微服务架构：Spring-Cloud

什么是微服务?

微服务就是把原本臃肿的一个项目的所有模块拆分开来并做到互相没有关联，甚至可以不使用同一个数据库。 比如：项目里面有User模块和Power模块，但是User模块和Power模块并没有直接关系，仅仅只是一些数据需要交互，那么就可以吧这2个模块单独分开来，当user需要调用power的时候，power是一个服务方，但是power需要调用user的时候，user又是服务方了， 所以，他们并不在乎谁是服务方谁是调用方，他们都是2个独立的服务，这时候，微服务的概念就出来了。

经典问题:微服务和分布式的区别

谈到区别，我们先简单说一下分布式是什么，所谓分布式，就是将偌大的系统划分为多个模块（这一点和微服务很像）部署到不同机器上（因为一台机器可能承受不了这么大的压力或者说一台非常好的服务器的成本可能够好几台普通的了），各个模块通过接口进行数据交互，其实 分布式也是一种微服务。 因为都是吧模块拆分开来变为独立的单元，提供接口来调用，那么 他们本质的区别在哪呢？ 他们的区别主要体现在“目标”上， 何为目标，就是你这样架构项目要做到的事情。 分布式的目标是什么？ 我们刚刚也看见了， 就是一台机器承受不了的，或者是成本问题， 不得不使用多台机器来完成服务的部署， 而微服务的目标 只是让各个模块拆分开来，不会被互相影响，比如模块的升级亦或是出现BUG等等...

讲了这么多，可以用一句话来理解：分布式也是微服务的一种，而微服务他可以是在一台机器上。

微服务与Spring-Cloud的关系（区别）

微服务只是一种项目的架构方式，或者说是一种概念，就如同我们的MVC架构一样， 那么Spring-Cloud便是对这种技术的实现。

微服务一定要使用Spring-Cloud吗?

我们刚刚说过，微服务只是一种项目的架构方式，如果你足够了解微服务是什么概念你就会知道，其实微服务就算不借助任何技术也能实现，只是有很多问题需要我们解决罢了例如：负载均衡，服务的注册与发现，服务调用，路由。。。。等等等等一系列问题，所以,Spring-Cloud 就出来了，Spring-Cloud将处理这些问题的的技术全部打包好了，就类似那种开袋即食的感觉。。

Spring-Cloud项目的搭建

因为spring-cloud是基于spring-boot项目来的，所以我们项目得是一个spring-boot项目，至于spring-boot项目，这节我们先不讨论，这里要注意的一个点是spring-cloud的版本与spring-boot的版本要对应下图：

Table 1. Release train Spring Boot compatibility

Release Train	Boot Version
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

Table 2. Release train content

在我这里我的版本是这样的

spring-boot：

```
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
    </parent>
```

spring-cloud:

```
  <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Finchley.SR2</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

当你项目里面有这些依赖之后，你的spring cloud项目已经搭建好了(初次下载spring-cloud可能需要一点时间)

Spring-Cloud组件：

eureka：

eureka是什么?

eureka是Netflix的子模块之一，也是一个核心的模块，eureka里有2个组件，一个是EurekaServer(一个独立的项目) 这个是用于定位服务以实现中间层服务器的负载平衡和故障转移，另一个便是EurekaClient（我们的微服务）它是用于与Server交互的，可以使得交互变得非常简单:只需要通过服务标识符即可拿到服务。

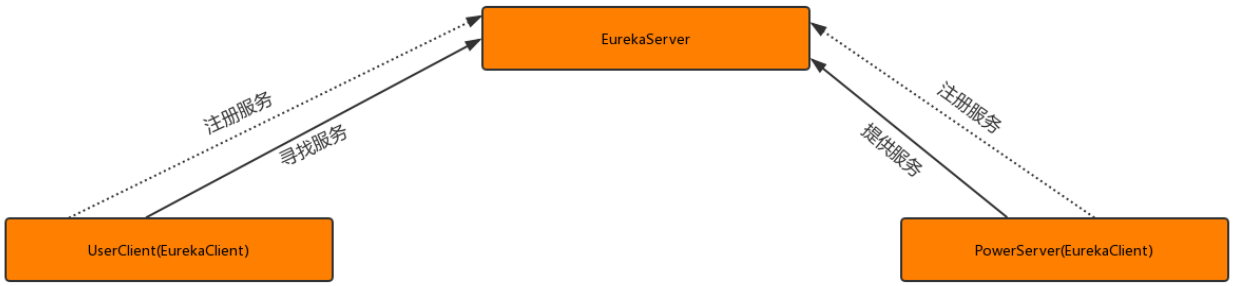
与spring-cloud的关系：

Spring Cloud 封装了 Netflix 公司开发的 Eureka 模块来实现服务注册和发现(可以对比Zookeeper)。

Eureka 采用了 C-S 的设计架构。Eureka Server 作为服务注册功能的服务器，它是服务注册中心。

而系统中的其他微服务，使用 Eureka 的客户端连接到 Eureka Server并维持心跳连接。这样系统的维护人员就可以通过 Eureka Server 来监控系统中各个微服务是否正常运行。SpringCloud 的一些其他模块（比如Zuul）就可以通过 Eureka Server 来发现系统中的其他微服务，并执行相关的逻辑。

角色关系图：



如何使用?

在spring-cloud项目里面加入依赖：

eureka客户端：

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

eureka服务端：

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

eureka服务端项目里面加入以下配置：

