微服务架构

将单体的应用程序分为多个应用程序，这些分出来的应用程序就是微服务，每个微服务运行在自己的进程中，并使用轻量级的机制通信；这些服务围绕业务能力来划分，并通过自动化部署机制来独立部署。这些服务可以使用不同的编程语言，不同数据库，以保证最低限度的集中式管理。

为什么要学习spring cloud

Spring cloud是基于springboot的，屏蔽掉了复杂的配置

Springcloud 开箱即用，使用springcloud只需要添加上对应的jar依赖就可以使用了

springcloud大多数子模块都是直击痛点，像zuul解决的跨域，fegin解决的负载均衡，hystrix的熔断机制等等等等

spring cloud是什么

springcloud是一系列框架的有效集合。它利用springboot的开发边便利性巧妙的简化了分布式系统基础设施的开发，如服务注册，配置中心，智能路由，消息总线，负载均衡，断路器，数据监控等，这些都是用springboot的开发风格做的，实现一键启动和部署

spring Cloud并没有重复制造轮子，它只是将各家公司开发的比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包

spring cloud的优缺点

优点

1.耦合度比较低。不会影响其他模块的开发。

2.减轻团队的成本，可以并行开发，不用关注其他人怎么开发，先关注自己的开发。

3.配置比较简单，基本用注解就能实现，不用使用过多的配置文件。

4.微服务跨平台的，可以用任何一种语言开发。

5.每个微服务可以有自己的独立的数据库也有用公共的数据库。

6.直接写后端的代码，不用关注前端怎么开发，直接写自己的后端代码即可，然后暴露接口，通过组件进行服务通信。

缺点

1.部署比较麻烦，给运维工程师带来一定的麻烦。

2.针对数据的管理比麻烦，因为微服务可以每个微服务使用一个数据库。

3.系统集成测试比较麻烦

4.性能的监控比较麻烦。【最好开发一个大屏监控系统】

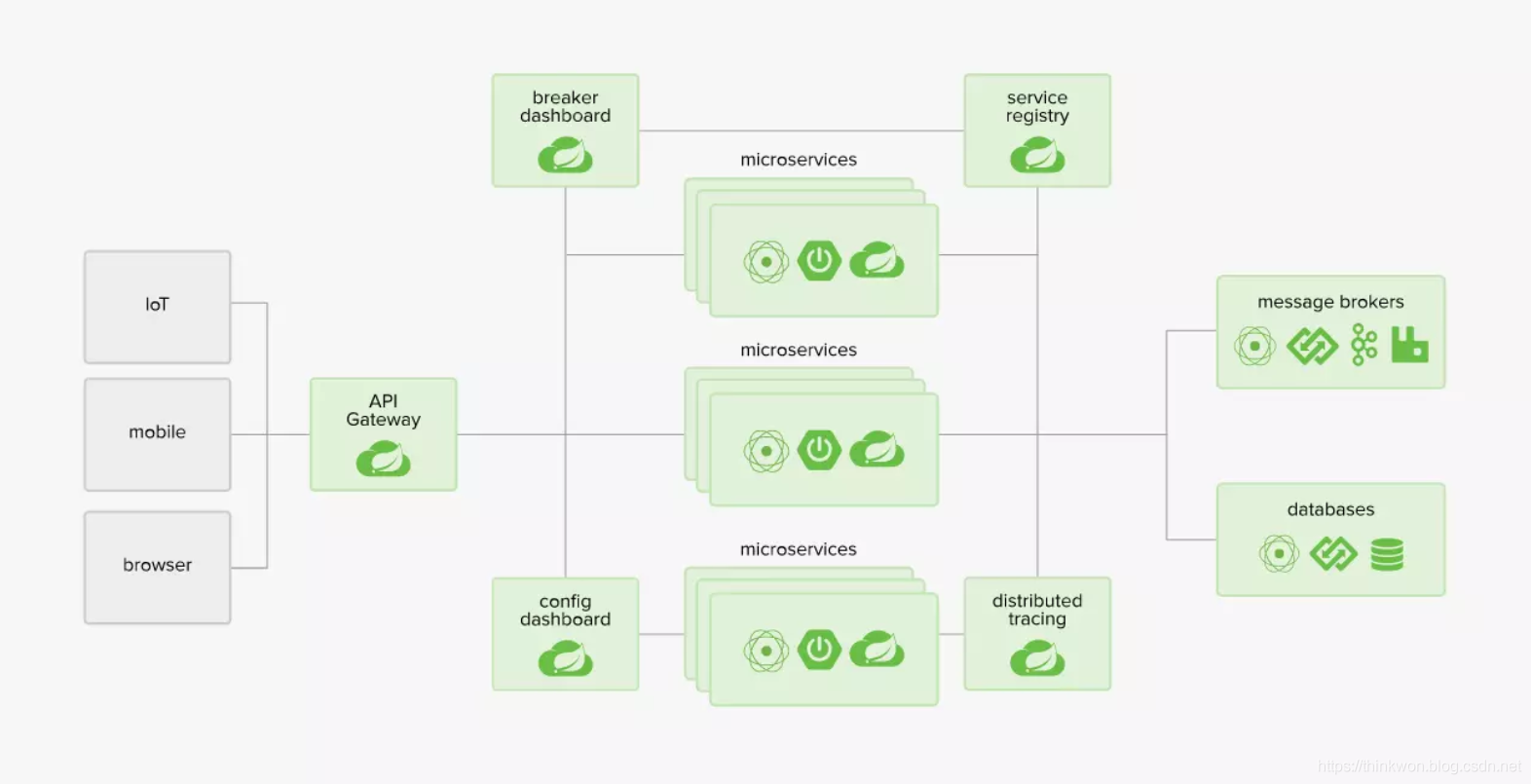
Spring boot和springcloud的区别

Springboot专注于快速方便的开发单个个体微服务

Springcloud是关注全局的微服务协调整理治理框架，将springboot开发的单个微服务整合并管理，为微服务之间提供配置管理，服务发现，断路器，路由，事件总线，全局锁，决策竞选，分布式会话等集成服务

