## <https://www.zhihu.com/question/283286745/answer/763040709>

## 服务治理框架**Eureka**

Eureka专门用于给其他服务注册的称为Eureka Server(服务注册中心)，其余注册到Eureka Server的服务称为Eureka Client，Eureka Client**分为服务提供者和服务消费者**。

通过服务名来调用

**RestTemplate**远程调用

**Ribbon**客户端负载均衡

## **Hystrix熔断器(服务者)**

## **Hystrix仪表盘**

/Hystrix.stream监控单个服务

/turbine.stream是对集群使用

## **Feign**

**整合**了 Spring Cloud Ribbon 与 Spring Cloud Hystrix, 除了整合这两者的强大功能之外，它还提供了**声明式的服务调用**(不再通过RestTemplate)。

## **Zuul API网关**

· SpringCloud Zuul通过与SpringCloud Eureka进行整合，将自身注册为Eureka服务治理下的应用，同时从Eureka中获得了所有其他微服务的实例信息。**外层调用都必须通过API网关**，使得**将维护服务实例的工作交给了服务治理框架自动完成**。

· 在API网关服务上进行统一调用来**对微服务接口做前置过滤**，以实现对微服务接口的**拦截和校验**。

Zuul天生就拥有线程隔离和断路器的自我保护功能，以及对服务调用的客户端负载均衡功能。也就是说：**Zuul也是支持Hystrix和Ribbon**。

* zuul是对外暴露的唯一接口相当于路由的是controller的请求，而Ribbon和Fegin路由了service的请求
* zuul做最外层请求的负载均衡 ，而Ribbon和Fegin做的是系统内部各个微服务的service的调用的负载均衡

## **SpringCloud Config**

Spring Cloud Config项目是一个解决分布式系统的配置管理方案。它包含了Client和Server两个部分，**server提供配置文件的存储、以接口的形式将配置文件的内容提供出去，client通过接口获取数据、并依据此数据初始化自己的应用**。

* 简单来说，使用Spring Cloud Config就是将配置文件放到**统一的位置管理**(比如GitHub)，客户端通过接口去获取这些配置文件。
* 在GitHub上修改了某个配置文件，应用加载的就是修改后的配置文件。