冰鉴科技 朱清

一年了，我们用SC干了啥

个人介绍

- 朱清，毕业于电子科技大学
- 现任冰鉴科技信息技术部总监，Spring cloud中国社区联合创始人
- 曾就职于腾讯视频
- 在冰鉴期间，设计并主导了冰鉴风控一体化平台和冰鉴信用大数据平台的研发，在互联网金融风控系统建设方面有着丰富的经验。研发成果中，拥有自建覆盖全国30+省IP代理、冰鉴大数据采集引擎的爬虫集群日处理上千万数据，并支持弹性扩容；冰鉴反欺诈引擎系国内首家支持任意数据来源、高度灵活配置的在线规则编辑引擎，同时包含了经过百万级贷后数据提炼的反欺诈和信用规则；冰鉴风险模型引擎支撑着冰鉴数据科学家团队研发的逻辑回归、机器学习等多种模型的动态部署；冰鉴决策引擎为冰鉴商务分析专家提供了动态信审规则的在线编辑、验证和灵活部署。

微服务构建框架

SPRING BOOT

SPRING BOOT 使配置变得简单

功能的组合配置

XML Conifg

setBean(Bean bean)

Java Conifg

@Autowire

部署环境的配置

多个 *.properties

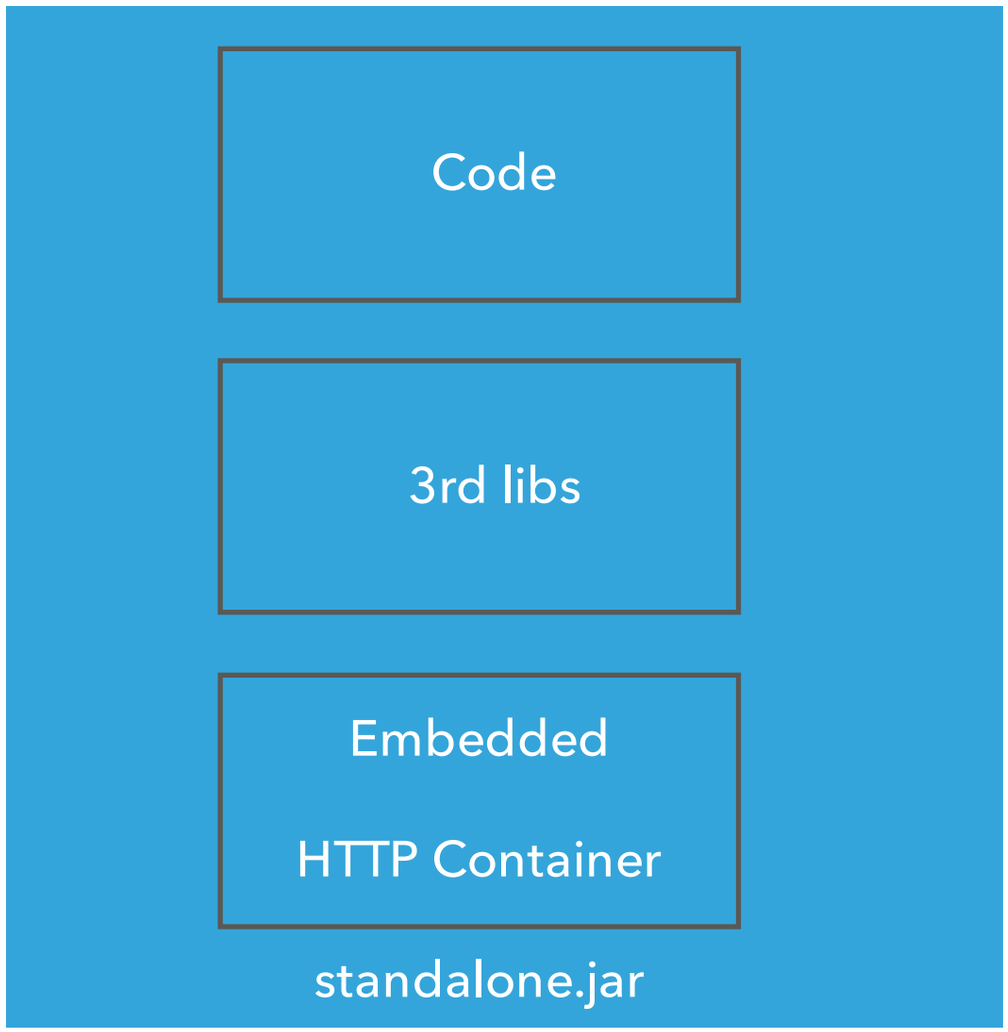
*.xml

application.yml

>>>

习惯优于配置！

SPRING BOOT 使部署变得简单



- 一键启动
- 不需要预部署应用服务器
- 降低对运行环境的基本要求： 有jdk即可，内嵌tomcat/jetty

SPRING BOOT 使监控变得简单

Spring boot actuator

主要暴露的功能

HTTP方法	路径	描述	鉴权
GET	/autoconfig	查看自动配置的使用情况	TRUE
GET	/configprops	查看配置属性，包括默认配置	TRUE
GET	/beans	查看bean及其关系列表	TRUE
GET	/dump	打印线程栈	TRUE
GET	/env	查看所有环境变量	TRUE
GET	/env/{name}	查看具体变量值	TRUE
GET	/health	查看应用健康指标	FALSE
GET	/info	查看应用信息	FALSE
GET	/mappings	查看所有url映射	TRUE
GET	/metrics	查看应用基本指标	TRUE
GET	/metrics/{name}	查看具体指标	TRUE
POST	/shutdown	关闭应用	TRUE

SPRING BOOT