Spring Cloud和SpringBoot版本对应关系

| **Spring Cloud Version** | **SpringBoot Version** |
| --- | --- |
| Hoxton | 2.2.x |
| Greenwich | 2.1.x |
| Finchley | 2.0.x |
| Edgware | 1.5.x |
| Dalston | 1.5.x |



Springcloud

开发中最重要的

Spring Cloud Eureka：服务注册与发现

Spring Cloud Zuul：服务网关

Spring Cloud Ribbon：客户端负载均衡

Spring Cloud Feign：声明性的Web服务客户端

Spring Cloud Hystrix：断路器

Spring Cloud Config：分布式统一配置管理

等20几个框架，开源一直在更新

开发分布式微服务面临的问题

（1）与分布式系统相关的复杂性-这种开销包括网络问题，延迟开销，带宽问题，安全问题。

（2）服务发现-服务发现工具管理群集中的流程和服务如何查找和互相交谈。它涉及一个服务目录，在该目录中注册服务，然后能够查找并连接到该目录中的服务。

（3）冗余-分布式系统中的冗余问题。

（4）负载平衡 --负载平衡改善跨多个计算资源的工作负荷，诸如计算机，计算机集群，网络链路，中央处理单元，或磁盘驱动器的分布。

（5）性能-问题 由于各种运营开销导致的性能问题。

（6）部署复杂性-Devops 技能的要求。

Springcloud和dubbo的区别

服务调用方式：springcloud使用rest的api，dubbo使用rpc

注册中心：dubbo是用zookeeper，springcloud使用Eureka，也可以使用zookeeper

服务网关：dubbo没有实现，需要整合第三方技术来支持，springcloud提供zuul路由来网关，作为路由服务器，进行消费者的请求分发,springcloud支持断路器，与git完美集成配置文件支持版本控制，事物总线实现配置文件的更新与服务自动装配等等一系列的微服务架构要素。

Eureka

什么是eureka

作为springcloud的服务注册功能服务器，它是服务注册中心，系统中的其他服务使用eureka客户端将其连接到eureka service中，并保持心跳，这样可以通过Eureka Service来监控各个微服务是否运行正常。

服务发现和注册是什么意思

当我们开始一个项目时，我们通常在属性文件中进行所有的配置。随着越来越多的服务开发和部署，添加和修改这些属性变得更加复杂。有些服务可能会下降，而某些位置可能会发生变化。手动更改属性可能会产生问题。 Eureka 服务注册和发现可以在这种情况下提供帮助。由于所有服务都在 Eureka 服务器上注册并通过调用 Eureka 服务器完成查找，因此无需处理服务地点的任何更改和处理

Eureka实现高可用

集群，注册多台Eureka，然后把SpringCloud服务互相注册，客户端从Eureka获取信息时，按照Eureka的顺序来访问。

什么是eureka的自我保护

Eureka在一定时间内没有接收到某个服务的心跳，eureka服务就会进入自我保护模式，在该模式下eurekaservice会保护服务注册表的信息，不删除注册表的数据，当节点恢复后，eurekaservice节点会自动退出自我保护模式

默认情况下，如果 EurakeServer 在一定时间内没有接收到某个服务实例的心跳，EurakeServer 将会注销实例（默认90秒）。但是当网络分区故障发生时，微服务与 EurakeServer 之间无法正常通信，以上行为可能变的非常危险，因为微服务本身是健康的，此时本不应该注销这个服务。Eurake 通过“自我保护”来解决这个问题，当 EurakeServer 节点在短时间内丢失过多客户时（可能发生了网络分区故障），那么这个节点就会进入自我保护模式。一旦进入该模式，EurakeServer 就会保护服务注册表中的信息，不在删除服务注册表中的数据（也就是不会注销任何微服务）。当故障恢复后，该 EurakeServer 节点会自动退出自我保护模式。

DiscoveryClient的作用

可以从注册中心中根据服务别名获取注册的服务器信息。

Eureka和zookeeper的区别

zookeeper 保证 CP

　　当向注册中心查询服务列表时，我们可以容忍注册中心返回的是几分钟以前的信息，但不能就收服务直接 down 掉不可用。也就是说服务注册的可用性要高于一致性

　　当时 zk 会出现这么一个情况，当 mastr 节点因网络故障和其他节点失去联系时，剩余节点会重新进行选举。问题在于，选举时间比较长，30s~120s，且选举期间，整个 zk 是不可用的。这就导致了在选举期间，注册服务的瘫痪。在云部署的环境下，因网络问题使 zk 集群时区 master 节点是交大概率会发生的事情，虽然服务能够最终恢复，但是漫长的选举时间导致的注册服务长期不可用是不能容忍的。

Eureka 保证 AP

　　Eureka 明白这一点，因此在设计时，就优先保证可用性.Eureka 各个节点是平等的，几个节点挂掉不会影响正常工作，只要有一台 Eureka 存在，就可以保证注册服务可用（保证可用性），只不过查到的信息可能不是最新的（不保证强一致性）。除此之外，Eureka 还有一种自我保护机制，如果在 15 分钟内超过 85% 的节点没有正常的心跳，那么 Eureka 就会认为客户端与注册中心出现了故障，此时会出现以下几种情况：

　　1、Eureka 不再从注册列表中移出因长时间没收到心跳而应该过期的服务

　　2、Eureka 仍然能够接受新服务的注册和查询要求，但是不会被同步到其他节点上（即保证当前节点依然可用）

　　3、当网络稳定时，当前实例新的注册信息会被同步到其他节点中

**结论**

　　Eureka 可以很好的应对因网络故障导致部分节点失去联系的情况，而不会像 zookeeper 那样使整个祖册服务瘫痪

Zuul

什么是网关

一个网络服务架构的入口，所有的网络请求都必须通过网关转发到具体的服务

网关的作用是什么

统一管理微服务请求，权限控制、负载均衡、路由转发、监控、安全控制黑名单和白名单等

网关与过滤器有什么区别

网关是对所有服务的请求进行分析过滤，过滤器是对单个服务而言。

常用网关框架有那些？

Nginx、Zuul、Gateway

Zuul与Nginx有什么区别？

Zuul是java语言实现的，主要为java服务提供网关服务，尤其在微服务架构中可以更加灵活的对网关进行操作。Nginx是使用C语言实现，性能高于Zuul，但是实现自定义操作需要熟悉lua语言，对程序员要求较高，可以使用Nginx做Zuul集群。

既然Nginx可以实现网关？为什么还需要使用Zuul框架

Zuul是SpringCloud集成的网关，使用Java语言编写，可以对SpringCloud架构提供更灵活的服务。

如何设计一套API接口

考虑到API接口的分类可以将API接口分为开方API接口和内网API接口，内网API接口用于局域网，为内部服务器提供服务。开放API接口用于对外部合作单位提供接口调用，需要遵循Oauth2.0权限认证协议。同时还需要考虑安全性、幂等性等问题。

ZuulFilter常用有那些方法

Run()：过滤器的具体业务逻辑

shouldFilter()：判断过滤器是否有效

filterOrder()：过滤器执行顺序

filterType()：过滤器拦截位置

如何实现动态Zuul网关路由转发

通过path配置拦截请求，通过ServiceId到配置中心获取转发的服务列表，Zuul内部使用Ribbon实现本地负载均衡和转发。

Zuul网关如何搭建集群

使用Nginx的upstream设置Zuul服务集群，通过location拦截请求并转发到upstream，默认使用轮询机制对Zuul集群发送请求。

Ribbon

Ribbon是什么

Ribbon是Netflix发布的开源项目，主要功能是提供客户端的软件负载均衡算法

Ribbon客户端组件提供一系列完善的配置项，如连接超时，重试等。简单的说，就是在配置文件中列出后面所有的机器，Ribbon会自动的帮助你基于某种规则（如简单轮询，随即连接等）去连接这些机器。我们也很容易使用Ribbon实现自定义的负载均衡算法。（有点类似Nginx）

Nginx和ribbon的区别

Nginx是反向代理同时可以实现负载均衡，nginx拦截客户请求采用负载均衡算法根据upstream配置进行转发，请求通过nginx服务器转发，ribbon是客户端负载均衡，从注册中心读取目标服务信息，将客户端采用轮询策略对服务直接访问，全称客户端操作

Ribbon底层实现原理

Ribbon使用discoveryClient从注册中心读取目标服务信息，对同一接口请求进行计数，使用%取余算法获取目标服务集群索引，返回获取到的目标服务信息。

@LoadBalanced注解的作用

​ 开启客户端负载均衡。

Hystrix

断路器

服务器雪崩：当一个服务调用另一个服务由于网络原因或自身原因出现问题，调用者就会等待被调用者的响应 当更多的服务请求到这些资源导致更多的请求等待，发生连锁效应（雪崩效应）

断路器有三种状态

打开状态：一段时间内 达到一定的次数无法调用 并且多次监测没有恢复的迹象 断路器完全打开 那么下次请求就不会请求到该服务

半开状态：短时间内 有恢复迹象 断路器会将部分请求发给该服务，正常调用时 断路器关闭

关闭状态：当服务一直处于正常状态 能正常调用

在分布式系统，我们一定会依赖各种服务，那么这些个服务一定会出现失败的情况，就会导致雪崩，Hystrix就是这样的一个工具，防雪崩利器，它具有服务降级，服务熔断，服务隔离，监控等一些防止雪崩的技术。

Hystrix有四种防雪崩方式:

服务降级：接口调用失败就调用本地的方法返回一个空

服务熔断：接口调用失败就会进入调用接口提前定义好的一个熔断的方法，返回错误信息

服务隔离：隔离服务之间相互影响

服务监控：在服务发生调用时,会将每秒请求数、成功请求数等运行指标记录下来。

在微服务中，如何保护服务?

一般使用使用Hystrix框架，实现服务隔离来避免出现服务的雪崩效应，从而达到保护服务的效果。当微服务中，高并发的数据库访问量导致服务线程阻塞，使单个服务宕机，服务的不可用会蔓延到其他服务，引起整体服务灾难性后果，使用服务降级能有效为不同的服务分配资源,一旦服务不可用则返回友好提示，不占用其他服务资源，从而避免单个服务崩溃引发整体服务的不可用.

