

目 录

1. 欢迎来到 Cloud 和 Spring	1
1.1 什么是微服务？	2
1.2 什么是 Spring，它和微服务有什么关联？	5
1.3 你将从这本书里学到什么？	6
1.4 为什么这本书与你有关？	7
1.5 使用 Spring Boot 构建一个微服务	8
1.6 为什么要改变我们构建应用程序的方式？	12
1.7 云究竟是什么？	13
1.8 为什么是云和微服务？	15
1.9 微服务不仅仅是编写代码	19
1.9.1 微服务核心开发模式	19
1.9.2 微服务路由模式	20
1.9.3 微服务客户端弹性模式	21
1.9.4 微服务安全模式	23
1.9.5 微服务日志记录和跟踪模式	24
1.9.6 微服务构建和部署模式	25
1.10 使用 Spring Cloud 构建微服务	26
1.10.1 Spring Boot	28
1.10.2 Spring Cloud Config	28
1.10.3 Spring Cloud Service Discovery	28
1.10.4 Spring Cloud/Netflix Hystrix and Ribbon	29

1.10.5 Spring Cloud/Netflix Zuul	29
1.10.6 Spring Cloud Stream	29
1.10.7 Spring Cloud Sleuth	29
1.10.8 Spring Cloud Security	30
1.10.9 准备些什么？	30
1.11 使用 Spring Cloud 举例	30
1.12 了解与我们相关的例子	33
1.13 小结	33
2. 使用 Spring Boot 构建微服务	35
2.1 架构师的故事：设计微服务架构	38
2.1.1 分解业务问题	38
2.1.2 确定服务粒度	41
2.1.3 互相交谈：服务接口	43
2.2 什么时候不使用微服务	44
2.2.1 分布式系统构建的复杂性	44
2.2.2 服务器扩展	44
2.2.3 应用类型	44
2.2.4 数据转换和一致性	45
2.3 开发：使用 Spring Boot 和 Java 构建一个微服务	45
2.3.1 从骨架项目快速入门	46
2.3.2 编写引导类,启动你的 Spring Boot 应用	47
2.3.3 创建微服务的访问入口：Spring Boot 控制器	48

2.4 运维：严格的运行时构建	53
2.4.1 服务装配：打包和部署微服务	56
2.4.2 服务引导：管理微服务的配置	58
2.4.3 服务注册与发现：客户端如何与微服务通信	59
2.4.4 监控微服务健康状况	60
2.5 观点汇总	62
2.6 小结	63
3. 使用 <i>Spring Cloud Configuration Server</i> 控制配置	64
3.1 管理配置（和复杂性）	65
3.1.1 配置管理架构	67
3.1.2 实现选择	69
3.2 建立 Spring Cloud Config Server	70
3.2.1 创建 Spring Cloud 配置引导类	74
3.2.2 通过配置文件使用 Spring Cloud Config Server	75
3.3 将 Spring Cloud 配置与 Spring Boot 客户端集成	77
3.3.1 配置 Spring Cloud Config Server 的依赖	79
3.3.2 Spring Cloud Config 使用配置	79
3.3.3 使用 Spring Cloud Configuration Server 配置数据源	83
3.3.4 使用@Value 注解直接读取属性	86
3.3.5 通过 Git 使用 Spring Cloud Config Server	87
3.3.6 通过 Spring Cloud Config Server 刷新属性	88
3.4 保护敏感配置信息	89

3.4.1 下载和安装加密所需的 Oracle JCE JAR 包	90
3.4.2 配置加密密钥	91
3.4.3 加密和解密属性	91
3.4.4 配置微服务使用加密的客户端	93
3.5 最后的思考	95
3.6 小结	95
4. 服务发现	96
4.1 我的服务在哪里？	97
4.2 关于云端服务发现的研究	100
4.2.1 服务发现架构	100
4.2.2 使用 Spring 和 Netflix Eureka 实现服务发现	103
4.3 创建 Spring Eureka Service	105
4.4 使用 Spring Eureka 注册服务	107
4.5 使用服务发现查找服务	111
4.5.1 使用 Spring DiscoveryClient 查找服务实例	112
4.5.2 使用 Ribbon-aware Spring RestTemplate 调用服务	114
4.5.3 使用 Netflix Feign client 调用服务	116
4.6 小结	118
5. 坏事发生时 使用 Spring Cloud 和 Netflix Hystrix 的客户端弹性模式	119
5.1 客户端的弹性模式	120
5.1.1 客户端负载均衡	121
5.1.2 断路器	122

5.1.3 回退处理	122
5.1.4 舱壁	122
5.2 客户端弹性的重要性	123
5.3 熔断机制	126
5.4 配置使用 Spring Cloud 和 Hystrix 的依赖	127
5.5 使用 Hystrix 实现断路器	128
5.5.1 调用微服务超时	131
5.5.2 自定义断路器上的超时时间	132
5.6 回退处理	133
5.7 舱壁模式的实现	136
5.8 深入理解 Hystrix	138
5.8.1 进一步理解 Hystrix 配置	142
5.9 线程上下文和 Hystrix	144
5.9.1 ThreadLocal 和 Hystrix	144
5.9.2 Hystrix 并发策略	147
5.10 小结	151
6. 使用 Spring Cloud 和 Zuul 进行服务路由	153
6.1 服务网关	154
6.2 Spring Cloud 和 Netflix Zuul	157
6.2.1 配置 Spring Boot 工程引用 Zuul 依赖	157
6.2.2 使用 SpringCloud 注解配置 Zuul 服务	157
6.2.3 配置 Zuul 与 Eureka 通信	158