组件脚手架

组件脚手架—常见集成

- MYSQL/ORACLE + MYBATIS
- MONGODB
- REDIS
- FASTDFS
- ROCKETMQ

组件脚手架—定制自己的STARTER

- 选择已有的starters，在此基础上进行扩展
- 创建自动配置文件并设定meta-inf/spring.factories里的内容
- 发布你的starters

业务系统开发人员可以直接引入最新的STARTER，进行开发，不用关心到底用什么组件！

SPRING BOOT

场景调用设计

场景设计—HTTP调用

（常见）A服务调用B服务，需要实时返回结果，调用方实时依赖执行结果的业务场景。

场景设计—异步消息

逻辑解耦 + 物理解耦的消息通信服务

适用场景：

- 数据驱动的任务依赖、用于传递任务执行完成的消息，并步用于传递输入输出数据。
- 上游不关心多下游执行结果
- 异步返回执行时间长

离微服务还差什么？

- 没有注册发现、授权等外围方案
- 没有统一的监控集成方案
- 配置还是跟应用关联在一起

>>>

只是一个微框架！

是构建微服务应用的基础框架！

微服务外围脚手架

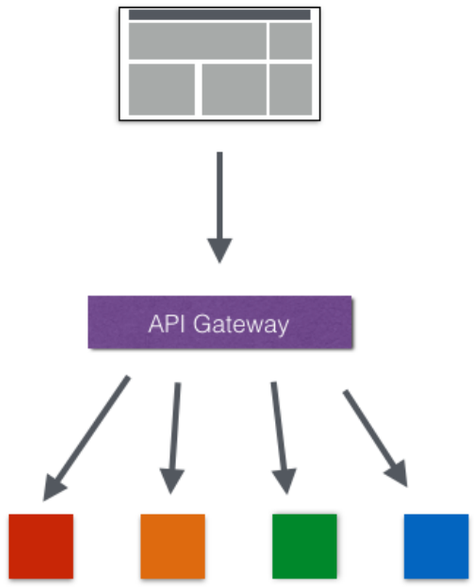
SPRING CLOUD

SPRING CLOUD

网关——ZUUL

网关

在微服务架构模式下后端服务的实例数一般是动态的，对于客户端而言很难发现动态改变的服务实例的访问地址信息。因此在基于微服务的项目中为了简化前端的调用逻辑，通常会引入API Gateway作为轻量级网关，同时API Gateway中也会实现相关的认证逻辑从而简化内部服务之间相互调用的复杂度。



网关

```
spring.application.name=gateway-service-zuul
server.port=8888

#这里的配置表示，访问/bd/** 直接重定向到http://www.baidu.com/**
zuul.routes.baidu.path=/bd/**
zuul.routes.baidu.url=http://www.baidu.com/

#在运用zuul作为服务网关进行反向代理时，服务名也是访问服务的默认路径
zuul.routes.api-a.path=/producer/**
zuul.routes.api-a.serviceId=spring-cloud-producer
```

