Arquitetura de MicroServices com Spring Cloud e Spring Boot — Parte 1

João Rafael Campos da Silva Follow
Jul 2, 2017 · 4 min read

Sumário

Introdução

Fala pessoal, estou começando hoje uma série de Stories onde vamos abordar a construção de uma Arquitetura de MicroServices utilizando os frameworks Spring Cloud e Spring Boot da Pivotal.

Antes de começar decidi fazer uma breve introdução sobre as diferenças entre a Arquitetura de MicroServices e a Monolítica assim como reiterar alguns conceitos que surgem quando se utiliza a mesma.

Serão stories semanais e o sumário estará presente em todas para facilitar a navegação entre os tópicos. Então chega de enrolação e vamos ao que interessa!

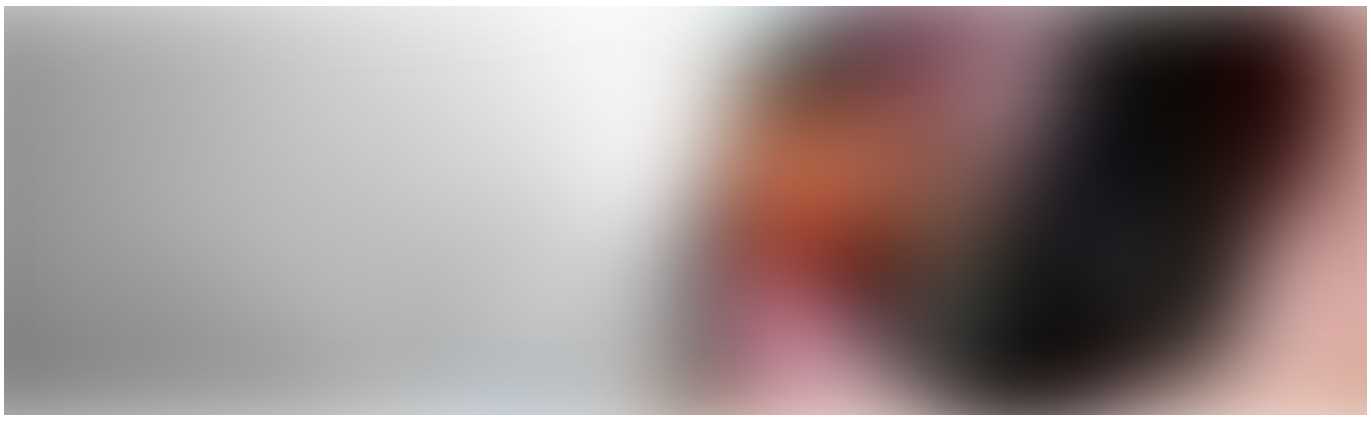
Servidor de Configuração

Como agora não teremos mais só uma aplicação, mas varias espalhadas por diversos servidores é necessário centralizar a configuração de todas as aplicações em um só lugar.

A Pivotal possui um projeto chamado Spring Cloud Config que possibilita a criação de uma configuração externalizada em um sistema distribuído. Com o Config Server você tem um lugar central, um repositório git por exemplo, para gerenciar os arquivos de configuração de cada aplicativo que se encontram rodando em outros ambientes.

As aplicações consultarão o Config Server para obter suas configurações na hora da inicialização. Podem ser desde configurações de acesso ao banco de dados até mesmo a porta em que desejamos que a aplicação suba.

Service Registry



Quando se tem muitas aplicações rodando em diferentes ambientes é difícil controlar qual o host ou porta onde cada um se encontra e se a mesma está online ou não.

O Service Registry é um banco de dados preenchido com informações sobre como enviar pedidos para instâncias de microservice. Cada instância que

sobe se cadastra no Service Registry informando que está Online para receber requisições.

Além de deter as informações de acesso as instâncias, o Service Registry também realizará checagens de saúde da aplicação e o balanceamento de carga de instâncias da mesma aplicação.

Para realizar essa função iremos utilizar o Eureka da Netflix.

Circuit Braker

Em uma aplicação distribuída é comum que haja chamadas entre diversos servidores na rede. Diferente da chamada em memória, as chamadas remotas podem falhar ou ficarem pendentes se o host destino estiver indisponível até que um tempo limite seja atingido.

Se vários clientes tentarem acessar esse mesmo recurso indisponível podemos ter uma falha crítica e isso pode afetar o funcionamento de todo o sistema.

O Circuit Break ficou popular por evitar esse tipo de cascata. Basicamente a request a um host que pode falhar é envolta em um Circuit Breaker e se essa chamada começar a falhar é retornado um erro conhecido e registrada alguma métrica informando que a request está falhando naquele local.

Tendo métricas em mãos dos locais exatos onde a aplicação está falhando, se torna menos complicada a busca pela resolução de erros ou gargalos na aplicação.

Para essa série de stories iremos utilizar o Hystrix da Netflix para tratar o Circuit Braker.

Gateway

Imagine que cada instância de um microservice sobe com uma porta diferente, e se você não tem um ponto de entrada único para sua aplicação como um todo, isso pode se tornar um caos na hora fornecer os recursos para um cliente web ou mobile.

O gateway funciona como uma porta de entrada da sua aplicação, todo o trafego passa por ele antes de ser encaminhado para o microservice específico respeitando as rotas que são configuradas no mesmo.

O Gateway geralmente recebe a requisição desejada pelo cliente e consulta no Service Registry qual instância de microservice responde por aquela rota. Se for uma rota segura, ele também irá realizar a autenticação junto ao servidor de autorização antes de fazer o redirecionamento.

Para essa série de stories iremos utilizar o Zuul da Netflix como Gateway.

Repositórios do Projeto

Configurações: https://github.com/rafaelcam/delivery-configs
Projeto: https://github.com/rafaelcam/delivery

Referências

https://auth0.com/blog/an-introduction-to-microservices-part-3-the-service-registry/
https://cloud.spring.io/spring-cloud-config/
https://github.com/Netflix/zuul
https://martinfowler.com/bliki/CircuitBreaker.html

Thanks to Diego Brener da Silva.

Spring Cloud Spring Boot Java Microservices Netflixoss