6.3 在 Zuul 中配置路由	159
6.3.1 通过服务发现自动映射路由	159
6.3.2 通过服务发现手动映射路由	161
6.3.3 使用静态 URL 手动映射路由	165
6.3.4 动态重新加载路由配置	168
6.3.5 Zuul 和服务超时	169
6.4 Zuul 过滤器	169
6.5 创建 pre 类型 Zuul 过滤器生成关联 ID	173
6.5.1 在服务调用中使用关联 ID	176
6.6 创建 post 类型 Zuul 过滤器接收关联 ID	182
6.7 创建动态路由过滤器	184
6.7.1 构建路由过滤器的框架	186
6.7.2 实现 run 方法	187
6.7.3 转发路由	188
6.7.4 把代码整合在一起	190
6.8 小结	191
7. 微服务安全	192
7.1 介绍 OAuth2	193
7.2 从小事做起：使用 Spring 和 OAuth2 保护一个单独的端点	195
7.2.1 配置 EagleEye OAuth2 认证服务	196
7.2.2 在 OAuth2 服务注册客户端应用	197
7.2.3 配置 EagleEye 用户	200

7.2.4 用户认证	202
7.3 使用 OAuth2 保护组织服务	205
7.3.1 为单独的服务添加 Spring Security 和 OAuth2 JAR 包	205
7.3.2 配置服务指向 OAuth2 认证服务	206
7.3.3 定义那些资源可以访问服务	207
7.3.4 传递 OAuth2 访问令牌	210
7.4 JavaScript Web Tokens 和 OAuth2	213
7.4.1 修改认证服务来发布 JavaScript Web Tokens	214
7.4.2 在微服务中消费 JavaScript Web Tokens	218
7.4.3 扩展 JWT Token	220
7.4.4 解析来自自定义字段的 JavaScript token	222
7.5 关于微服务安全的思考	224
7.6 小结	227
8. Spring Cloud Stream 的事件驱动架构	228
8.1 消息，EDA 和微服务的案例	229
8.1.1 使用同步请求响应的方法传递状态改变	230
8.1.2 在服务之间使用消息传递状态改变	233
8.1.3 消息架构的缺点	235
8.2 介绍 Spring Cloud Stream	236
8.2.1 Spring Cloud Stream 体系结构	237
8.3 写一个简单的消息生产者和消费者	238
8.3.1 在组织服务中编写消息生产者	239

8.3.2 在许可服务中编写消息消费者	244
8.3.3 查看运行中的消息服务	247
8.4 Spring Cloud Stream 使用案例：分布式缓存	249
8.4.1 使用 Redis 缓存查找	250
8.4.2 定义自定义 channels	256
8.4.3 将它们放在一起：收到消息时清除缓存	257
8.5 小结	258
9. 使用 <i>Spring Cloud Sleuth</i> 和 <i>Zipkin</i> 进行分布式跟踪	259
9.1 Spring Cloud Sleuth 和关联 ID	260
9.1.1 添加 Spring Cloud Sleuth	261
9.1.2 分析 Spring Cloud Sleuth Trace	262
9.2 日志聚合和 Spring Cloud Sleuth	263
9.2.1 实战中的 Spring Cloud Sleuth/Papertrail 实现	265
9.2.2 创建一个 Papertrail 账户并配置 syslog 连接器	267
9.2.3 将 Docker 输出重定向到 Papertrail	268
9.2.4 在 Papertrail 里搜索 Spring Cloud Sleuth Trace ID	270
9.2.5 使用 Zuul 将关联 ID 添加到 HTTP 响应中	272
9.3 使用 Zipkin 实现分布式跟踪	274
9.3.1 配置 Spring Cloud Sleuth 和 Zipkin 依赖	275
9.3.2 配置服务指向 Zipkin	275
9.3.3 安装和配置 Zipkin Server	276
9.3.4 设置跟踪级别	278

9.3.5 使用 Zipkin 跟踪交易	278
9.3.6 可视化更复杂的交易	281
9.3.7 捕获消息跟踪	282
9.3.8 添加自定义 span	284
9.4 小结	287
10. 部署微服务	288
10.1 EagleEye : 在云环境中设置核心基础设施	290
10.1.1 使用 Amazon RDS 创建 PostgreSQL 数据库	293
10.1.2 在 Amazon 创建 Redis 集群	296
10.1.3 创建 ECS (弹性可伸缩的计算服务) 集群	298
10.2 除了基础设施 : 部署 EagleEye	302
10.2.1 在 ECS 中手动部署 EagleEye 服务	303
10.3 构建/部署管道的架构	305
10.4 构建和部署管道	309
10.5 开始你的构建/部署管道 : GitHub 和 Travis CI	311
10.6 在 Travis CI 中构建服务	312
10.6.1 核心构建运行时配置	315
10.6.2 安装预构建工具	318
10.6.3 执行构建	320
10.6.4 标记源代码控制代码	320
10.6.5 构建微服务和创建 Docker 镜像	321
10.6.6 将镜像发布到 Docker Hub	322

10.6.7 在 Amazon ECS 环境中启动服务	323
10.6.8 开展平台测试	323
10.7 关于构建/部署管道的总结	325
10.8 小结	325
附录 A : 在桌面上运行云	327
附录 B : OAuth2 授权类型	336