```
server:
  port: 3000
eureka:
  server:
    enable-self-preservation: false  #关闭自我保护机制
    eviction-interval-timer-in-ms: 4000 #设置清理间隔（单位：毫秒 默认是60*1000）
  instance:
    hostname: localhost


  client:
    registerWithEureka: false #不把自己作为一个客户端注册到自己身上
```

```
    fetchRegistry: false  #不需要从服务端获取注册信息（因为在这里自己就是服务端，而且已经禁用自己注册了）
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka
```

当然，不是全部必要的，这里只是把我这里的配置copy过来了

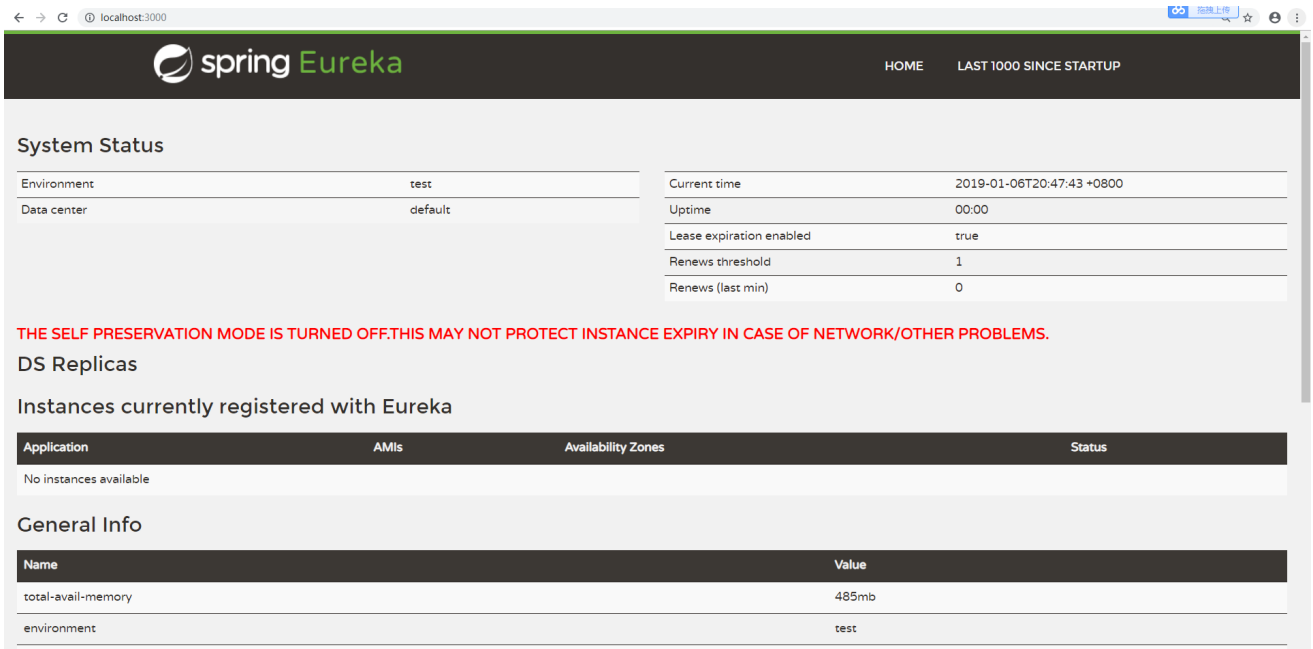
然后在spring-boot启动项目上 加入注解:@EnableEurekaServer 就可以启动项目了

