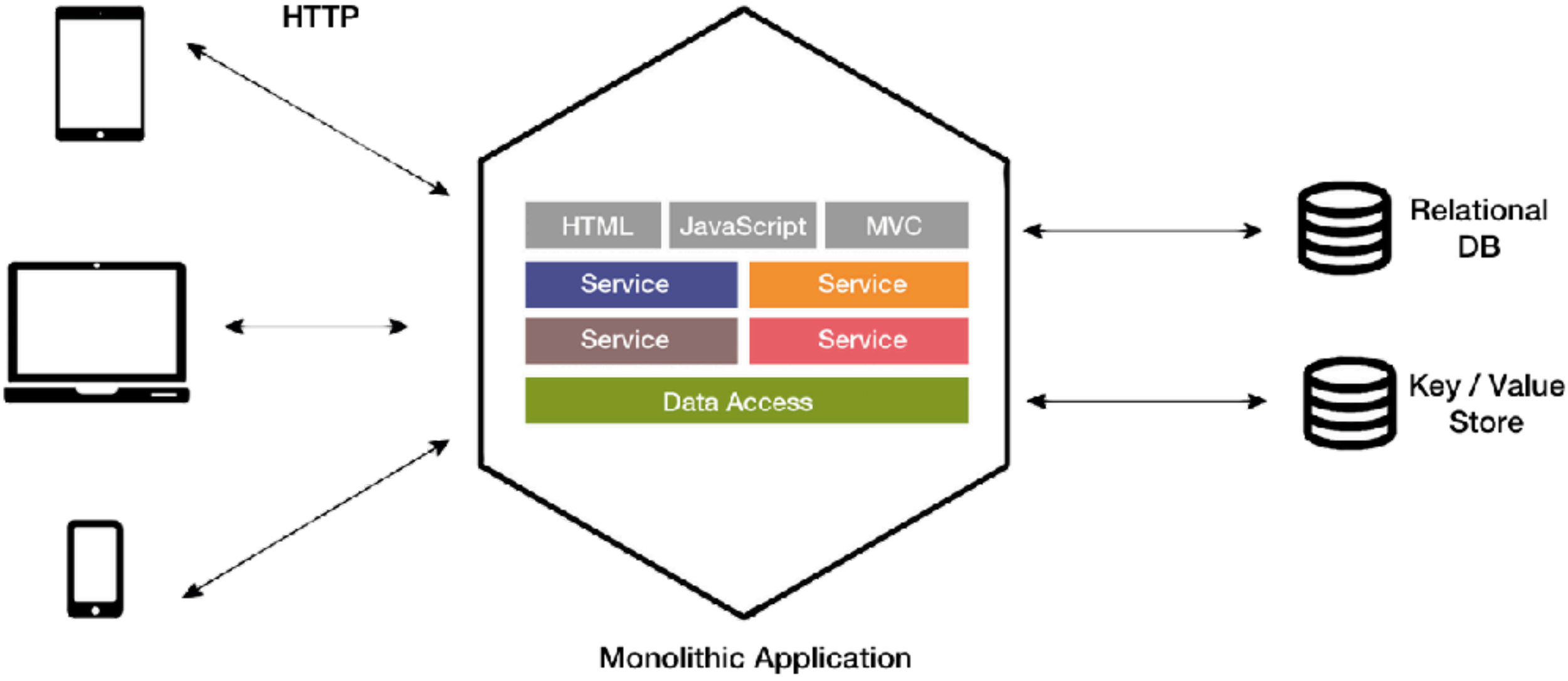
基于Spring-Cloud搭建Microservices架构

—轻量级分布式系统

单体应用架构图


HTTP
HTML
JavaScript
MVC
Service
Service
Service
Service
Data Access
Relational DB
Key / Value Store
Monolithic Application

单体应用优点

- 一套代码库
- 开发测试部署简单，IDE友好支持
- 容易横向扩展，通过增加应用服务器数量
- 适合团队集中式开发

单体应用缺点

- 代码庞大臃肿，对新来开发者不友好
- 开发部署周期长
- 语言、技术单一
- 模块之间耦合性强，扩展差
- 启动费时，占用系统资源
- 牵一发而动全身

什么是微服务

- 轻量级的分布式系统架构
- 单一职责
- 每个微服务都是一个完整的应用，独立开发和部署
- 轻量级的通讯协议，Rest API
- 对语言、技术框架没有限制
- 快速开发，快速交付，持续分发
- 有效的横向扩展，按需扩展
- 新来开发人员容易快速理解掌握业务

