# Spring Could的诞生

尽管微服务架构为复杂业务提供了很好的解决方案，但也给开发、测试、运维带来了非 常大的挑战。站在开发人员的角度、首先，需要考虑各个微服务之间的远程通信，市场 上有非常多的RPC框架，比如Thrift、Dubbo、Motan、gRPC等。其次，有了远程通信 之后，还需要考虑服务构成大规模集群之后如何做好服务的动态感知，比如A服务要 调用B服务，但是B服务部署了10个节点，那么A服务如何动态维护B服务的10个 不同的地址信息呢？这就需要采用一些第三方组件来管理目标服务的地址。

在微服务一体化解决方案出现之前，各个公司在实现服务化的过程中都比较痛苦，不仅 仅需要对各种开源技术进行横向对比及整合，还需要针对公司内部业务特征对这些开源 组件进行包装和优化，这是一个痛苦的过程，直到Spring Colud出现。Spring Could是 Spring团队研发的。

# 什么是Spring Could

Spring Could提供了一些可以让开发者快速构建微服务应用的工具，比如配置管理、服 务发现、熔断、智能路由等，这些服务可以在任何分布式环境下很好地工作。Spring Cloud 主要致力于解决如下问题：

·Distributed/versioned configuration，分布式版本化配置

·Service registration and discovery，服务注册与发现

·Routing，服务路由

·Service-to-service calls，服务调用

·Load balancing，负载均衡

·Circuit Breakers，断路器

·Global locks，全局锁

·Leadership election and cluster state，Leader选举及集群状态

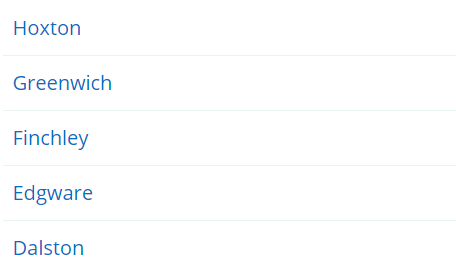
·Distrubuted messaging，分布式消息

需要注意的是，Spring Could并不是Spring团队全新研发的框架，它只是把一些比较优 秀的解决微服务架构中常见问题的开源框架基于Spring Could规范进行了整合，通过S pring Could规范进行了整合，通过Spring Boot这个框架进行再次封装后屏蔽掉了复杂 的配置，给开发者提供良好的开箱即用的微服务开发体验。不难看出Spring Could其实 就是一套规范，而Spring Cloud Netflix、Spring Cloud Consul、Spring Could Alibaba才是 Spring Cloud规范的实现。

# Spring Could版本介绍

1）Spring Could是一套整合了各大公司开源技术的规范，而这些开源技术的版本发布时 由各个公司来维护的，每个子项目都维护了自己的发布版本号，所以Spring Could 版本的命名不像传统意义上的版本命令，而是采用了伦敦地铁站的名字根据字母表

的顺序结合对应版本的时间顺序来定义一个大版本。Spring Could以往的版本发布顺 序排列如下：



Hoxton是目前最新的版本。

2）Spring Could的每一个大版本又分为很多小版本，通常在Spring Could大版本号后面

多了一个.SRX，X是一个递增的数字。这是因为Spring Could项目的发布内容积累到

一个临界点或者解决一些严重的Bug后，会发布一个Service Release的版本，简称 SRX，X是一个递增的数字。如下：



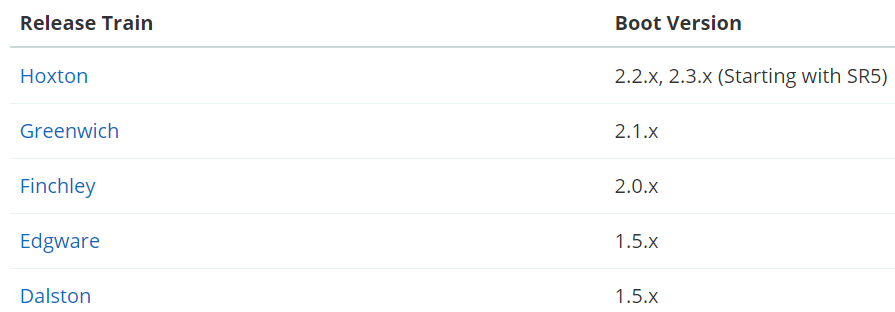
3）选择一个Spring Could版本，使用各个子项目时也需要选择对应子项目的版本。如

