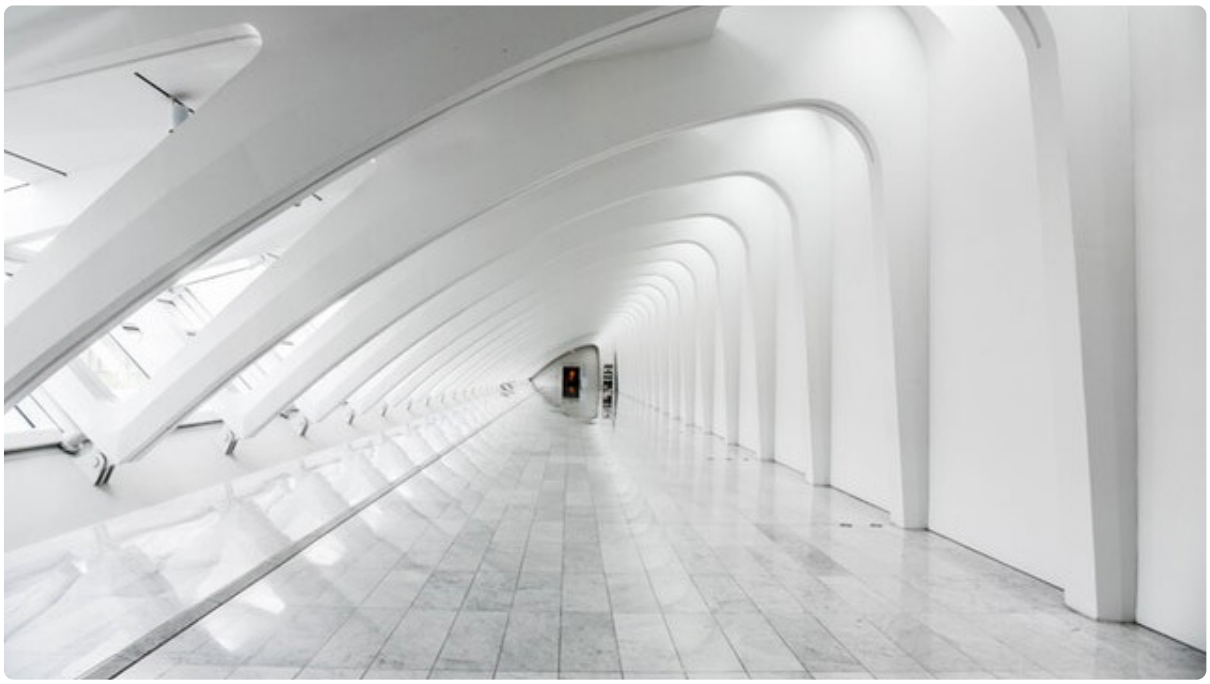
03 Spring Cloud 介绍以及发展前景

更新时间：2019-06-19 17:53:32



> 才能一旦让懒惰支配，它就一无可为。
>
> ——克雷洛夫

这些年 Spring Cloud 作为 Spring 官网重点推荐的产品，在社区内使用度越来越广泛。18年伴随着几次子框架停更事件，引发了技术社区的担心和大讨论，但其实 Spring Cloud 并没有受到这件事情的影响，反而发展越来越好，这是为什么呢？本节专栏为大家介绍这些有关 Spring Cloud 的趣事。

Spring Cloud 是什么？

Spring Cloud 是一系列框架的有序集合。Spring Cloud 内部包含了众多框架，这些框架相互协调构建分布式系统。Spring Cloud 构建在 Spring Boot 的基础上，利用了 Spring Boot 诸多特性简化了分布式系统基础设施的开发，让基础框架都可以做到一键启动和部署，使开发人员能够很容易地开始工作，并快速提高生产效率。

Spring Cloud 提供了快速构建分布式系统中一些常见模式的工具，例如配置管理、服务发现、断路器、智能路由、微代理、控制总线。利用好 Spring Cloud 相关工具，开发人员可以快速实现这些模式的服务和应用程序。Spring Cloud 为最常见的分布式系统模式提供了一个简单、可访问的编程模型，帮助开发人员构建弹性、可靠和协调的应用程序，让开发人员使用 Spring Cloud 部署分布式系统简单而又不容易出错。

