【Spring Cloud Sleuth】

# 什么是链路追踪

官网：<https://spring.io/projects/spring-cloud-sleuth>

链路追踪就是：追踪微服务的调用路径

# 链路追踪的由来

在微服务框架中，一个由客户端发起的请求在后端系统中会经过多个不同的服务节点调用来协同产生最后的请求结果，每一个请求都会开成一条复杂的分布式服务调用链路，链路中的任何一环出现高延时或错误都会引导起整个请求最后的失败。（不建议微服务中链路调用超过 3次）

# 分布式链路调用的监控

sleuth+zipkin（zipkin 就是一个可视化的监控控制台）

Zipkin 是 Twitter 的一个开源项目，允许开发者收集 Twitter 各个服务上的监控数据，并提供查询接口。

该系统让开发者可通过一个 Web 前端轻松的收集和分析数据，例如用户每次请求服务的处理时间等，可方便的监测系统中存在的瓶颈。

1. **Zipkin(**可视化平台**)**
   1. 下载 **zipkin**

SpringCloud 从 F 版以后已不需要自己构建 Zipkin server 了，只需要调用 jar 包即可 <https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/>

<https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/2.12.9/>

登不上去，只能去<https://pan.baidu.com/s/1rVb_R8i6uCiFXEWxCiDkcg?pwd=dljd> 扒一个

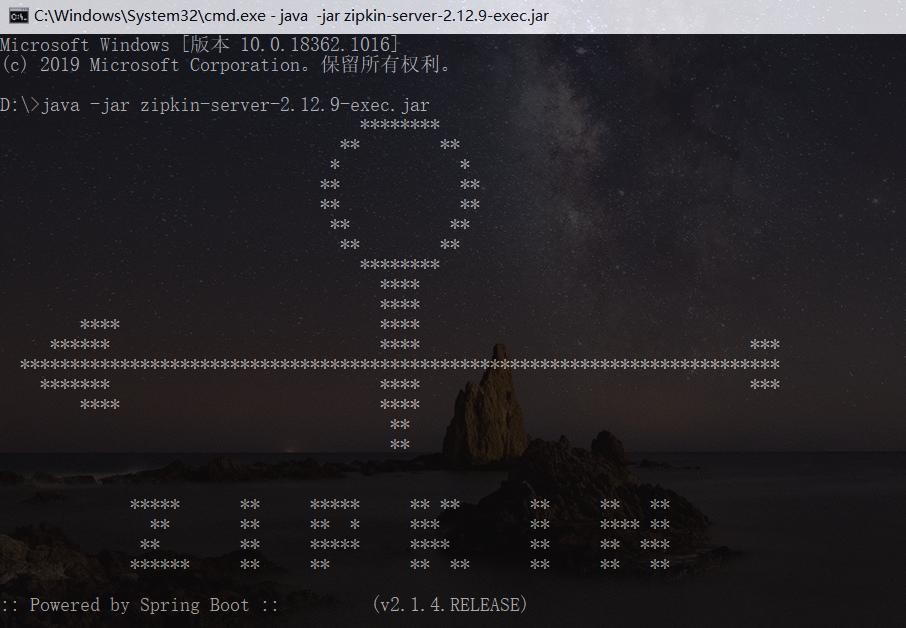
目前放这里了：E:\BaiduNetdiskDownload\软件



* 1. 运行 **zipkin**

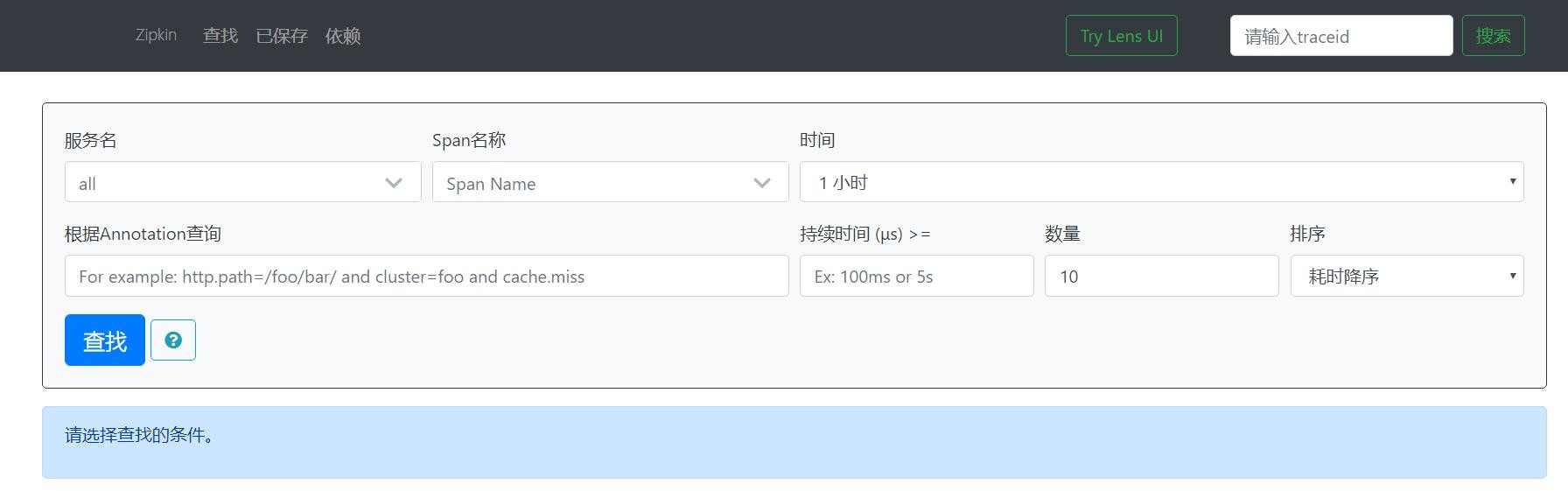
windos下直接运行

java -jar zipkin-server-2.12.9-exec.jar



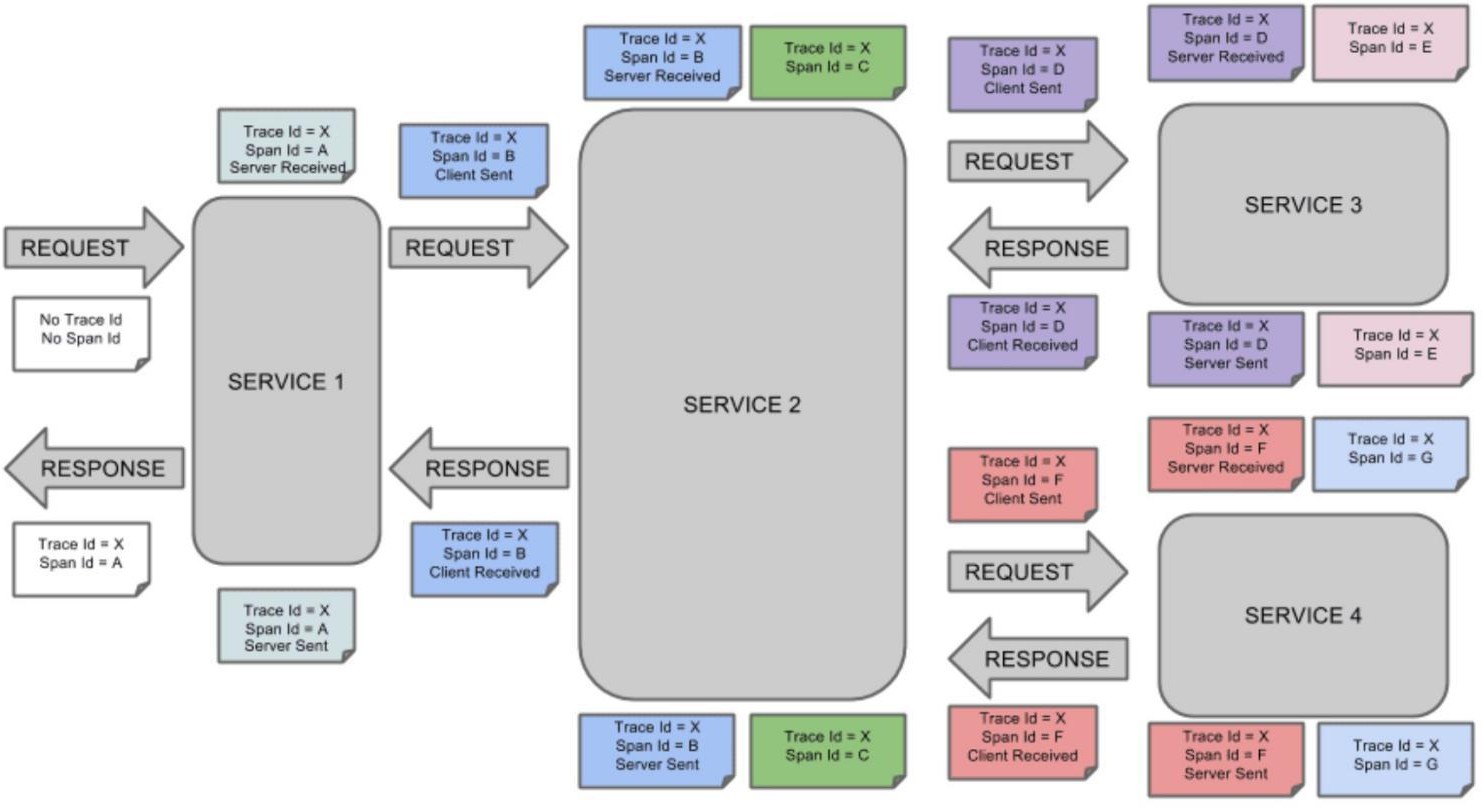
* 1. 查看 **zipkin** 的控制台

http://localhost:9411

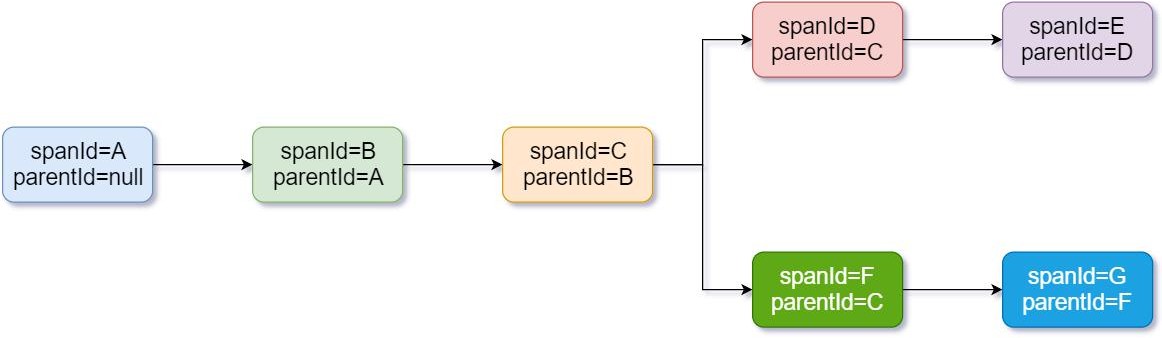


# 相关术语和名词解释

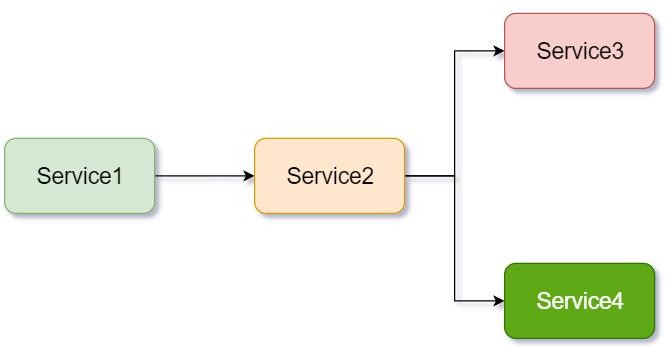
## 完整调用链路



* 1. 简化后



调用链路的依赖关系