下图所示，如果引入的Spring Could版本是Hoxton.SR8，那么依赖的Spring Could Aws 版应该是2.2.4.RELEASE。



4）Spring Could中所有子项目都依赖Spring Boot框架，所以Spring Boot框架的版本号

和Spring Could的版本号之间也存在依赖及兼容关系。



# Spring Could规范下的实现

在Spring Could这个规范下，有很多实现，比如：



通过这些实现（它们都是别人造好的轮子），然后基于Spring Could规范进行整合，使 用者只需要使用非常简单的配置即可完成微服务架构下复制的需求。

# Spring Could解决方案

Spring Could生态下服务治理的解决方案主要有两个：Spring Could Netflix和Spring Could Alibaba，这两个解决方案分别是针对Netflix OSS及Alibaba的服务治理体系基于Spring Could规范做的整合。

# Spring Could Netflix

Spring Could Netflix主要为微服务架构下的服务治理提供解决方案，包括以下组件：

·Eureka，服务注册与发现

·Zuul，服务网管

·Ribbon，负载均衡

·Feign，远程服务的客户端代理

·Hystrix，断路器，提供服务熔断和限流功能

·Turbine，将各个服务实例上的Hystrix监控信息进行统一聚合

Spring Cloud Netflix是Spring Boot和Netflix OOS在Spring Cloud规范下的集成，其中， Netflix OOS（Netflix Open Source Softw）是由Netflix公司开发的一套开源框架和组件库， Eureka、Zuul等都是Netflix OOS中的开源组件。而Spring Cloud只是把这些组件进行了 整合，使得使用者可以更快速、更简单地构建微服务，以及解决微服务下的服务治理等 难题，但由于Netflix公司对Zuul、Ribbon等组件维护不利，Spring Cloud决定这些组件 改为维护模式（进入维护模式意味着这些组件以后不会有大的功能更新，只会修复Block 级别的Bug及安全问题）。当然，这些组件短期仍然可以继续使用，但是长期来看显然 是不合适的，但Spring官方提供了替换的建议，建议使用其他的组件来代替这些进入 了维护模式的组件。当然，除了Spring Cloud Netflix外，在服务治理的解决方案上，还 会有更多的选择，如Spring Cloud Alibaba。

# Spring Cloud Alibaba

## 认识Spring Cloud Alibaba

Spring Cloud Alibaba是阿里巴巴集团下的开源组件和云产品在Spring Cloud规范下的实 现。后入驻Spring Cloud官方孵化器，随后发布了第一个毕业版本。

Spring Cloud Alibaba主要为微服务开发提供一站式的解决方案，使开发者通过Spring Cloud编程模型轻松地解决微服务架构下的各类技术问题。以下是Spring Cloud Alibaba 生态下的主要功能组件，这些组件包含开源组件和阿里云产品组件（云产品组件需要付 费使用）：

