



跟我做一个Java微服务项目

Week #3 微服务间关系 / 刘俊强



欢迎关注StuQ公众号



欢迎关注我的微信公众号

大纲

- 微服务注册与发现
- 服务调用与通信
- API 网关

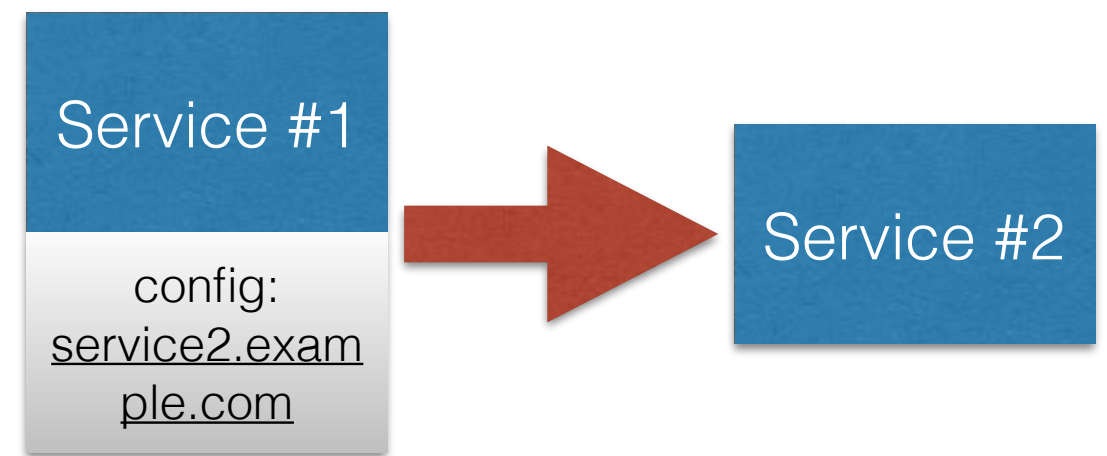
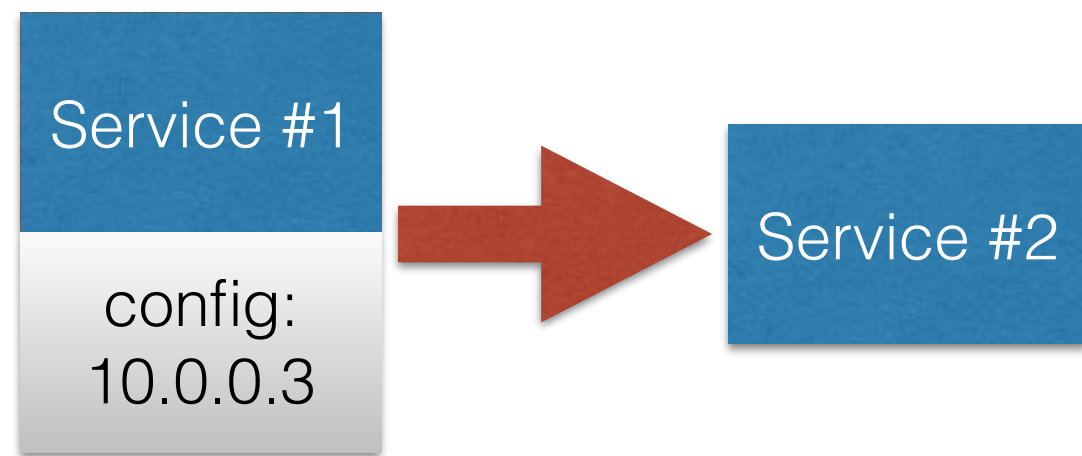
Terms 术语约定

- Service Registry 服务注册表
- Service Discovery 服务发现
- Service Registration 服务注册
- Service De-registration 服务注销
- API Gateway API 网关