微服务的不足

- 分布式系统的固有复杂性
- 系统间集成测试复杂性
- 分布式事务的一致性

IOS

Web UI

Anroid

API GateWay

服务注册中心

统一配置中心

Git-repo

Microservice1

Microservice2

Microservice3

Message Broker

SQL DB

NOSQL DB

Document DB

微服务框架

- Dubbo
- Java EE
- Thrift
- Spring Cloud (推荐)

Spring Cloud简介

- 构建在Spring Boot框架之上,遵循12-Factor
- Netflix、Consul、Zookeeper等开源第三方Microservices框架集成进Spring Boot框架中
- 基于约定大于配置思想，开箱即用
- 搭建Cloud Native，也可部署到第三方云计算平台

Spring Cloud 主要特性

- 分布式 / 版本化 统一配置
- 服务注册与发现
- 路由
- 服务调用
- 负载均衡
- 熔断器与故障容错
- 分布式锁
- 分布式消息代理

Netflix OSS

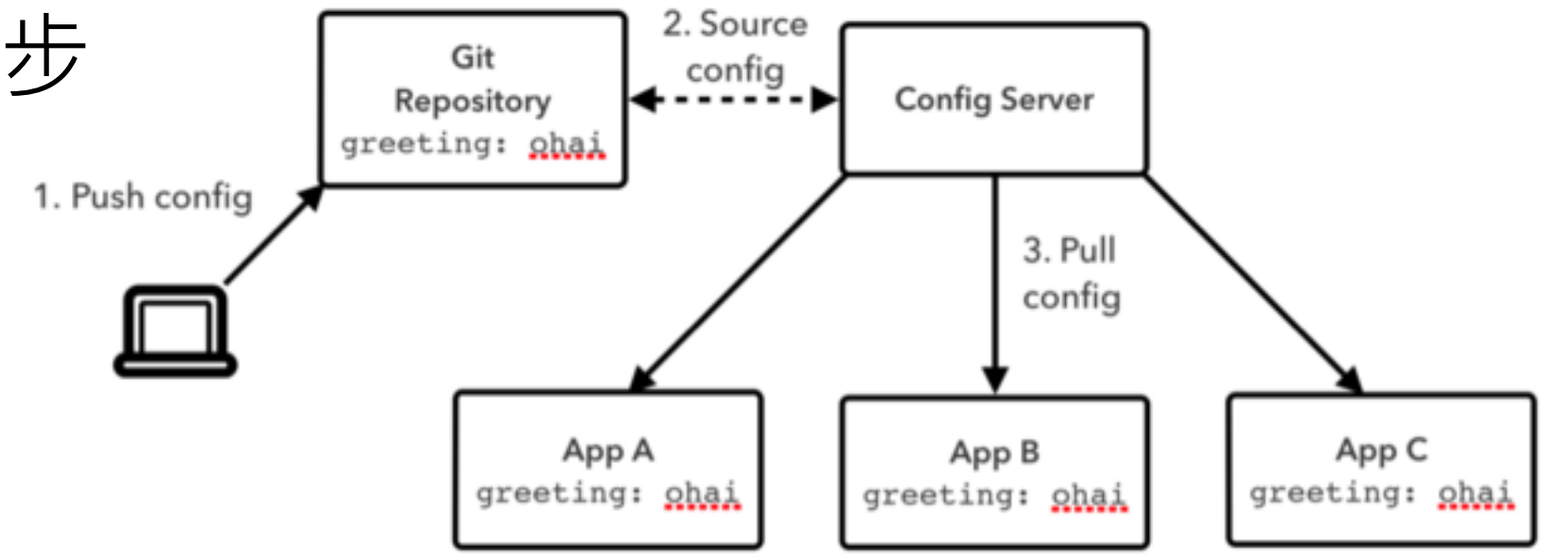
- Eureka
- Zuul
- Hystrix & Tubine
- Ribbon
- Feign
- Archaius