## 名词解释

Trace:类似于树结构的 Span 集合，表示一条调用链路，存在唯一标识 span:表示调用链路来源，通俗的理解 span 就是一次请求信息

1. **Sleuth** 快速入门

## 案例说明

这里我们就不搭建项目了，直接用之前的案例 eureka-server

consumer-user-service provider-order-service

## 添加依赖

consumer-user-service 和 provider-order-service 都要加

（这两个是04-hystrix/04-feign-project下的模块

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-zipkin</artifactId>

</dependency>

## 修改配置文件

consumer-user-service 和 provider-order-service 都要该配置文件

spring:

zipkin:

base-url: http://localhost:9411 sleuth:

sampler:

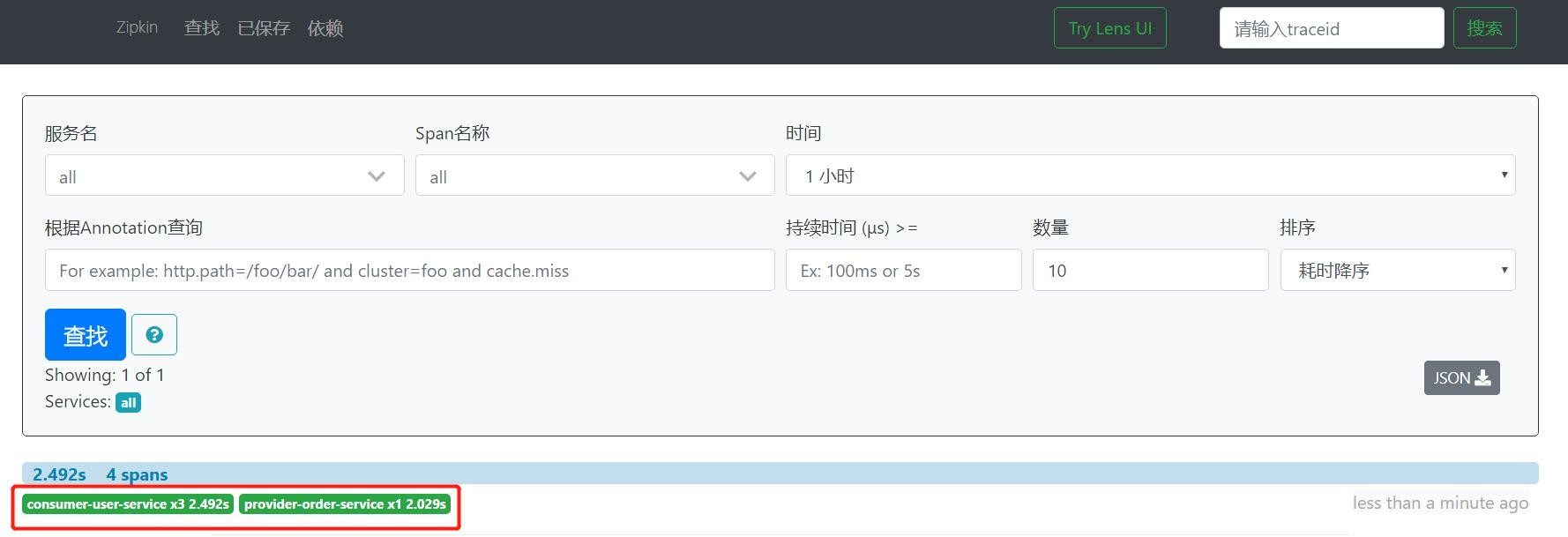
probability: 1 #配置采样率 默认的采样比例为: 0.1，即 10%，所设置的值介于 0 到 1 之间，1 则表示全部采集

