# SpringCloud- Eureka服务注册与发现

## 1.概述

springcloud是一个非常优秀的微服务框架，要管理众多的服务，就需要对这些服务进行治理，管理每个服务与每个服务之间的依赖关系，可以实现服务调用、负载均衡、服务容错、以及服务的注册与发现。  
​ 如果微服务之间存在调用依赖，就需要得到目标服务的服务地址，也就是微服务治理的服务发现。要完成服务发现，就需要将服务信息存储到某个载体，载体本身即是微服务治理的服务注册中心，而存储到载体的动作即是**服务注册。**

## 2.使用注册中心前服务间的调用

**使用RestTemplate**

服务消费方创建一个RestTemplate，通过http请求服务提供方暴露的接口形式

简单代码

消费方：

//bean

@Configuration

**public** **class** ConfigBean {

@Bean

**public** RestTemplate getRestTemplate() {

**return** **new** RestTemplate();

}

}

//controller

@Autowired

private RestTemplate restTemplate;

@RequestMapping(value = "/consumer/dept/add")

public boolean add(Dept dept)

{

return restTemplate.postForObject("http://localhost:8080" + "/dept/add", dept, Boolean.class);

}

有什么缺陷？

1. 路径硬编码，如果服务提供方端口或地址变更，不方便切换；
2. 如果服务提供方是多实例集群模式一个路径如何访问集群的api；(加nginx（或硬件f5），那是不是每个集群都要加呢？这样会造成nginx泛滥。)

3.难以实现如实健康检查，负载均衡调用、高可用性等等。

## 3.Eureka服务注册与发现

Eureka是Spring Cloud微服务套件中的一部分，可以与Springboot构建的微服务很容易的整合起来。  
Eureka包含了**服务器端**和**客户端**组件。

### 3.1 Eureka Server

Eureka Server 主要对外提供了三个功能：

1.服务注册，所有的服务都注册到 Eureka Server 上面来。

2.提供注册表，注册表就是所有注册上来服务的一个列表，Eureka 内部通过一个二层缓存机制来维护这个注册表。Eureka Client 在调用服务时，需要获取这个注册表，一般来说，这个注册表会缓存下来，如果缓存失效，则直接获取最新的注册表。

3.同步状态，Eureka Client 通过注册、心跳等机制，和 Eureka Server 同步当前客户端的状态，以便 Eureka Client 能够及时感知到变化。

### 3.2 Eureka Client

1.服务注册，将自己注册到服务注册中心(Eureka Server)

2.获取注册表信息

Eureka Client 从 Eureka Server 上获取服务的注册信息，「并将其缓存在本地」，这句是关键。

当 Eureka Client 在需要调用远程服务时，会从该信息中查找远程服务所对应的 IP 地址、端口等信息。Eureka Client 上缓存的服务注册信息会定期更新(30 秒)，如果 Eureka Server 返回的注册表信息与本地缓存的注册表信息不同的话，Eureka Client 会自动处理。

1. 服务续约

Eureka CLient 每隔 30 秒就要向 Eureka Server 发送一条心跳消息，来告诉 Eureka Server 我还在运行。

如果 Eureka Server 连续 90 秒都有没有收到 Eureka Client 的续约消息(连续三次没发送)，它会认为 Eureka Client 已经掉线了，会将掉线的 Eureka Client 从当前的服务注册列表中剔除。

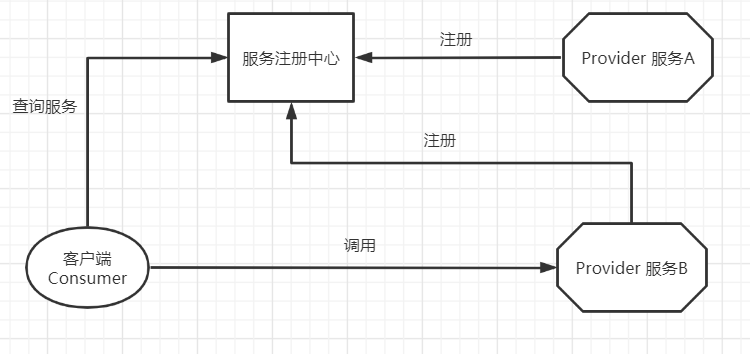
1. 服务下线。当 Eureka Client 下线时，它会主动发送一条消息，告诉 Eureka Server ，我下线啦。

从上面的介绍可以看出，Eureka Client 会自动拉取、更新以及缓存 Eureka Server 中的信息，这样，即使 Eureka Server 所有节点都宕机，Eureka Client 依然能够获取到想要调用服务的地址(前提是服务地址没有发生变化)