Spring Cloud Config

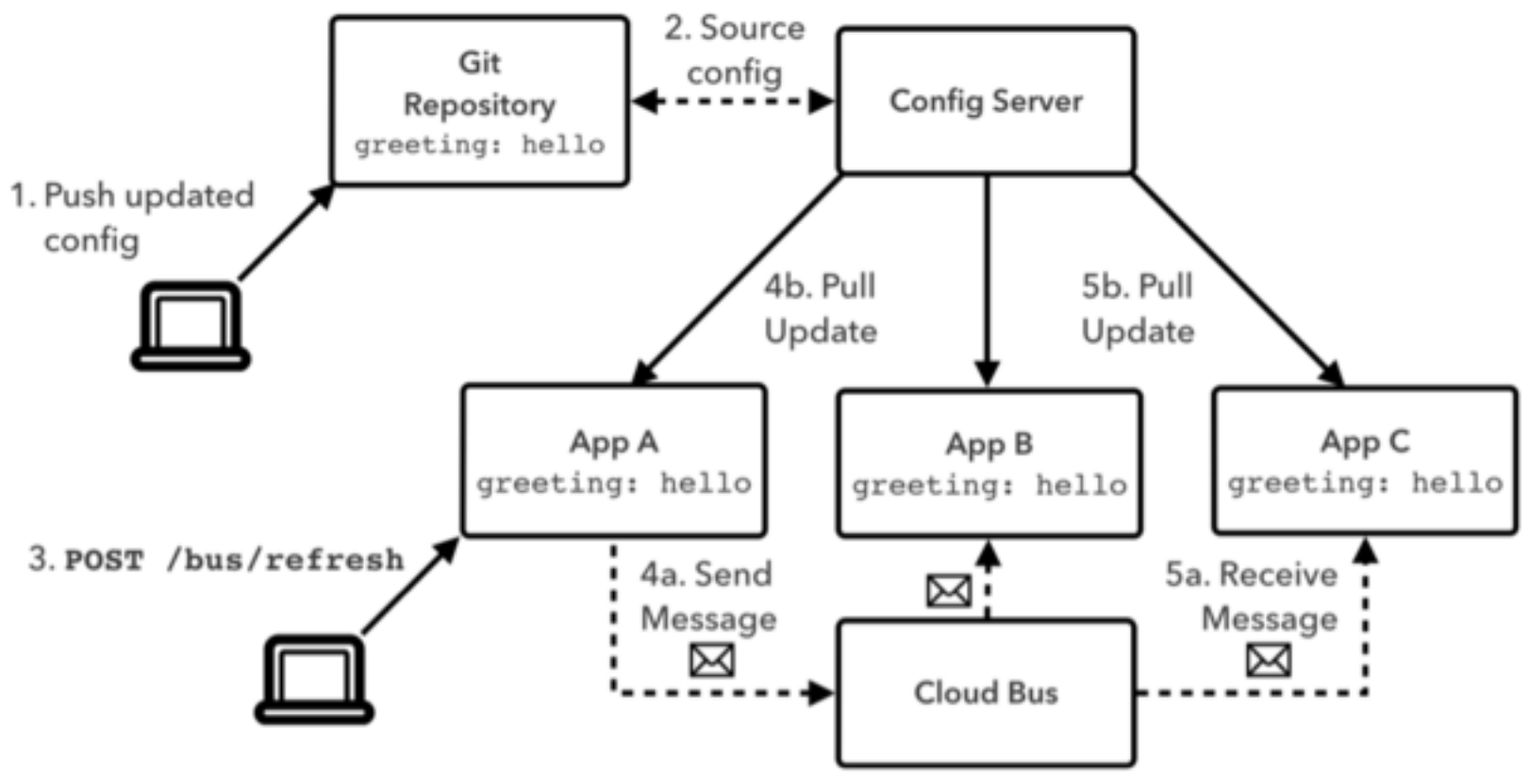
- Yaml \ Properties, 客户端自动绑定
- 基于Git 实现
- Version 控制
- 运行时修改配置信息
- 提供 REST API
- 配置信息加密 / 解密