服务注册与发现

传统方法

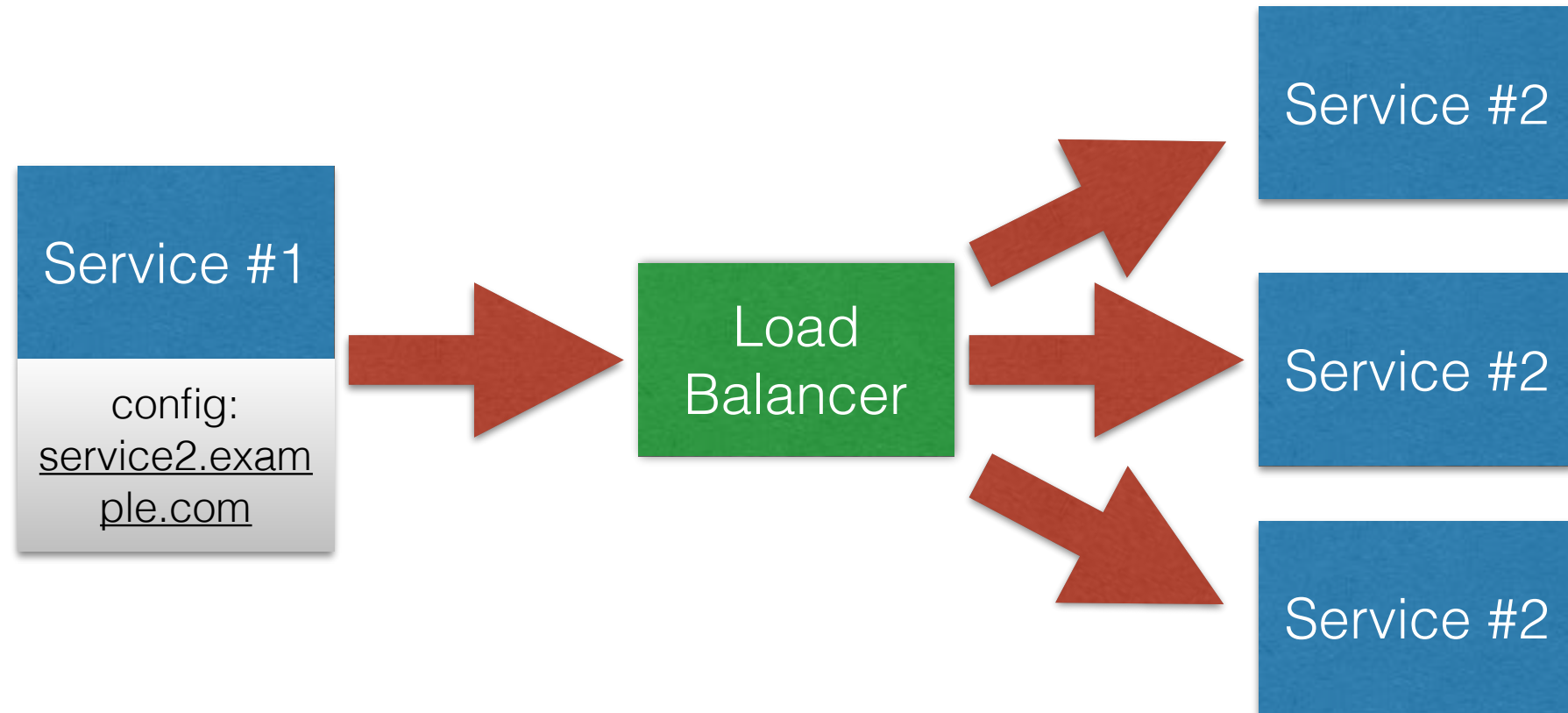
硬编码IP 或 DNS查询



DNS查询使用简单
需要管理域名配置文件
如何处理故障？

处理故障

将DNS解析指向负载均衡器



如何检查服务是否健康?
如何注册服务?