·Sentinel，流量控制和服务降级

·Nacos，服务注册与发现

·Nacos，分布式配置中心

·RocketMQ，消息驱动

·State，分布式事务

·Dubbo，RPC通信

·OSS，阿里云对象存储（收费的云服务）

## Spring Cloud Alibaba的优势

相对于Spring Cloud Netflix来说，Spring Cloud Alibaba的优势有很多，如：

1. Alibaba的开源组件在没有织入Spring Cloud生态之前，已经在各大公司广泛应用。

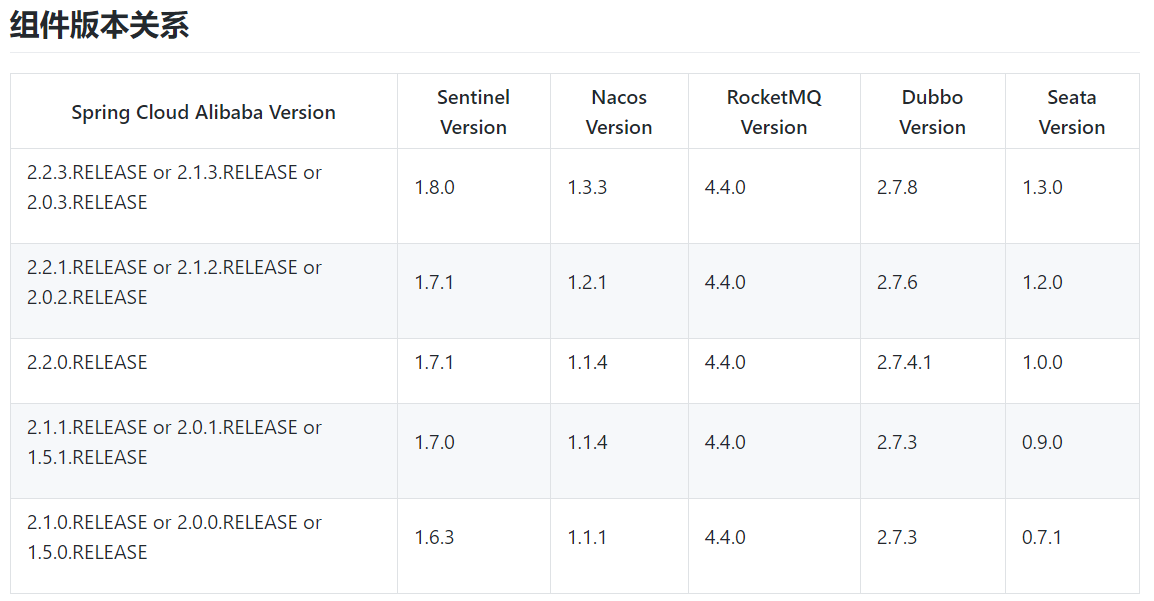
我所在的公司使用WebService来实现服务的远程通信，但是WebService在服务治理这块的能力是缺失的，因此使用Dubbo进行了全部替换。由于Dubbo天然支持多协议，因此在迁移和改造过程中并没有投入太多的成本。所以后来，在后续的几个公司中，也都是通过Dubbo来实现服务通信及服务治理的。

1. Alibaba的开源组件在服务治理上和处理高并发的能力上也有天然的优势，毕竟这些组件经历过了数次双11的考验，也在各大互联网公司大规模应用过。同时，Spring

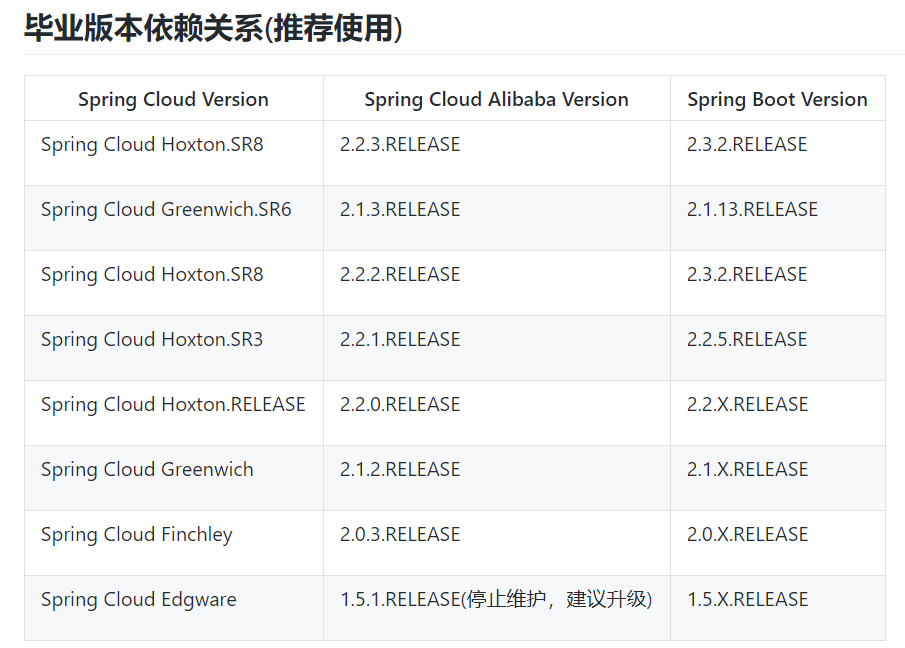
Cloud Alibaba 在功能上 不仅完全覆盖了Spring Cloud Netflix原生特性，而且还提供了更加稳定和成熟的实现。且更Spring Cloud在服务治理这块的能力更适合国内的技术场景。

## Spring Cloud Alibaba的版本

Spring Cloud Alibaba目前发布了4个毕业版本，每一个毕业版本有3个小版本，Spring Cloud Alibaba版本与组件的版本 之间的依赖关系如下：



同时，Spring Cloud大版本与Spring Cloud Alibaba和Spring Boot之间的依赖关系如下：



不难发现，Spring Cloud Alibaba的版本是从Spring Cloud Edgware这个大版本开始支持的。 这些版本依赖信息可以在<https://github.com/alibaba/spring-cloud-alibaba/wiki/版本信息>

查看。