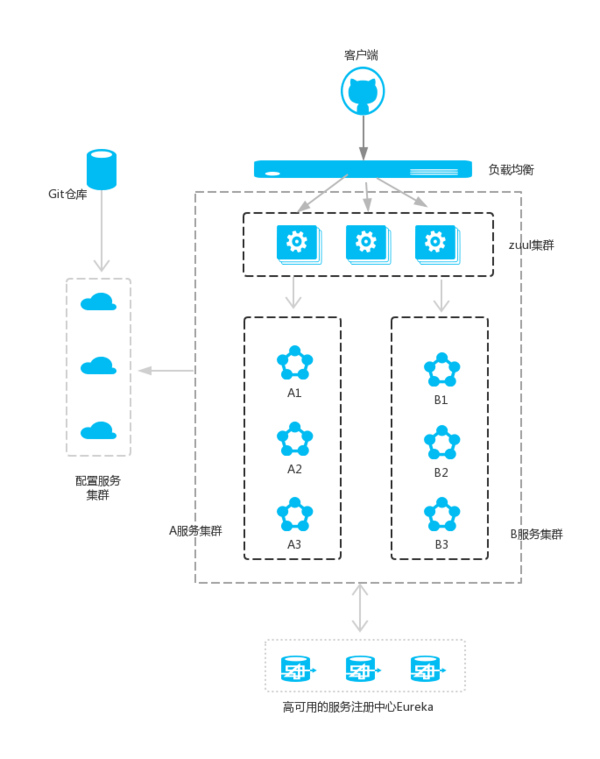
路由网关

2018年5月4日

16:55

在微服务架构中，需要几个基础的服务治理组件，包括服务注册与发现、服务消费、负载均衡、断路器、智能路由、配置管理等，由这几个基础组件相互协作，共同组建了一个简单的微服务系统。一个简单的微服务系统如下图：



**注意：A服务和B服务是可以相互调用的。并且配置服务也是注册到服务注册中心的。**

在Spring Cloud微服务系统中，一种常见的负载均衡方式是，客户端的请求首先经过负载均衡（zuul、Ngnix），再到达服务网关（zuul集群），然后再到具体的服。服务统一注册到高可用的服务注册中心集群，服务的所有的配置文件由配置服务管理（下一篇文章讲述），配置服务的配置文件放在git仓库，方便开发人员随时改配置。

## 1.zuul简介

Zuul的主要功能是路由转发和过滤器。路由功能是微服务的一部分，比如／api/user转发到到user服务，/api/shop转发到到shop服务。zuul默认和Ribbon结合实现了负载均衡的功能。

zuul有以下功能：

* Authentication
* Insights
* Stress Testing
* Canary Testing
* Dynamic Routing
* Service Migration
* Load Shedding
* Security
* Static Response handling
* Active/Active traffic management

## 2.创建工程

New Module 
Dependencies 
Core 
Web 
Template Engines 
SQL 
NoSQL 
Integration 
Cloud Core 
Cloud Config 
Cloud Discovery 
oud Routing 
Cloud Circuit Breaker 
Cloud Tracing 
Cloud Messaging 
Cloud AWS 
Cloud Contract 
Pivotal Cloud Foundry 
Azure 
Spring Cloud GCP 
Ops 
Spring Boot 2.0.1 
Zuul 
J Gateway 
Ribbon 
Feign 
Zuul 
Intelligent and programmable routing with 
spring-cloud- netflix Zuul 
A Routing and Filtering 
Previous 
Selected Dependencies 
Cloud Routing 
zuul 
Cancel 
Help 

在其入口applicaton类加上注解@EnableZuulProxy，开启zuul的功能：

J@SpringBootApplication 
@EnableEurekaServer 
public class ServiceZuulApplication { 
public static void main (StringC] args) I 
SpringAppl ication. run(ServiceZuu1App1ication. class, 
args) , 

配置文件

leureka : 
client: 
serviceUr1 : 
defaul tZone : 
I server : 
port: 8769 
I spring: 
appl i cation: 
http://localhost : 8761/eureka/ 
name: service—zuu 
Izuul : 
routes: 
apl—a: 
path: 
serviceld: 
service-ri bbon 
api—b: 
path: 
sr.rvirr•ld: 

首先指定服务注册中心的地址为<http://localhost:8761/eureka/>，服务的端口为8769，服务名为service-zuul；以/api-a/ 开头的请求都转发给service-ribbon服务；以/api-b/开头的请求都转发给service-feign服务；

<http://localhost:8769/api-a/test?name=齐涛>

<http://localhost:8769/api-b/test?name=齐涛>

## 3.zuul的服务过滤

@Component 
public class MyFi1ter extends ZuulFi1ter 
private static Logger logger = LoggerFactory. getLogger(MyFil ter. class) 
@0verr i d 
public String filterType() 
return FilterConstants. PRE T JPE 
@0verride 
public int filter0rder() ( return 0; 
@0verride 
public boolean shouldFi1ter() { return true: 
@0verride 
public Object run() 
RequestContext context = RequestContext. getCurrentContext() 
context. getRequest ( ) • 
HttpServ1etRequest request = 
jogger. info (String. 
request. getMethod ( ) , request. getRequestLRL ( ) . toString 
Object accessToken = request. getParameter( s: "token") 
if (accessToken == null) { 
jogger. is empty") 
context. setSendZuu1Response (false) 
context. setResponseStatusCode (401) 
context. getResponse() . getWriter() . write( s: 
"token is empty") • 

filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体如下：

* pre：路由之前
* routing：路由之时
* post： 路由之后
* error：发送错误调用

filterOrder：过滤的顺序

shouldFilter：这里可以写逻辑判断，是否要过滤，本文true,永远过滤。

run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。

访问http://localhost:8769/api-a/test?name=%E9%BD%90%E6%B6%9B时

Eureka 
token is empty 
localhost:8765/test?nat 
localhost:8769/api-a/tc x 

访问http://localhost:8769/api-a/test?name=%E9%BD%90%E6%B6%9B&token=111时

Eureka 
i am from 8763 
local x 
localhost:8769/api-a/tc x 
=111 