为何需要服务注册表

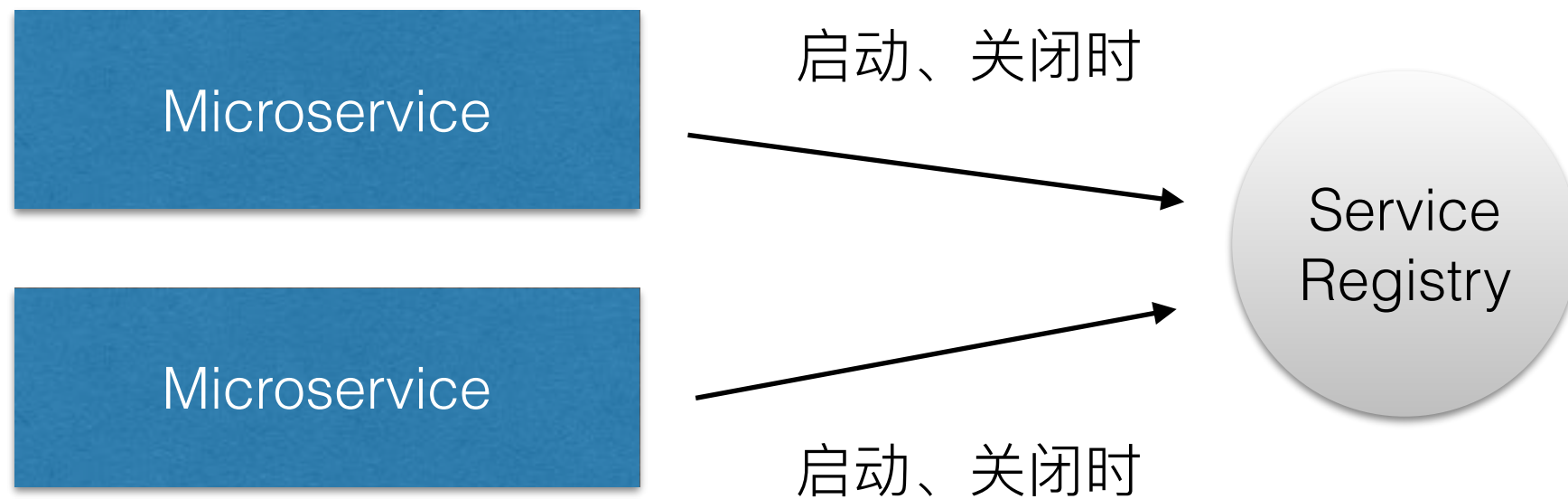
- 服务注册
- 健康检测
- 服务发现
- 服务注销

服务注册

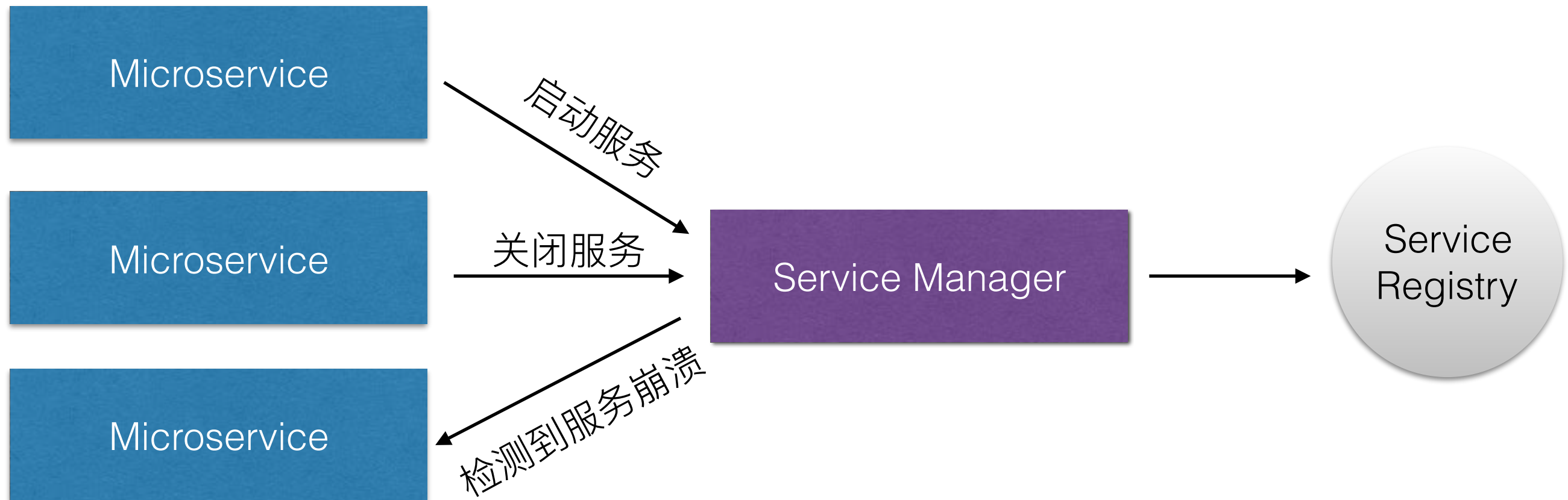
服务注册方式

- 自注册 Self-registration
- 第三方注册 Third-party registration

Self-registration