网关

通过Filter，我们可以实现安全控制，比如，只有请求参数中有用户名和密码的客户端才能访问服务端的资源。

1、继承ZuulFilter类，实现对应的方法

2、filter类型和order

pre：可以在请求被路由之前调用

route：在路由请求时候被调用

post：在route和error过滤器之后被调用

error：处理请求时发生错误时被调用

SPRING CLOUD

服务注册与发现-CONSUL

服务注册与发现

定制服务名称：

要设置自动注册时采用的服务名，只需要给服务容器添加标签SERVICE_NAME=[服务名]即可，也可以通过设置环境变量SERVICE_NAME=[服务名]来达到同样的效果。

健康检查：

我们可以通过如下方式让Registrator在注册服务的同时，注册该服务的健康检查。以Http检查为例，设置标签或环境变量SERVICE_CHECK_HTTP=/health设置健康检查路径为/health；

设置标签或环境变量SERVICE_CHECK_INTERVAL=15s设置健康检查间隔为15秒；设置标签或环境变量SERVICE_CHECK_TIMEOUT=5s设置健康检查请求超时时间为5秒。

SPRING CLOUD

配置中心——CONFIG

配置中心

config-server 配置服务端，服务管理配置信息

config-client 客户端，客户端调用server端暴露接口获取配置信息

获取git上的资源信息遵循如下规则：

/{application}/{profile}[/{label}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties

SPRING CLOUD

熔断器——HYSTRIX

熔断器

熔断器功能的理解：在有限大小的线城池中执行某种操作（限流）；当操作超时时主动打断；操作失败或超时自动调用降级方法；一段时间内（默认10s）失败超过一定次数或比例时，在接下来的一定时间段内自动调用降级方法（熔断）。

- 在需要由熔断器管理的方法上添加注解（该方法所属的对象须由Spirng管理，不能和调用者为同一对象）
- 创建Fallback方法，调用失败或熔断时会调用Fallback，接收参数和正常方法相同
- groupKey相同的方法会共用线程池，fallbackMethod写降级方法的方法名
- 可以根据需要配置commandProperties和threadPoolProperties，更多配置请参考 https://github.com/Netflix/Hystrix/wiki/Configuration

SPRING CLOUD

服务调用——FEIGN

服务间调用

声明Feign接口

创建一个接口，添加注解 @FeignClient(serviceId = "被调用服务名")。

创建调用远端服务api的方法，用 @RequestMapping 注解为方法指定远程调用的路径（服务内部路径）、请求方法和参数。该方法应和被调用服务的Controller方法基本一致。

创建专用于调用服务的Component，注入Feign接口，Feign会自动实现该接口，请无视IDE警告。写需要调用远程服务的方法，添加HystrixCommand注解设定好熔断指标，并依需要写有逻辑的fallbak或固定错误响应。用法详见熔断器使用。

在Service实现中注入用于调用服务的Component，调用加Hystrix的方法。

整体调用逻辑为 Controller->Service->专用组件->Feign接口->远端服务api。

SPRING CLOUD

服务调用链追踪—SLEUTH

SLEUTH

针对服务化应用全链路追踪的问题，Google发表了Dapper论文，介绍了他们如何进行服务追踪分析。其基本思路是在服务调用的请求和响应中加入ID，标明上下游请求的关系。利用这些信息，可以可视化地分析服务调用链路和服务间的依赖关系。

对应Dpper的开源实现是Zipkin，支持多种语言包括JavaScript，Python，Java, Scala, Ruby, C#, Go等。其中Java由多种不同的库来支持。

Spring Cloud Sleuth是对Zipkin的一个封装，对于Span、Trace等信息的生成、接入HTTP Request，以及向Zipkin Server发送采集信息等全部自动完成。

此外我们同时针对mq消息传递中对sleuth的集成。

```
2017-10-07 22:54:29.526+0800 INFO  [MSexample-0.0.1, ed6e863365896d02,ed6e863365896d02,false] [0.0-8080-exec-1] : receive hello
```