rate: 10 #为了使用速率限制采样器,选择每秒间隔接受的 trace 量，最

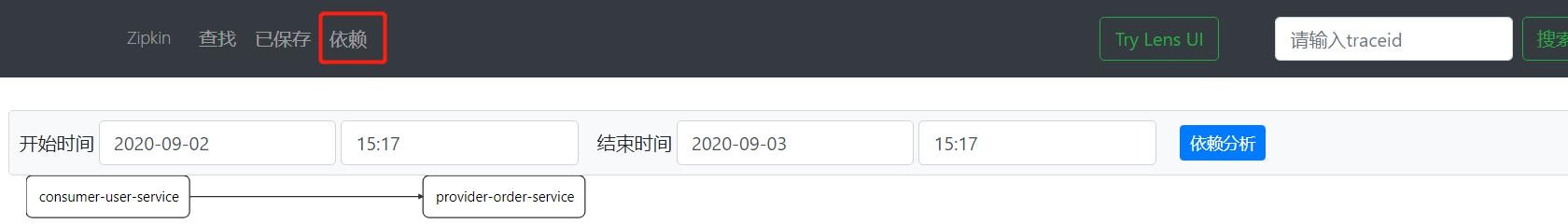
小数字为 0，最大值为 2,147,483,647（最大 int） 默认为 10。

## 启动访问，远程调用一下

查看 zipkin



点一下依赖



建议：远程调用的次数不要超过 **3** 层 **2** 层最好

补充：要监控服务咋办。用admin

对应的代码是05.admin-server

加依赖（不够就去源码找

<dependency>  
 <groupId>de.codecentric</groupId>  
 <artifactId>spring-boot-admin-starter-server</artifactId>  
</dependency>

配置

server:  
 port: 10086 *# 端口号范围 0-65535*spring:  
 application:  
 name: admin-server  
eureka:  
 client:  
 service-url:  
 defaultZone: http://192.168.10.100:8761/eureka  
 instance:  
 hostname: localhost  
 instance-id: ${eureka.instance.hostname}:${spring.application.name}:${server.port}  
management:  
 endpoints:  
 web:  
 exposure:  
 include: '\*' *# 暴露所有的监控端点 # 如果一个服务需要被监控 那么就要讲自身的一些情况(一些信息接口)暴露出去。服务本身没有暴露，那admin也监控不到*

启动就可以了

通过eureka拿到服务的ip和端口

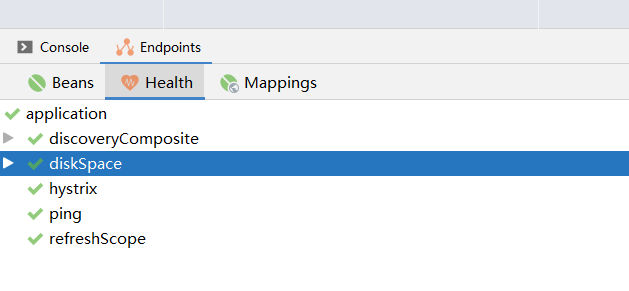
被监控服务可以通过以下来暴露一些检查端点

*<!-- 暴露自身检查端点 endPoints 一个依赖 -->*<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>

再加配置

management:  
 endpoints:  
 web:  
 exposure:  
 include: '\*'

加了后IDEA的有图形界面展示



数据暴露在<http://localhost:8081/actuator/beans>等路径了，要的话自己看