Config 同步机制

单节点同步



多节点同步



服务注册&发现

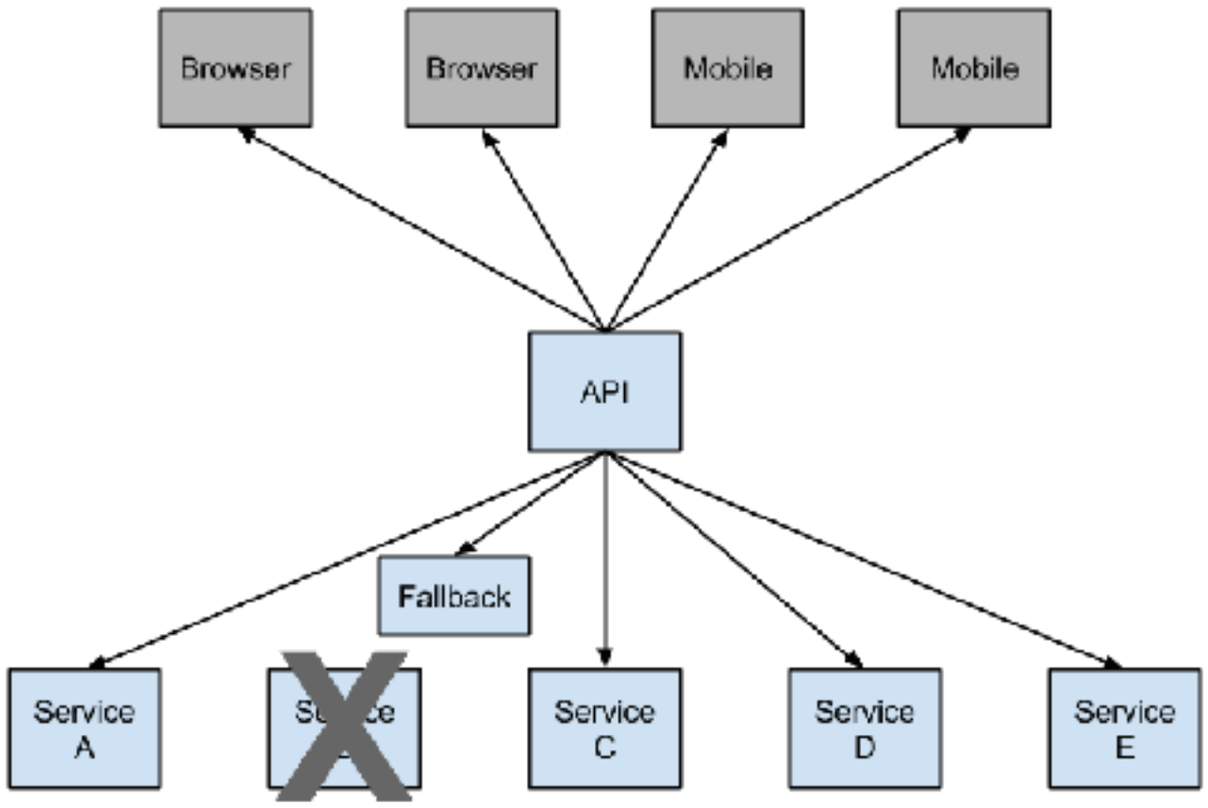
- Spring Cloud 提供三种实现方式
  - ★ Eureka （推荐）
  - ★ Consul
  - ★ Zookeeper
- Eureka特点
  - ★ 提供Rest API，跨语言
  - ★ 高可靠性
  - ★ 支持多节点

