**Spring Cloud构建微服务架构（六）高可用服务注册中心**

**原创**

[2016-09-05](http://blog.didispace.com/springcloud6/)

 翟永超

[Spring Cloud](http://blog.didispace.com/categories/Spring-Cloud/)

被围观 **26464** 次

近期因工作原因减缓了更新频率，同时为了把[Spring Cloud中文社区](http://bbs.springcloud.com.cn/)搭建起来也费了不少时间，几乎每天都在挤牙膏般的凑时间出来做一些有意义的事。未能按原计划更新博文，在此对持续关注我博客的朋友们深表歉意。

之前在写Spring Cloud系列文章的时候，列过一个较粗的计划，现在由于收到不少反馈和问题，因此准备做一些调整，先将一些大家关注较为集中的点拉出来写一些内容。

今天这篇主要就说说Eureka Server的高可用问题。

**前言**

在[Spring Cloud](http://blog.didispace.com/tag/spring-cloud/)系列文章的开始，我们就介绍了[服务注册与发现](http://blog.didispace.com/springcloud1/)，其中，主要演示了如何构建和启动服务注册中心Eureka Server，以及如何将服务注册到Eureka Server中，但是在之前的示例中，这个服务注册中心是单点的，显然这并不适合应用于线上生产环境，那么下面在前文的基础上，我们来看看该如何构建高可用的Eureka Server集群。

单点Eureka Server的样例：

* [GitHub](https://github.com/dyc87112/SpringCloud-Learning/tree/master/Chapter1-1-1)
* [开源中国](http://git.oschina.net/didispace/SpringCloud-Learning/tree/master/Chapter1-1-1)

**Eureka Server的高可用**

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，所以我们只需要将Eureke Server配置其他可用的serviceUrl就能实现高可用部署。

下面以[Chapter1-1-1](https://github.com/dyc87112/SpringCloud-Learning/tree/master/Chapter1-1-1)中的eureka-server为基础，对其改造，构建双节点的服务注册中心。

* 创建application-peer1.properties，作为peer1服务中心的配置，并将serviceUrl指向peer2

|  |  |
| --- | --- |
| 1  2  3  4  5 | spring.application.name=eureka-server  server.port=1111  eureka.instance.hostname=peer1  eureka.client.serviceUrl.defaultZone=http://peer2:1112/eureka/ |

* 创建application-peer2.properties，作为peer2服务中心的配置，并将serviceUrl指向peer1

|  |  |
| --- | --- |
| 1  2  3  4  5 | spring.application.name=eureka-server  server.port=1112  eureka.instance.hostname=peer2  eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/ |

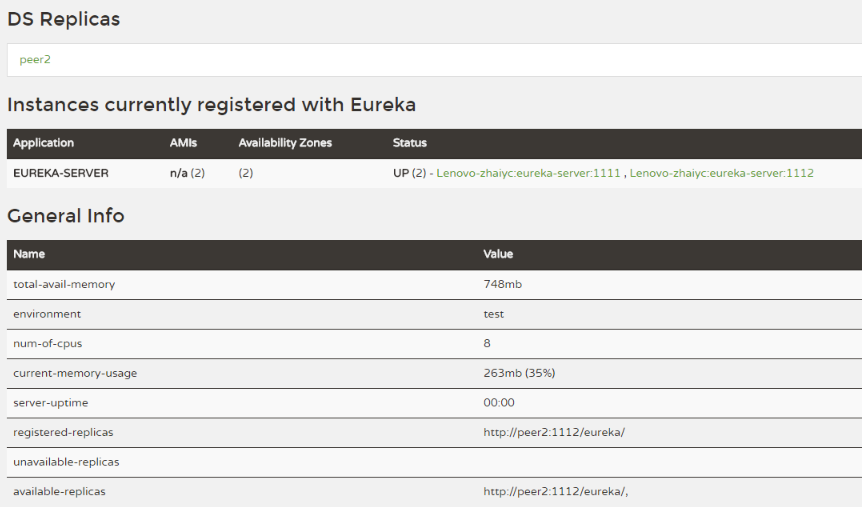
* 在/etc/hosts文件中添加对peer1和peer2的转换

|  |  |
| --- | --- |
| 1  2 | 127.0.0.1 peer1  127.0.0.1 peer2 |

* 通过spring.profiles.active属性来分别启动peer1和peer2

|  |  |
| --- | --- |
| 1  2 | java -jar eureka-server-1.0.0.jar --spring.profiles.active=peer1  java -jar eureka-server-1.0.0.jar --spring.profiles.active=peer2 |

* 此时访问peer1的注册中心：http://localhost:1111/，如下图所示，我们可以看到registered-replicas中已经有peer2节点的eureka-server了。同样地，访问peer2的注册中心：http://localhost:1112/，能看到registered-replicas中已经有peer1节点，并且这些节点在可用分片（available-replicase）之中。我们也可以尝试关闭peer1，刷新http://localhost:1112/，可以看到peer1的节点变为了不可用分片（unavailable-replicas）。

[](http://blog.didispace.com/content/images/2016/09/QQ--20160906132517.png)