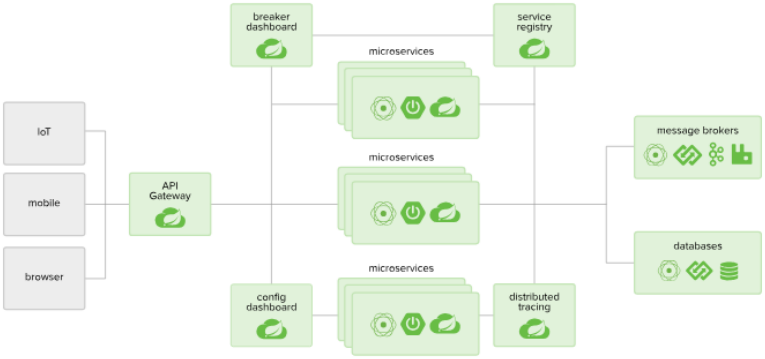
Spring Cloud 特性

Spring Cloud 致力于为典型用例提供良好的开箱即用体验，并提供覆盖其他用例的可扩展性机制，下面为它的核心特性：

- 分布式/版本化配置
- 服务注册和发现
- 路由

- 服务与服务之间的调用
- 负载均衡
- 断路器
- 分布式消息

我们用一张 Spring 官网的图，来展示这些核心组件之间是如何相互配合运行的：



1、外部物联网设备、手机或者浏览器统一通过服务网关来访问内部服务，服务网关负责分发到具体服务

2、服务运行过程中通过分布式配置中心获取相关配置信息

3、多次调用失败服务会进入断路器

4、服务与服务之间通过注册中心来相互调用

5、通过分布式链路跟踪来查看调用链

6、服务通过 Message Broker 来传递消息

7、服务运行过程中的数据保存在数据库中

8、通过页面可以监控服务熔断、配置信息、服务调用和链路跟踪等信息

Spring Cloud 命名故事

我们知道 Spring Cloud 本身并不是一个框架，是一系列框架的有序集合，每个子项目有不同的发行节奏，都维护着自己的发布版本号。如果 Spring Cloud 也按照传统的方式命名，就难免与子项目的发布号混淆，所以没有采用版本号的方式，而是使用了另外一种命名方式。

这些版本名称的命名方式采用了伦敦地铁站的名称，同时根据字母表的顺序来对应版本时间顺序，比如：最早的 Release 版本：Angel，第二个 Release 版本：Brixton，然后是 Camden、Dalston、Edgware、Finchley，目前最新的是 Greenwich 版本。平时为了方便会以首字母来简称 Spring Cloud 的版本，比如 Angel 就是 A 版本，最新的 Greenwich 简称它为 G 版本。

> 注意: Angel 和 Brixton 两个版本已于2017年7月终止不再进行维护。

同时在这些大版本命名之后，我们还会经常看到 M、RC、SR 等小版本，在此也给大家解释一下：

- M 是 milestone（里程碑）的缩写，代表了项目开发到达一个关键的里程碑，一般在开发过程中会经历很多个里程碑，所以会有 M1、M2 这样的命名；

- RC 是 Release Candidate（候选人）的缩写，代表了项目处于候选版本的状态，大多用在正式版本发布之前的一个阶段，因此也会有 RC1、RC2 这样的命名；
- SR 是 Service Release 的缩写，代表了项目发布正式稳定的版本，一般等同于 GA 版本：(GenerallyAvailable)。SR1 表示第 1 个正式版本，后面依次累加数字，会有 SR2、SR3 这样的命名；
- SNAPSHOT 快照版本，可以稳定使用，且仍在继续改进版本。

在一般的版本发布过程中会先发布 M 版本，再发布 RC 版本，最后发布 SR 版本。目前 Spring Cloud 的最新版本是 Greenwich.SR1，也就是 G 版本发布的第一个稳定版本，这也是我们本专栏课程采用的版本。

Spring Cloud 停更风波

Spring Cloud 并没有重复制造轮子。早期为了快速上线充分发挥了开源社区的力量，集成了业内诸多成熟的开源框架，其中就包含了 Netflix 开源的诸多产品，其中就有 Eureka、Hystrix、Zuul、Ribbon 等。

2018年6月突然传来消息，Eureka 2.0 停止开发，如果将 Eureka 2.0 用于生产，将后果自负。这个消息传开之后一度引发了大家对使用 Eureka 的恐慌，但其实真正使用 Eureka 2.0 的公司几乎没有，大家使用的都是 Eureka 1.X 系列，并不受到影响。另外 Spring Cloud 还有很多其它子框架来替代 Eureka 的使用，比如比较流行的 Consul、Zookeeper 等。

到了2018年的11月，Netflix 团队突然宣布 Hystrix 停止开发新版本，但仍然会维护后期可能出现的 Bug；其实除了 Hystrix 外， Spring Cloud 体系内同样存在几款替代框架，比如 Resilience4J、Sentinel 等。

到目前为止 Netflix 已经有多款都处于维护状态。在今年 1 月 Greenwich 版本发布时，以下 Netflix 产品都处于维护状态。

- spring-cloud-netflix-archaius
- spring-cloud-netflix-hystrix-contract
- spring-cloud-netflix-hystrix-dashboard
- spring-cloud-netflix-hystrix-stream
- spring-cloud-netflix-hystrix
- spring-cloud-netflix-ribbon
- spring-cloud-netflix-turbine-stream
- spring-cloud-netflix-turbine
- spring-cloud-netflix-zuul