Microservices 服务间通讯

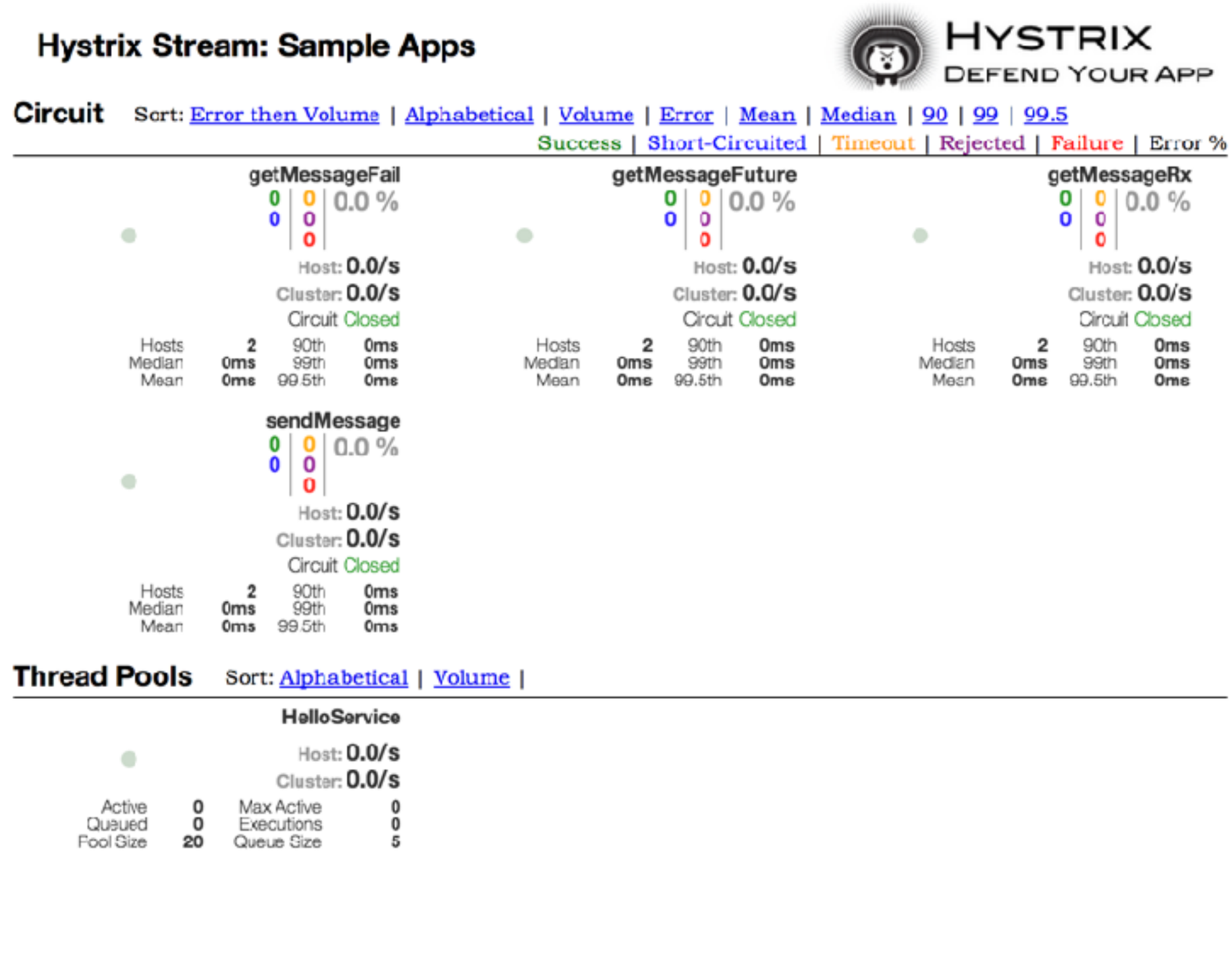
- Http请求响应—即时通讯模式
  - ✦ 声明式Rest Client — Feign (推荐)
  - ✦ Spring RestTemplate
  - ✦ 其他第三方Http Client库 — Okhttp、HttpClient等
- 基于事件驱动的消息通知— 异步通知模式（推荐）
  - ✦ 增大系统吞吐量、并发数，提高系统性能
  - ✦ Spring Cloud Stream，简化实现方式
  - ✦ Message Broker — Kafaka 、RabbitMQ

Circuit Breaker:Hystrix

- Fast Fail， 不要 Retry
- 延迟和故障容错
- 隔离防问
- 避免级联效应
- Hystrix Dashboard 健康检查



Hystrix仪表盘


Hystrix Stream: Sample Apps
HYSTRIX
DEFEND YOUR APP
Circuit
Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5
Success | Short-Circuited | Timeout | Rejected | Failure | Error %
getMessageFail
0 0 0 0 0 0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms
getMessageFuture
0 0 0 0 0 0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms
getMessageRx
0 0 0 0 0 0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms
sendMessage
0 0 0 0 0 0.0 %
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts 2 90th 0ms
Median 0ms 99th 0ms
Mean 0ms 99.5th 0ms
Thread Pools
Sort: Alphabetical | Volume |
HelloService
Host: 0.0/s
Cluster: 0.0/s
Active 0
Queued 0
Pool Size 20
Max Active 0
Executions 0
Queue Size 5

路由：Zuul

- 基于JVM的路由和负载均衡服务端实现
- 类似Nginx、httpd的路由角色
- 丰富的Router 规则，通过Spring Boot配置文件定制
- Router自定义扩展，实现自己的Router
- 自定义Filter，如增加访问控制和安全权限控制

客户端负载均衡：Ribbon

- 实现客户端http / tcp请求的负载均衡
- 多种负载均衡策略，简单轮询、加权响应、随机等
- Feign底层已默认实现Ribbon
- RestTemplate 添加注解 @LoadBalancer

未讲主题

- 微服务trace
- 微服务的安全问题
- 微服务的一致性
- 微服务的可视化监控
- 微服务的Docker部署

Demo