Third-party registration

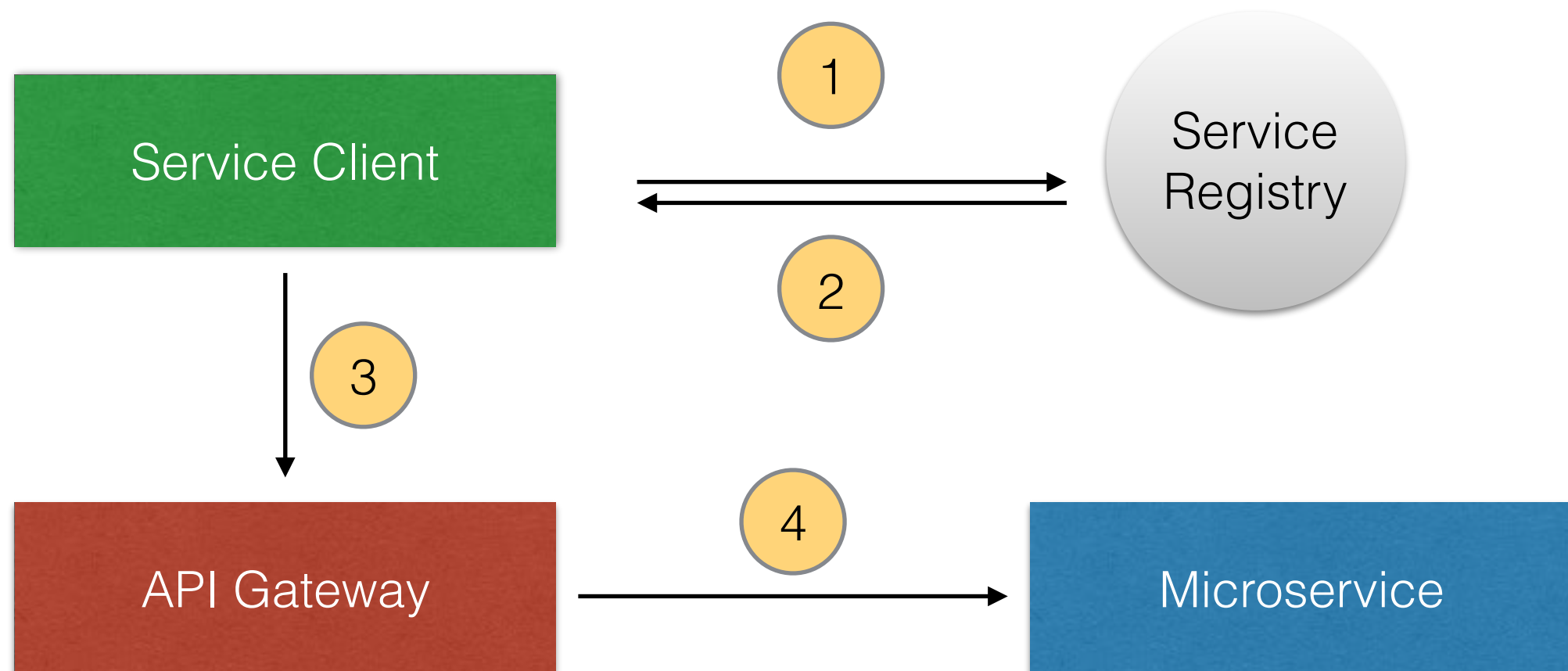


服务发现

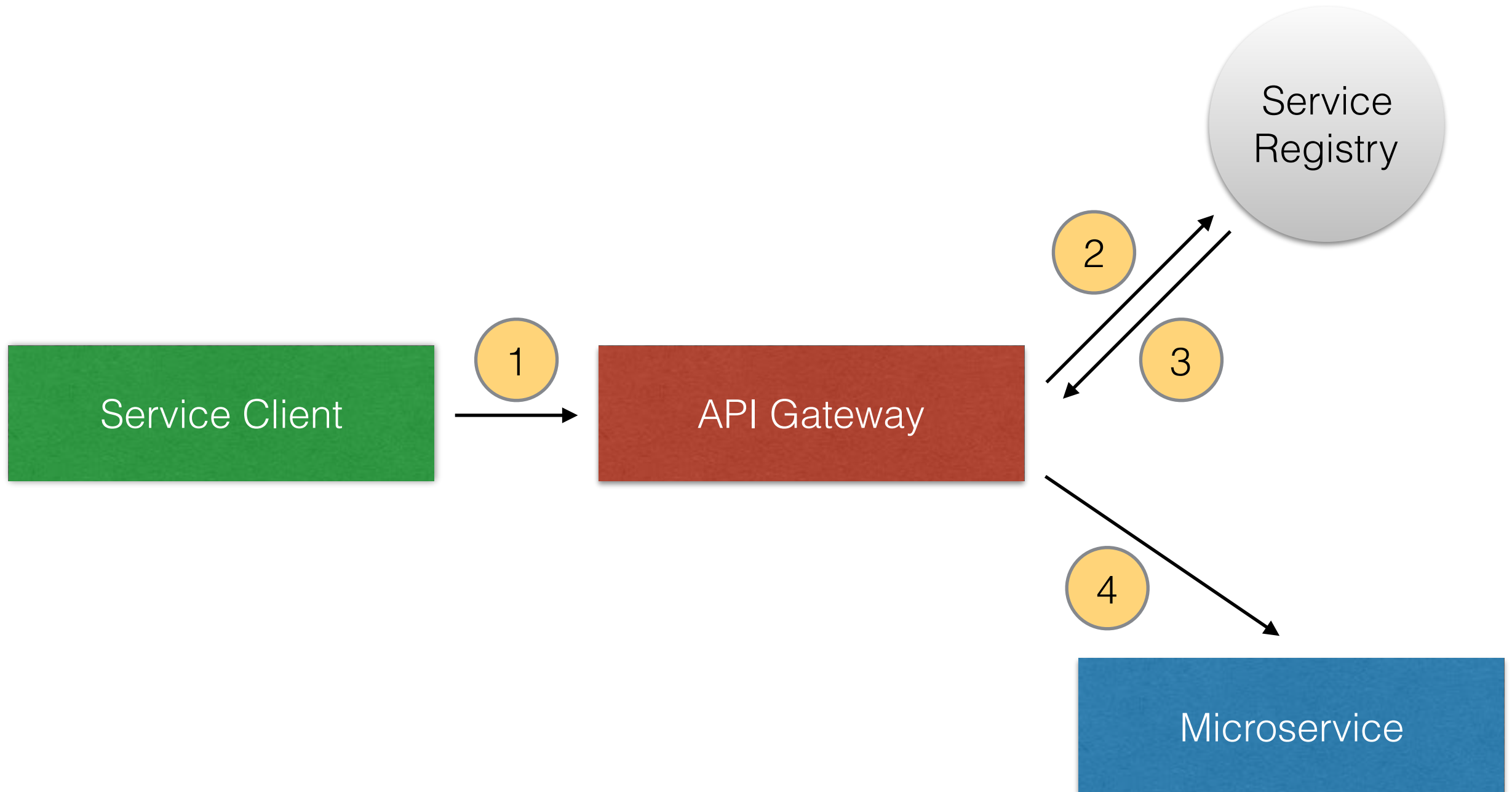
服务发现方式

- Client-side discovery
- Server-side discovery

Client-side discovery







Server-side discovery



常用“服务注册表”

常用服务注册表 Service Registry

- Zookeeper <https://zookeeper.apache.org/> 
- etcd <https://github.com/coreos/etcd>  etcd
- Hashicorp Consul <https://www.consul.io/> 
- Eureka <https://github.com/Netflix/eureka> 