**服务注册与发现**

在设置了多节点的服务注册中心之后，我们主需要简单需求服务配置，就能将服务注册到Eureka Server集群中。我们以[Chapter1-1-1](https://github.com/dyc87112/SpringCloud-Learning/tree/master/Chapter1-1-1)中的compute-service为基础，修改application.properties配置文件：

|  |  |
| --- | --- |
| 1  2  3  4 | spring.application.name=compute-service  server.port=2222  eureka.client.serviceUrl.defaultZone=http://peer1:1111/eureka/,http://peer2:1112/eureka/ |

上面的配置主要对eureka.client.serviceUrl.defaultZone属性做了改动，将注册中心指向了之前我们搭建的peer1与peer2。

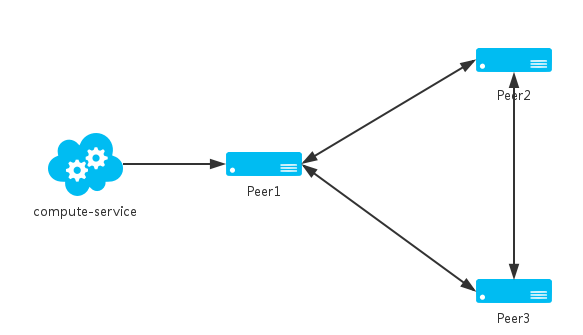
下面，我们启动该服务，通过访问http://localhost:1111/和http://localhost:1112/，可以观察到compute-service同时被注册到了peer1和peer2上。若此时断开peer1，由于compute-service同时也向peer2注册，因此在peer2上其他服务依然能访问到compute-service，从而实现了高可用的服务注册中心。

**深入理解**

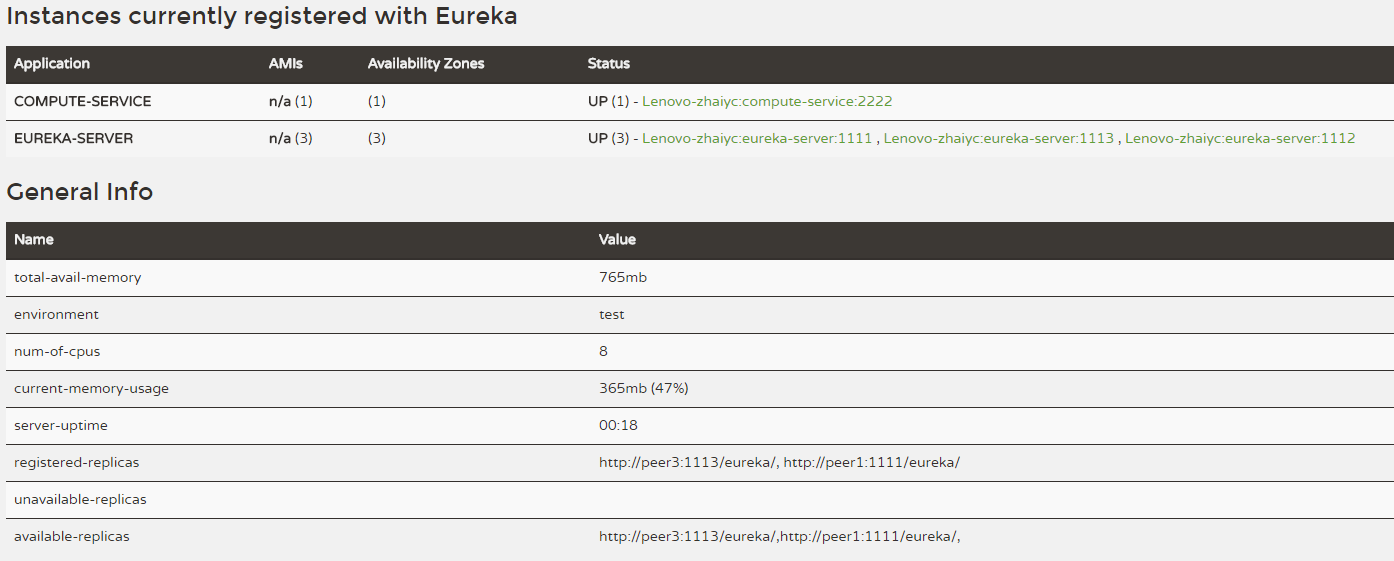
虽然上面我们以双节点作为例子，但是实际上因负载等原因，我们往往可能需要在生产环境构建多于两个的Eureka Server节点。那么对于如何配置serviceUrl来让集群中的服务进行同步，需要我们更深入的理解节点间的同步机制来做出决策。

Eureka Server的同步遵循着一个非常简单的原则：只要有一条边将节点连接，就可以进行信息传播与同步。什么意思呢？不妨我们通过下面的实验来看看会发生什么。

* 场景一：假设我们有3个注册中心，我们将peer1、peer2、peer3各自都将serviceUrl指向另外两个节点。换言之，peer1、peer2、peer3是两两互相注册的。启动三个服务注册中心，并将compute-service的serviceUrl指向peer1并启动，可以获得如下图所示的集群效果。

[](http://blog.didispace.com/content/images/2016/09/s1.png)

访问http://localhost:1112/，可以看到3个注册中心组成了集群，compute-service服务通过peer1同步给了与之互相注册的peer2和peer3。

[](http://blog.didispace.com/content/images/2016/09/QQ--20160906171357.png)

通过上面的实验，我们可以得出下面的结论来指导我们搭建服务注册中心的高可用集群：

* 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