```
/**
 * 想要咨询vip课程相关的同学加一下木兰老师QQ：2746251334
 * 想要往期视频的同学加一下安其拉老师QQ：3164703201
 * author：鲁班学院-商鞅老师
 */
@EnableEurekaServer
@SpringBootApplication
public class AppEureka {

    public static void main(String[] args) {
        SpringApplication.run(AppEureka.class);

    }
}
```

如果看见这个图片，那么说明你就搭建好了:



这个警告只是说你把他的自我保护机制关闭了

eureka客户端配置:

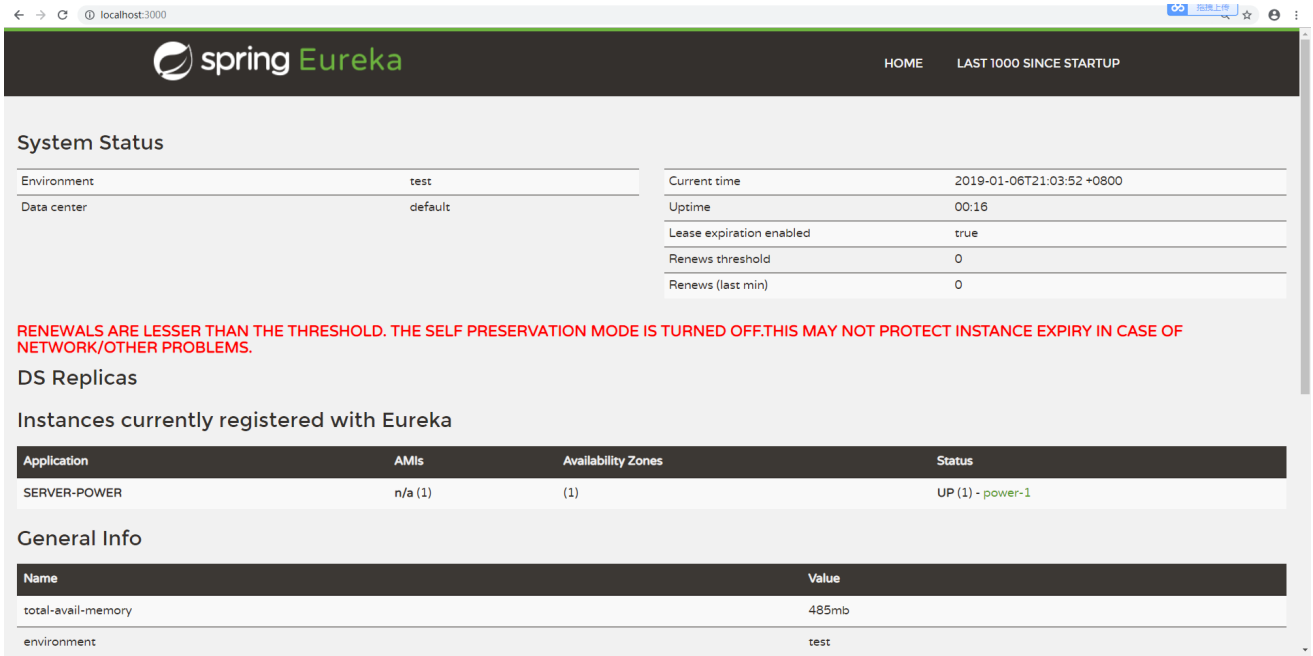
```
server:
  port: 6000
eureka:
```

```
  client:
    serviceUrl:
        defaultZone: http://localhost:3000/eureka/  #eureka服务端提供的注册地址 参考服务端配置的这个路径
    instance:

      instance-id: power-1 #此实例注册到eureka服务端的唯一的实例ID
      prefer-ip-address: true #是否显示IP地址
      leaseRenewalIntervalInSeconds: 10 #eureka客户需要多长时间发送心跳给eureka服务器，表明它仍然活着,默认为30 秒 (与下面配置的单位都是秒)
      leaseExpirationDurationInSeconds: 30 #Eureka服务器在接收到实例的最后一次发出的心跳后，需要等待多久才可以将此实例删除，默认为90秒

spring:
  application:
    name: server-power #此实例注册到eureka服务端的name
```

然后在客户端的spring-boot启动项目上 加入注解:@EnableEurekaClient 就可以启动项目了 这里就不截图了我们直接来看效果图:



这里我们能看见 名字叫server-power的（图中将其大写了） id为 power-1的服务 注册到我们的Eureka上面来了 至此，一个简单的eureka已经搭建好了。