### **3.3服务注册与发现原理图**：

服务注册 -存储服务的主机及端口信息

服务发现-允许其他用户发现注册阶段存储的信息



概况：Provider 和 Consumer 分别作为服务启动，并且注册到 Eureka 上面去，以 provider 为例，provider 注册时会告诉 eureka，我叫 provider，我的地址是 xx.xx.xx.xx，我的端口是 xx，我的 xx 是 xx，就是说，provider 会将自己的一些元数据信息告诉 eureka；同理，consumer 也是如此。

接下来，consumer 要调用 provider 的接口，但是它不知道 provider 的地址是什么，他只知道要调用的服务叫 provider，于是 consumer 找到 eureka，从 eureka 上查询出来 provider 的具体地址和端口，这个具体的地址和端口，可能是一个，也可能是多个(集群化部署)。

## 4.项目集成搭建

#### 4.1创建服务中心Eureka

创建子项目microservicecloud-eureka-7001

Pom 添加依赖

<!--eureka-server服务端 -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka-server</artifactId>

</dependency>

配置文件

application.yml

server:

port: 7001

eureka:

instance:

hostname: eureka7001.com #eureka服务端的实例名称

client:

register-with-eureka: **false** #false表示不向注册中心注册自己。

fetch-registry: **false** #false表示自己端就是注册中心，我的职责就是维护服务实例，并不需要去检索服务

service-url:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/ #设置与Eureka Server交互的地址查询服务和注册服务都需要依赖这个地址（单机）。

启动类：

@SpringBootApplication

@EnableEurekaServer // EurekaServer服务器端启动类,接受其它微服务注册进来

public class EurekaServer7001\_App {

public static void main(String[] args) {

SpringApplication.run(EurekaServer7001\_App.class, args);

}

}

启动测试：

<http://localhost:7001/>

#### 4.2将服务的注册进Eureka

创建提供服务的客户端，并向服务注册中心注册自己。

1.引两个包

如修改8001工程，pom.xml 添加这两个包：

<!-- 将微服务provider侧注册进eureka -->

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

2.修改8001 的yml，后面追加

eureka:

client: #客户端注册进eureka服务列表内

service-url:

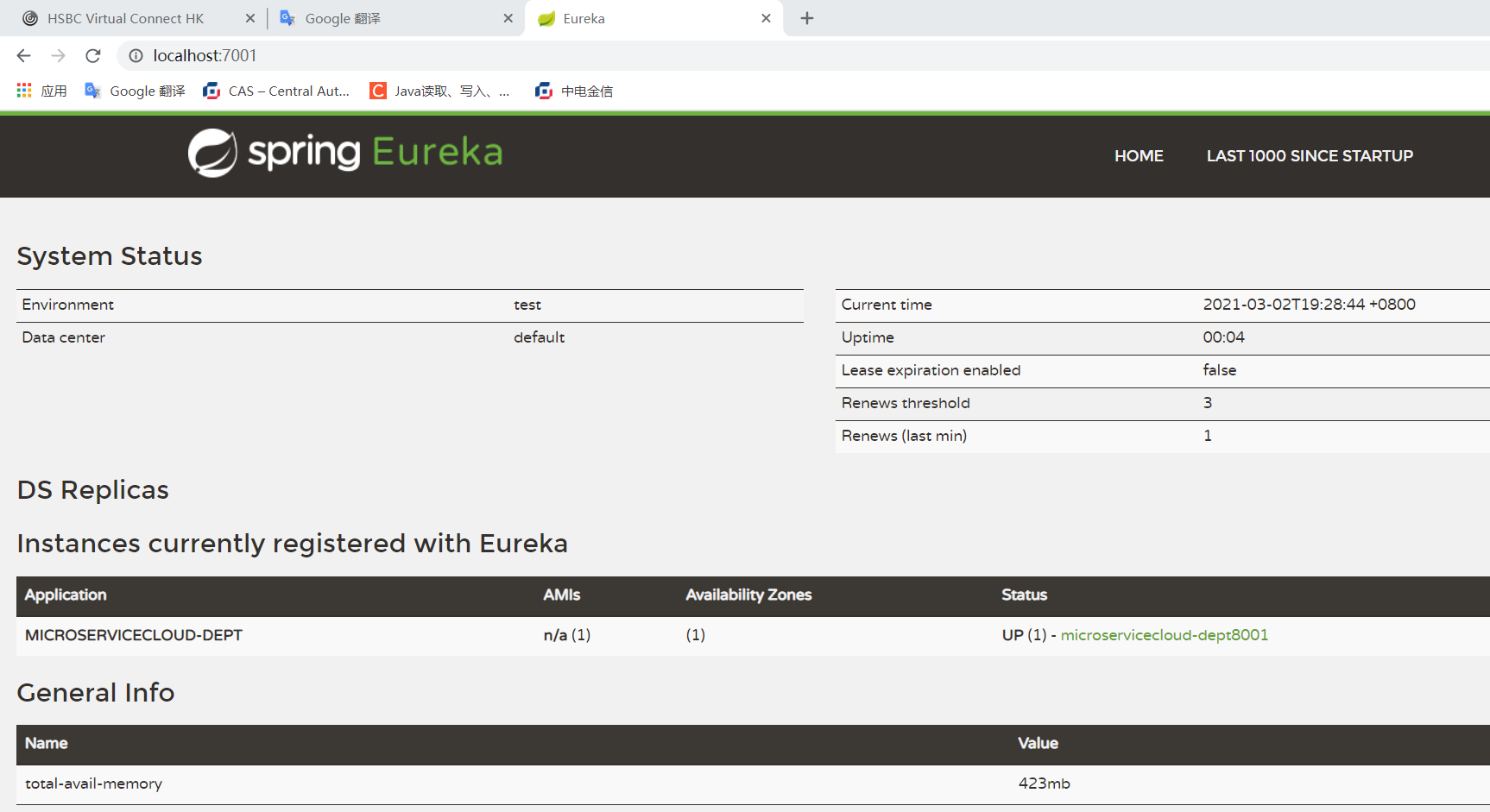
defaultZone: <http://localhost:7001/eureka>