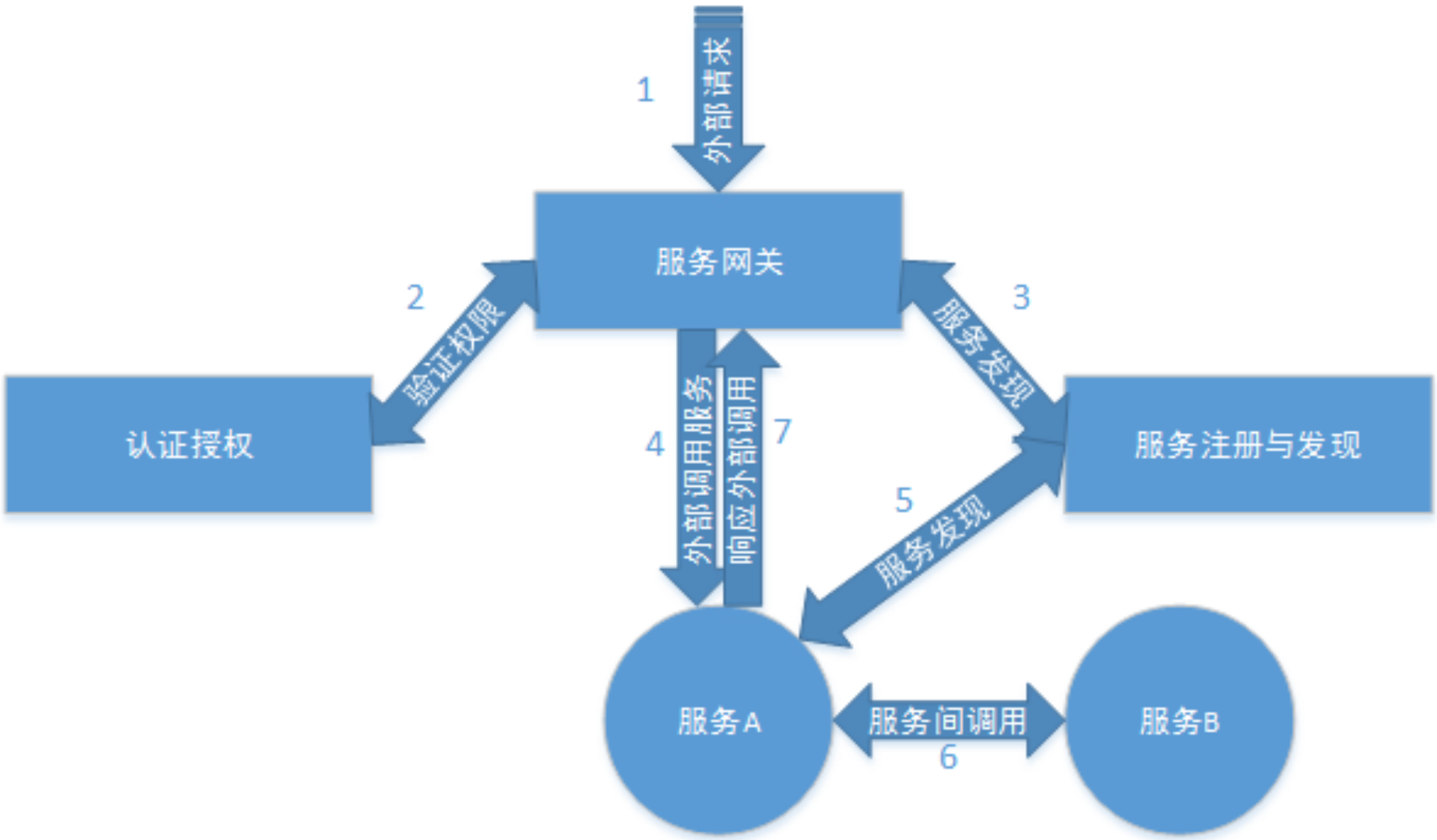
ZIPKIN

Twitter的zipkin是一个致力于收集Twitter所有的分布式服务的时间数据的分布式跟踪系统。它提供了收集数据，和查询数据两大服务。系统的理论模型来自于Google Dapper 论文。通过采集跟踪数据可以帮助开发者深入了解在分布式系统中某一个特定的请求时如何执行的。

>>>

由于使用elk检索日志，sleuth帮助记录了traceid，故觉得有点鸡肋，直接采用了检索traceid

服务访问结构


1
外部请求
服务网关
2
验证权限
认证授权
3
服务发现
服务注册与发现
4
外部调用服务
7
响应外部调用
5
服务发现
服务A
服务间调用
6
服务B

其他

运维与测试能力是积淀

冰鉴科技 朱清

欢迎关注我的个人公众号

谢谢