注册中心(eureka server)：保存服务的地址信息 可以集群，互相备份

服务提供者(eureka client):自身提供服务 注册到注册中心

服务消费者(eureka client):连接到注册中心 调用服务（resttemplate、feign） 使用ribbon负载均衡调哪个、 使用hystrix以防调用的服务中断、

Hystrix-Dashboard:可以单独建一个项目对使用了hystrix的服务消费者进行监控；

Hystrix-Dashboard项目 -------监控----> 服务消费者（使用了hystrix,并且需要管理ServletRegistrationBean）

<http://localhost:10000/hystrix> -----> <http://localhost:9000/actuator/hystrix.stream>

Turbine集群监控

再建一个turbine监控工程 通过配置application.properties

turbine.app-config=eureka-client-consumer

turbine.combine-host-port=true

turbine.cluster-name-expression=new String("default")

说明要监控哪些服务，监控集群的名称default

这样就相当于把多个单个服务汇集在一块，启动turbine项目，浏览器输入

<http://localhost:20000/turbine.stream>可以看到请求情况的数据；

打开dashboard工程启动，在<http://localhost:10000/hystrix>中输入

<http://localhost:20000/turbine.stream>可以可视化看到情况；

<http://localhost:20000/turbine.stream>类比于 <http://localhost:9000/actuator/hystrix.stream>

注意点

1.被监控消费者需要向注册中心注册自己eureka.client.register-with-eureka=true

2、turbine.cluster-name-expression=new String("default") 这个default不能改（这是自己测试发现的）

总结

Dashboard监控工程提供了一个可视化页面 <http://localhost:10000/hystrix>

监控单个项目只需要消费者使用了hystrix, 并且需要管理ServletRegistrationBean，对应展示json格式的监控地址就是 <http://localhost:9000/actuator/hystrix.stream>，填入可视化页面即可可视化；

监控集群需要另开turbine监控工程，配置一下application.properties，并且有2个注意点，

然后启动项目，对应展示json格式的监控地址就是<http://localhost:20000/turbine.stream>，填入可视化页面即可可视化；

Zuul就是加一层大门，不直接调用服务，提供zuul转发调用服务，于是就可以做一些过滤器操作，比如ip黑名单；在zuul层面调用服务就跟服务消费者调用服务提供者一样会存在容错、回退（降级）机制；服务之间粒度是接口，zuul跟服务之间粒度就是服务；

如果想要 API 网关也能随时水平扩展，那么我们可以用脚本来动态修改 Nginx 的配置，通过脚本操作，发现有新加入的网关服务或者下线的网关服务，直接修改 Nginx 的 upstream，然后通过重载（reload）配置来达到网关的动态扩容。  
  
如果不用脚本结合注册中心去做的话，就只能提前规划好 N 个节点，然后手动配置上去。

Springcloud config 就是把配置文件交由github存储，然后本地拉取； 配合actuator 和fresh机制更新 需要手动调接口 配合github的webhook可以触发调用接口；如果有多个端要出发就需要引入bus 通过rabbitmq通知所有客户端更新配置代码；一篇看懂Springcloud config <https://www.cnblogs.com/fengzheng/p/11242128.html>

由于Springcloud config支持性、功能性不太好，还要依赖git，所以选择使用Apollo;