3.修改8001启动类，添加注解

@EnableEurekaClient//表明是Eureka客户端

4.测试<http://localhost:7001/>

就可以看到服务名了。



#### 4.3 Eureka的高可用性

Eureka Server除了单点运行之外，还可以通过运行多个实例，并进行互相注册的方式来实现高可用的部署。

Eureka的集群配置

1.原理：把功能多部署几份应当高并发的请求提供服务。

2.搭建eureka-7002，eureka-7003，

复制eureka-7001的pom文件、启动类、yml文件，改相应的名称；

修改系统hosts文件映射：

添加

127.0.0.1 eureka7001.com

127.0.0.1 eureka7002.com

127.0.0.1 eureka7003.com

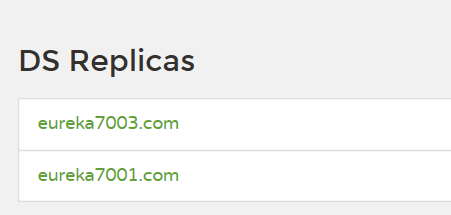
每个注册中心yml文件，两两注册，配置集群：

defaultZone: http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/

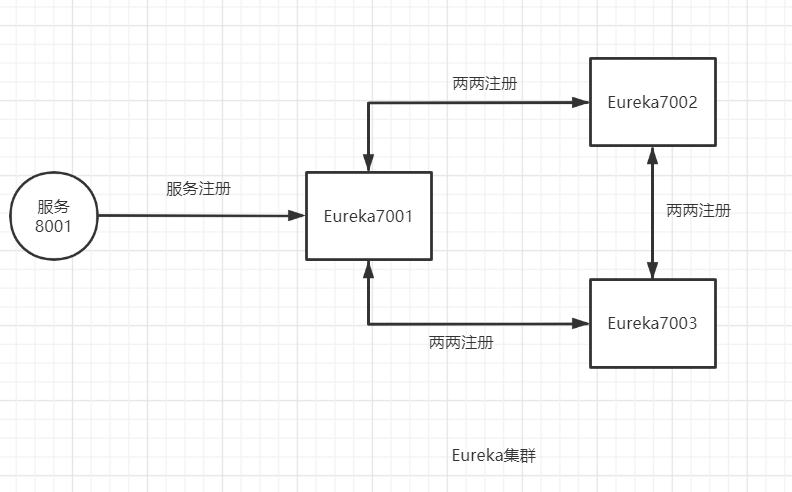
相应的客户端8001也修改yml文件

defaultZone: <http://eureka7001.com:7001/eureka/,http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/>

可以看到集群信息：



两两注册的方式可以实现集群中节点完全对等的效果，实现最高可用性集群，任何一台注册中心故障都不会影响服务的注册与发现



#### 4.4 Eureka自我保护

一句话：某时刻某一个微服务不可用了（如8001），Eureka-server不会立刻清理，依旧会对该微服务的消息进行保存。

默认情况下，如果Eureka-Server在一定时间内没有接收到某个微服务实例的心跳，如果 Eureka Server 连续 90 秒都有没有收到 Eureka Client 的续约消息(连续三次没发送)，它会认为 Eureka Client 已经掉线了，会将掉线的 Eureka Client 从当前的服务注册列表中剔除。

