixEurekaNamingServerApplication {
        ...
}
```

Eureka client

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

```
eureka.client.service-url.default-zone: http://localhost:8761/eureka
```

```
@SpringBootApplication
@EnableDiscoveryClient
public class CurrencyExchangeServiceApplication {
        ...
}
```

Feign user

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-feign</artifactId>
    <version>1.4.7.RELEASE</version>
</dependency>
```

```
@SpringBootApplication
@EnableFeignClients
@EnableDiscoveryClient
public class CurrencyConversionServiceApplication {
        ...
}
```

```
@FeignClient(name = "<service-to-consume-name>", url = “<service-to-consume-name>")
public interface ServiceToConsumeClient {
    @RequestMapping(method = RequestMethod.<METHOD>, value = “<service-to-consume-uri>")
    public DetinationBean getMethod(
            @PathVariable String any_path_variable
    );

}

...

@Autowired
private ServiceToConsumeClient serviceToConsumeClient;
```

Load balancer

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-loadbalancer</artifactId>
</dependency>
<dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

```
@SpringBootApplication
@EnableFeignClients
@EnableDiscoveryClient
public class CurrencyConversionServiceApplication {
         ...
}
```

```
@FeignClient(name = "<service-to-consume-name>")
@LoadBalancerClient(name = "<service-to-consume-name>", configuration= LoadBalancerConfiguration.class)
public interface ServiceToConsumeClient {
         ...
}
```

Load balancer

```
@Configuration
public class LoadBalancerConfiguration {

    @Bean
    public ServiceInstanceListSupplier
    discoveryClientServiceInstanceListSupplier(ConfigurableApplicationContext context) {
        System.out.println("Configuring Load balancer to prefer same instance");
        return ServiceInstanceListSupplier.builder()
                .withBlockingDiscoveryClient()
                .withSameInstancePreference()
                .build(context);
    }

}
```

🔗 **Lanzar un Segundo servicio**

```
mvn clean install
cd target
java -jar <app>.jar --server.port=<port>
```

API Gateway

```
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

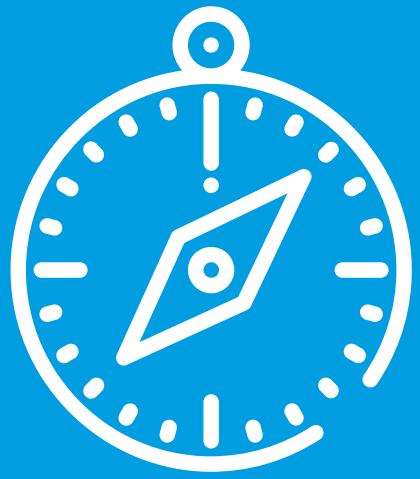
```
@SpringBootApplication
@EnableDiscoveryClient
public class GatewayServerApplication {
        ...
}
```

API Gateway

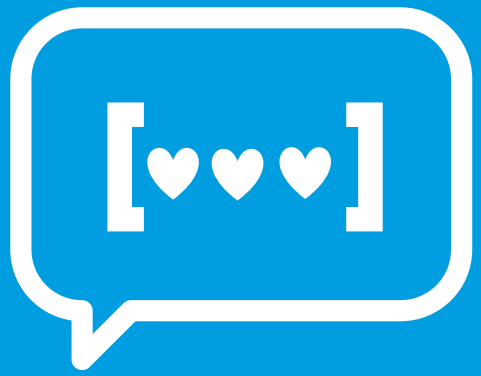
```
spring:
  application.name: gateway-server
  cloud:
    gateway:
      discovery.locator.enabled: true
      routes:
        - id: LocalIdentifier1
          uri: "lb://<service-name-1>/path/"
          predicates:
            - Path=/<gtw_path_1>/**
        - id: LocalIdentifier2
          uri: "lb://<service-name-2>/path/"
          predicates:
            - Path=/<gtw_path_2>/**
eureka.client:
  service-url.default-zone: http://localhost:8761/eureka
  health-check.enabled: true

management:
  endpoint:
    gateway:
      enabled: true
  endpoints:
    web:
      exposure:
        include: gateway
```

API Gateway

```
@Configuration
public class SpringCloudConfig {

    @Bean
    public RouteLocator gatewayRoutes(RouteLocatorBuilder builder) {
        return builder.routes()
                .route(r -> r.path("/<gtw_path_1>/**")
                        .uri("lb://<service-name-1>/path/"))

                .route(r -> r.path("/<gtw_path_2>/**")
                        .uri("lb://<service-name-2>/path/"))

                .build();
    }

}
```



Next steps



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: