1. SpringCloud
2. SpringCloud微服务与Dubbo的区别

dubbo的服务是通过RPC实现的，接口强依赖（RPC远程过程调用）通过TCP等一大堆复杂的过程

SpringCloud是使用REST API ,REST 相比RPC更为轻量化，服务方和调用方的依赖只是靠一纸契约，不存在代码级别的强依赖。SpringCloud组合了大量的组件

通过上面再几个环节上的分析，相信大家对Dubbo和Spring Cloud有了一个初步的了解。就我个人对这两个框架的使用经验和理解，打个不恰当的比喻：使用Dubbo 构建的微服务架构就像组装电脑，各环节我们的选择自由度很高，但是最终结果很有可能因为一条内存质量不行就点不亮了，总是让人不怎么放心，但是如果你是一 名高手，那这些都不是问题；而Spring Cloud就像品牌机，在Spring Source的整合下，做了大量的兼容性测试，保证了机器拥有更高的稳定性，但是如果要在使用非 原装组件外的东西，就需要对其基础有足够的了解。

1. SpringCloud与SpringBoot 区别

SpringBoot 是Spring快速配置脚手架，可以基于SpringBoot快速开发单个微服务，SpringBoot基于Spring，是Spring的学习和使用便可快速，不仅适合原有工程结构，更适合微服务开发

SpringCloud基于SpringBoot 为微服务体系开发中的架构问题，提供了一整套的解决方案，微服务注册与发现，服务消费，服务保护与熔断，网关，分布式调用追踪，分布式配置管理等。SpringCloud依赖于SpringBoot

1. Eureka
2. 服务器雪崩

在微服务架构中，根据业务来拆分成一个个的服务，服务与服务之间可以相互调用（RPC）,在SpringCloud可以用RestTemplate+Ribbon和Feign来调用。为了保证高可用，单个服务通常会集群部署。由于网络原因或者服务自身的原因，服务并不能保证100%可用，如果单个服务出现问题，调用这个服务就会出现线程阻塞，此时若有大量的请求涌入，Servlet容器的线程资源会被消耗完毕，从而导致服务器瘫痪。服务与服务之间的依赖性，故障会传播，会对整个微服务系统造成灾难性的严重后果，这就是服务故障雪崩效应

1. 如何防止雪崩

解决方案：对依赖做隔离，hystrix就是处理依赖隔离的框架，同时也是可以帮助我们做依赖服务的治理和监控。

如何解决依赖隔离：

1. Hystrix使用命令模式HystrixCommand（command）包装依赖调用逻辑，每个命令在单独线程中/信号授权下执行
2. 可配置依赖调用超时时间，超时时间一般设置为比99.5%平均时间略高即可，当调用超时时，直接返回或执行fallback逻辑
3. 为每个依赖提供一个小的线程池（或信号），如果线程池已满调用将被立即拒绝，默认不采用排队，加速判断时间。
4. 依赖调用结果分：成功，失败（抛出异常），超时，线程拒绝，短路。请求失败（异常，拒绝，超时，短路）时执行fallback(降级逻辑)
5. 提供熔断器组件，可以自动运行或手动调用，停止当前依赖一段时间（10秒），熔断器默认错误率阈值为50%，查过将自动运行。
6. 提供实时依赖的统计和监控