**谈谈服务降级、熔断、服务隔离**

* 服务降级：当客户端请求服务器端的时候，防止客户端一直等待，不会处理业务逻辑代码，直接返回一个友好的提示给客户端。
* 服务熔断是在服务降级的基础上更直接的一种保护方式，当在一个统计时间范围内的请求失败数量达到设定值（requestVolumeThreshold）或当前的请求错误率达到设定的错误率阈值（errorThresholdPercentage）时开启断路，之后的请求直接走fallback方法，在设定时间（sleepWindowInMilliseconds）后尝试恢复。
* 服务隔离就是Hystrix为隔离的服务开启一个独立的线程池，这样在高并发的情况下不会影响其他服务。服务隔离有线程池和信号量两种实现方式，一般使用线程池方式。

Feign

声明一个web服务客户端，这使得编写web服务更容易，他将我们需要调用的服务方法定义为抽象方法保存在本地就可以，不需要自己构建HTTP请求，直接调用接口就行了，不过要注意，调用方法要和本地抽象方法的签名完全一致

Springcloud调用接口的方式

Feign

RestTemplate

Ribbon和Feign调用服务的区别

调用方式不同：Ribbon需要我们自己构建Http请求，模拟Http请求然后通过RestTemplate发给其他服务，步骤相当繁琐

而Feign则是在Ribbon的基础上进行了一次改进，采用接口的形式，将我们需要调用的服务方法定义成抽象方法保存在本地就可以了，不需要自己构建Http请求了，直接调用接口就行了，不过要注意，调用方法要和本地抽象方法的签名完全一致。

Bus

Spring cloud bus:就像一个分布式执行器，用于扩展的spring boot应用程序的配置文件，但也可以用作应用程序之间的通信

Spring Cloud Bus 不能单独完成通信，需要配合MQ支持

Spring Cloud Bus一般是配合Spring Cloud Config做配置中心的

Springcloud config实时刷新也必须采用SpringCloud Bus消息总线

Config

什么是Spring Cloud Config?

Spring Cloud Config为分布式系统中的外部配置提供服务器和客户端支持，可以方便的对微服务各个环境下的配置进行集中式管理。Spring Cloud Config分为Config Server和Config Client两部分。Config Server负责读取配置文件，并且暴露Http API接口，Config Client通过调用Config Server的接口来读取配置文件。

分布式配置中心有那些框架？

Apollo、zookeeper、springcloud config。

分布式配置中心的作用？

动态变更项目配置信息而不必重新部署项目。

Springcloud config可以实现实时刷新

Springcloud config实时刷新采用spring cloud bus消息总线

GateWay

Spring cloud gateway

Spring Cloud Gateway是Spring Cloud官方推出的第二代网关框架，取代Zuul网关。网关作为流量的，在微服务系统中有着非常作用，网关常见的功能有路由转发、权限校验、限流控制等作用。

使用了一个RouteLocatorBuilder的bean去创建路由，除了创建路由RouteLocatorBuilder可以让你添加各种predicates和filters，predicates断言的意思，顾名思义就是根据具体的请求的规则，由具体的route去处理，filters是各种过滤器，用来对请求做各种判断和修改。

Spring cloud的主要项目

Spring Cloud的子项目，大致可分成两类，一类是对现有成熟框架"Spring Boot化"的封装和抽象，也是数量最多的项目；第二类是开发了一部分分布式系统的基础设施的实现，如Spring Cloud Stream扮演的就是kafka, ActiveMQ这样的角色

Spring Cloud Config

Config能够管理所有微服务的配置文件

集中配置管理工具，分布式系统中统一的外部配置管理，默认使用Git来存储配置，可以支持客户端配置的刷新及加密、解密操作。

Spring Cloud Netflix(重点，这些组件用的最多)

Netflix OSS 开源组件集成，包括Eureka、Hystrix、Ribbon、Feign、Zuul等核心组件。

Eureka：服务治理组件，包括服务端的注册中心和客户端的服务发现机制；

Ribbon：负载均衡的服务调用组件，具有多种负载均衡调用策略；

Hystrix：服务容错组件，实现了断路器模式，为依赖服务的出错和延迟提供了容错能力；

Feign：基于Ribbon和Hystrix的声明式服务调用组件；

Zuul：API网关组件，对请求提供路由及过滤功能。

Spring Cloud Bus

用于传播集群状态变化的消息总线，使用轻量级消息代理链接分布式系统中的节点，可以用来动态刷新集群中的服务配置信息。

简单来说就是修改了配置文件，发送一次请求，所有客户端便会重新读取配置文件。

需要利用中间插件MQ

Spring Cloud Consul

Consul 是 HashiCorp 公司推出的开源工具，用于实现分布式系统的服务发现与配置。与其它分布式服务注册与发现的方案，Consul 的方案更“一站式”，内置了服务注册与发现框架、分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案，不再需要依赖其它工具（比如 ZooKeeper 等）。使用起来也较为简单。Consul 使用 Go 语言编写，因此具有天然可移植性(支持Linux、windows和Mac OS X)；安装包仅包含一个可执行文件，方便部署，与 Docker 等轻量级容器可无缝配合。