但是当网络分区故障发生时，微服务于EurekaServer之间无法正常通信，以上行为可能变得非常危险了—因为微服务本身其实是健康的，此时本不应该注销这个微服务。Eureka通过“自我保护模式”来解决这个问题—当Eureka Server节点短时间内丢失过多客户端时（可能发送了网络分区故障），那么这个节点就会进入自我保护模式。一旦进入该模式，EurekaServer就会保护服务注册表中的消息，不再删除服务注册表的数据（也就是不会注销任何微服务）。当网络故障恢复后，该EurekaServer节点会自动退出自我保护模式。

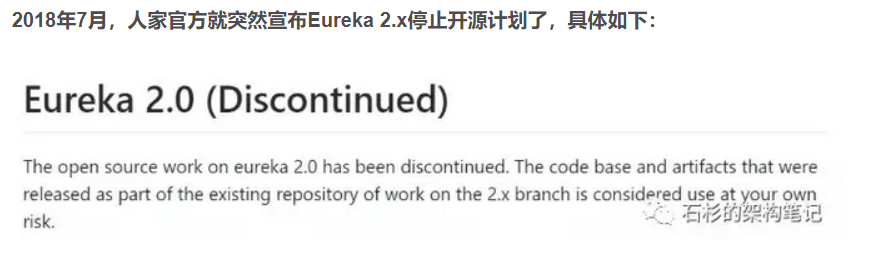
7001 yml文件：

eureka:  
 server:  
 *#关闭自我保护false（缺省true）,一般不建议这么做。* enable-self-preservation: false  
 *#续期时间，即扫描失效服务的间隔时间（缺省为90\*1000ms）* eviction-interval-timer-in-ms: 10000

## 与其他注册中心PK

后续：

Eureka闭源,目前2.0不更新



替代方案：  
Nacos:阿里开源2019.6正式1.0企业级,致力于帮助您发现、配置和管理微服务.  
Zookeeper：分布式协调工具，可以实现注册中心功能  
consul：简化了分布式环境中的服务的注册和发现流程，通过 HTTP 或者 DNS 接口发现。

这四个组件虽然都实现了注册中心的功能，但是他们的功能和实现方式都有不同的地方，也各有各的优点,单从注册中心方面来比价四个注册中心。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | Eureka | Nacos | Consul | ZK |
| **CAP定理(协议)** | AP | AP(或CP) | CP | CP |
| 负载均衡策略 | Ribbon | 权重/metadata/Selecto | Fabio | - |
| 雪崩保护 | 有 | 有 | 无 | 无 |
| SpringCloud集成 | 支持 | 支持 | 支持 | - |
| 多数据中心 | 支持 | 支持 | 支持 | - |

CAP原则又称CAP定理，指的是在一个分布式系统中，一致性（Consistency）、可用性（Availability）、分区容错性（Partition tolerance）。CAP 原则指的是，这三个要素最多只能同时实现两点，不可能三者兼顾。

CAP原则的精髓就是要么AP，要么CP，要么AC，但是不存在CAP。

· Consistency 一致性:所有数据备份，在同一时刻是否同样的值。（等同于所有节点访问同一份最新的数据副本）

· Availability 可用性:在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（对数据更新具备高可用性）

· Partition Tolerance 容错性:以实际效果而言，分区相当于对通信的时限要求。系统如果不能在时限内达成数据一致性，就意味着发生了分区的情况，必须就当前操作在C和A之间做出选择。

## 思考问题

# 如果Eureka server都宕机了，微服务还能调通吗（微服务消费方调微服务提供方）？为什么？

## 7.参考文献

*[Spring Cloud Eureka配置文件详解](https://www.cnblogs.com/sharpest/p/13678513.html)*

*<https://www.cnblogs.com/sharpest/p/13678513.html>*

*注册中心的比较*

*<https://www.jianshu.com/p/9b8a746e0d90>*

*[SpringCloud之服务注册与发现Eureka](https://www.cnblogs.com/grasp/p/9258811.html)*

*<https://www.cnblogs.com/grasp/p/9258811.html>*

*Spring Cloud构建微服务架构：服务注册与发现（Eureka、Consul）【Dalston版】*

*<https://blog.didispace.com/spring-cloud-starter-dalston-1/>*

*Eureka 都挂了，微服务还能调通吗？*

*<https://blog.csdn.net/weixin_26757939/article/details/112497977>*

*画图工具*

*<https://www.processon.com/diagrams>*