Consul

- 服务注册与发现
- 健康检测
- 配置管理
- 跨机房支持
- DNS、REST接口支持

Consul DEMO

Eureka

- 服务注册与发现
- 基于Java语言
- 跨机房支持
- REST接口支持

Eureka DEMO

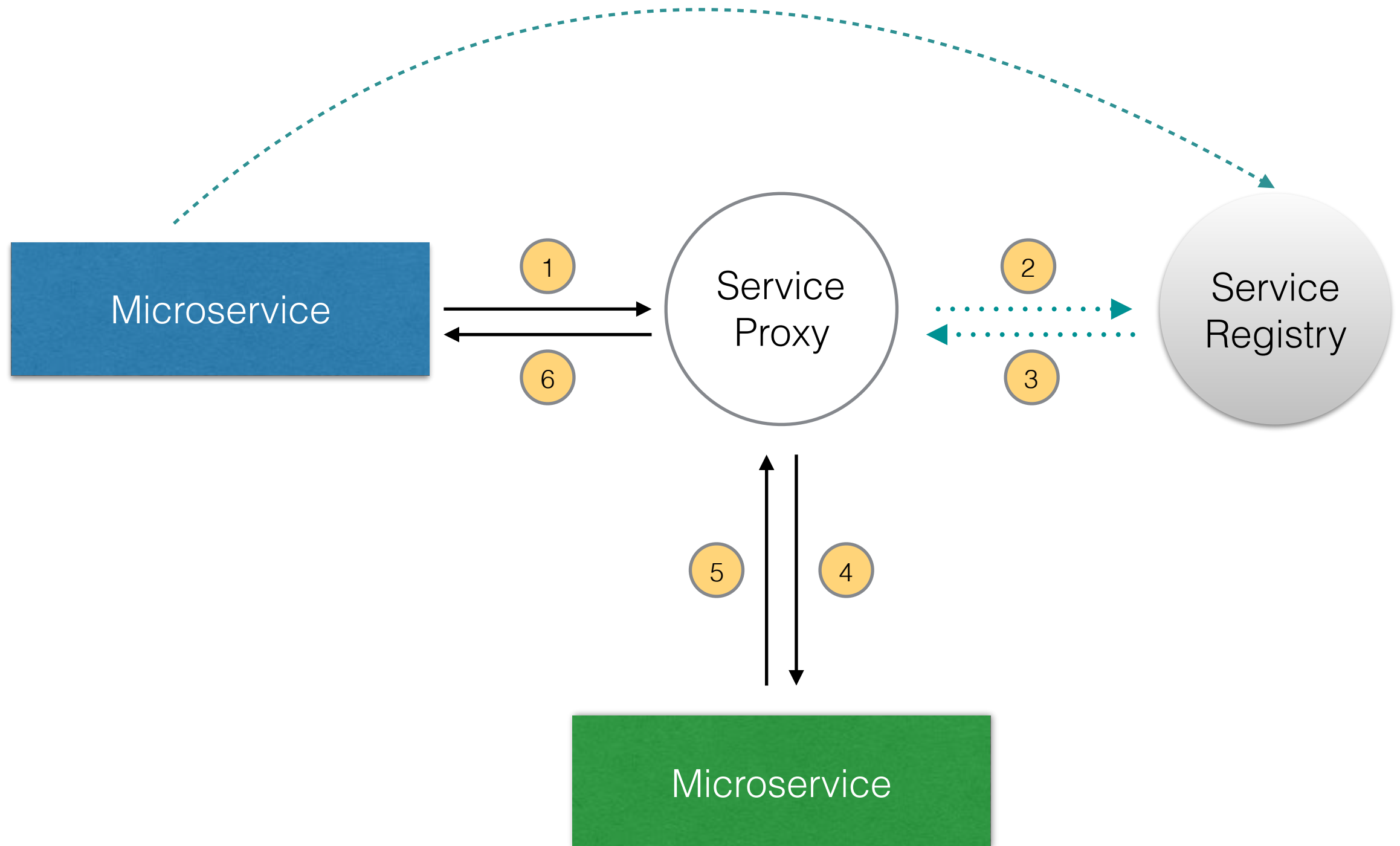
服务调用与通信

服务间调用

服务间调用方式