Spring Cloud Security

安全工具包，他可以对

对Zuul代理中的负载均衡从前端到后端服务中获取SSO令牌

资源服务器之间的中继令牌

使Feign客户端表现得像OAuth2RestTemplate（获取令牌等）的拦截器

在Zuul代理中配置下游身份验证

Spring Cloud Security提供了一组原语，用于构建安全的应用程序和服务，而且操作简便。可以在外部（或集中）进行大量配置的声明性模型有助于实现大型协作的远程组件系统，通常具有中央身份管理服务。它也非常易于在Cloud Foundry等服务平台中使用。在Spring Boot和Spring Security OAuth2的基础上，可以快速创建实现常见模式的系统，如单点登录，令牌中继和令牌交换。

Spring Cloud Sleuth

在微服务中，通常根据业务模块分服务，项目中前端发起一个请求，后端可能跨几个服务调用才能完成这个请求（如下图）。如果系统越来越庞大，服务之间的调用与被调用关系就会变得很复杂，假如一个请求中需要跨几个服务调用，其中一个服务由于网络延迟等原因挂掉了，那么这时候我们需要分析具体哪一个服务出问题了就会显得很困难。Spring Cloud Sleuth服务链路跟踪功能就可以帮助我们快速的发现错误根源以及监控分析每条请求链路上的性能等等。

Spring Cloud Stream

轻量级事件驱动微服务框架，可以使用简单的声明式模型来发送及接收消息，主要实现为Apache Kafka及RabbitMQ。

Spring Cloud Task

Spring Cloud Task的目标是为Spring Boot应用程序提供创建短运行期微服务的功能。在Spring Cloud Task中，我们可以灵活地动态运行任何任务，按需分配资源并在任务完成后检索结果。Tasks是Spring Cloud Data Flow中的一个基础项目，允许用户将几乎任何Spring Boot应用程序作为一个短期任务执行。

Spring Cloud Zookeeper

SpringCloud支持三种注册方式Eureka， Consul(go语言编写)，zookeeper

Spring Cloud Zookeeper是基于Apache Zookeeper的服务治理组件。

Spring Cloud Gateway

Spring cloud gateway是spring官方基于Spring 5.0、Spring Boot2.0和Project Reactor等技术开发的网关，Spring Cloud Gateway旨在为微服务架构提供简单、有效和统一的API路由管理方式，Spring Cloud Gateway作为Spring Cloud生态系统中的网关，目标是替代Netflix Zuul，其不仅提供统一的路由方式，并且还基于Filer链的方式提供了网关基本的功能，例如：安全、监控/埋点、限流等。

Spring Cloud OpenFeign

Feign是一个声明性的Web服务客户端。它使编写Web服务客户端变得更容易。要使用Feign，我们可以将调用的服务方法定义成抽象方法保存在本地添加一点点注解就可以了，不需要自己构建Http请求了，直接调用接口就行了，不过要注意，调用方法要和本地抽象方法的签名完全一致。

Spring Cloud的版本关系

* Spring Cloud是一个由许多子项目组成的综合项目，各子项目有不同的发布节奏。 为了管理Spring Cloud与各子项目的版本依赖关系，发布了一个清单，其中包括了某个Spring Cloud版本对应的子项目版本。 为了避免Spring Cloud版本号与子项目版本号混淆，Spring Cloud版本采用了名称而非版本号的命名，这些版本的名字采用了伦敦地铁站的名字，根据字母表的顺序来对应版本时间顺序，例如Angel是第一个版本，Brixton是第二个版本。 当Spring Cloud的发布内容积累到临界点或者一个重大BUG被解决后，会发布一个"service releases"版本，简称SRX版本，比如Greenwich.SR2就是Spring Cloud发布的Greenwich版本的第2个SRX版本。目前Spring Cloud的最新版本是Hoxton。