并不包含 Eureka 和并发限制相关模块。

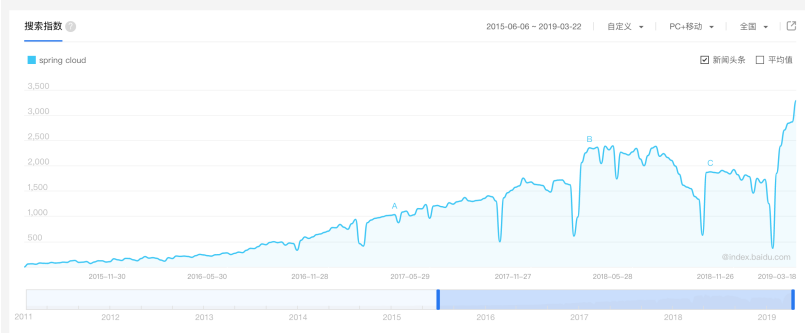
但一系列停更事件，也间接地反应了 Netflix 公司对开源软件的支持面临着不确定性，因此 Spring Cloud 在最新发布的 Greenwich 版本中，建议替换掉 Netflix 的几款相关组件：

现在的	替换的
Hystrix	Resilience4j
Hystrix Dashboard / Turbine	Micrometer + Monitoring System
Ribbon	Spring Cloud Loadbalancer
Zuul 1	Spring Cloud Gateway
Archaius 1	Spring Boot external config + Spring Cloud Config

本系列专栏会一一介绍上述建议替换的框架使用。

Spring Cloud 发展前景

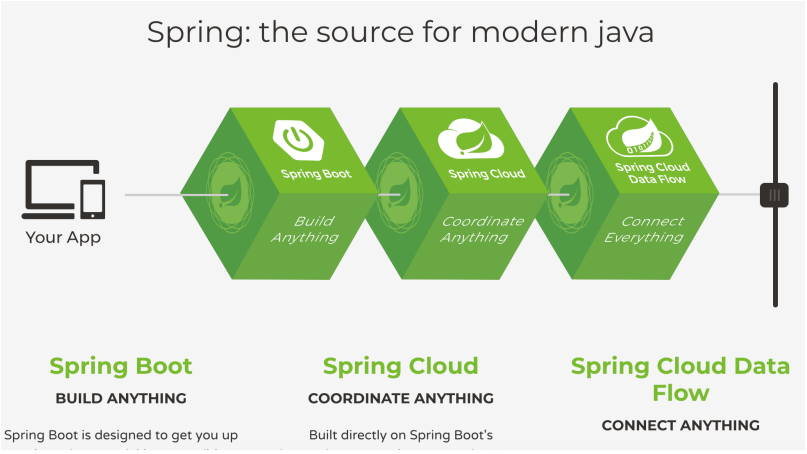
2015年6月 Spring Cloud 发布了第一个正式版本 Angel.SR1，到现在已经过去了快四年。在这四年时间中 Spring Cloud 从一个新生的框架，慢慢成长为微服务领域事实上的标准，在开源社区的知名度也越来越高，我们从 Spring Cloud 的百度指数也可以看出这个趋势。



上图为2015年到现在的 Spring Cloud 百度指数，其中掉下来的点是中国的春节。也可以明显地看出， 2019 年，社区对 Spring Cloud 的关注度更一步地提高。

随着 Spring Cloud 的不断发展，去年2018年10月，国内巨头阿里巴巴宣布正式入驻 Spring Cloud 官方孵化器。并且在加入不久后就贡献了几个重量级的框架，其中就有大名鼎鼎的 Sentinel、Nacos 以及阿里本身几款和微服务相关的中间件产品，关于阿里巴巴和 Spring Cloud 的相关内容本期专栏在后期也会详细介绍。

Spring 官方也非常重视 Spring Cloud 的后续发展，已经将它作为公司最顶级的项目来推广，放到了官网上最核心的位置。从近几期 Spring Cloud 大版本的发布频率，也能感受到 Spring 官方对 Spring Cloud 的重视， 相信未来 Spring 官方会继续投入更多的力量来发展 Spring Cloud 。



从上述来看，不论是官方的重视程度、社区内的关注度，还是各大巨头对 Spring Cloud 的参与度来讲，Spring Cloud 的未来发展必定会越来越好。

小结

Spring Cloud 对于中小型互联网公司来说是一种福音，因为这类公司往往没有实力或者没有足够的资金投入去开发自己的分布式系统基础设施，使用 Spring Cloud 一站式解决方案能在从容应对业务发展的同时大大减少开发成本。

同时，随着近几年微服务架构和 Docker 容器概念的火爆，也会让 Spring Cloud 在未来越来越“云”化的软件开发风格中立有一席之地，尤其是在目前五花八门的分布式解决方案中提供了标准化的、全站式的技术方案，意义可能会堪比微服务规范的诞生，有效推进服务端软件系统技术水平的进步。

本文作者：纯洁的微笑、江南一点雨

02 为什么要使用微服务架构？

04 Spring Boot 的设计理念和简单实践