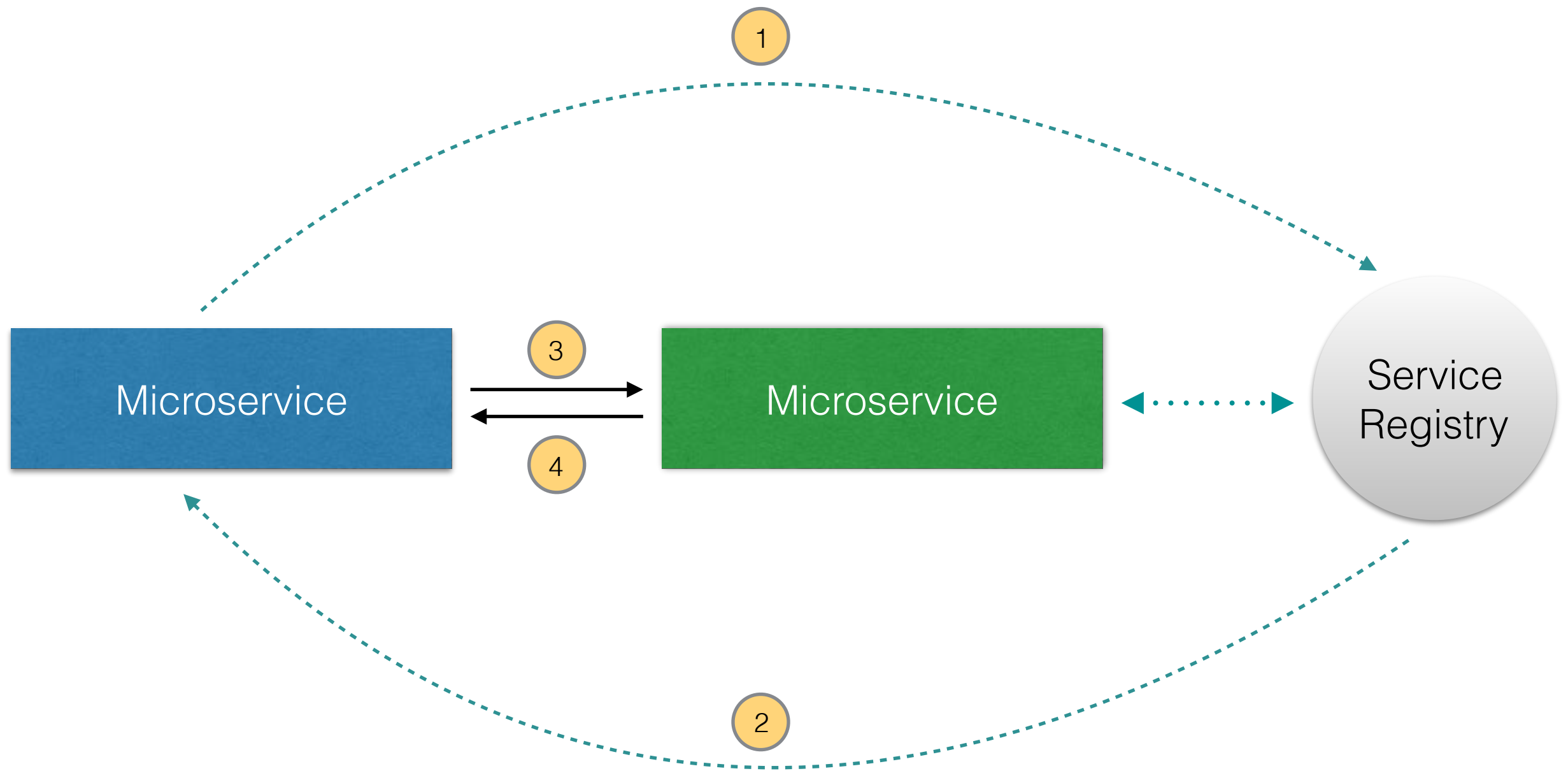
- Server-side 调用
- Client-side 调用

Server-side 调用



..... Registry requests
—— API Requests

Client-side 调用



..... Registry requests
—— API Requests

服务间通信

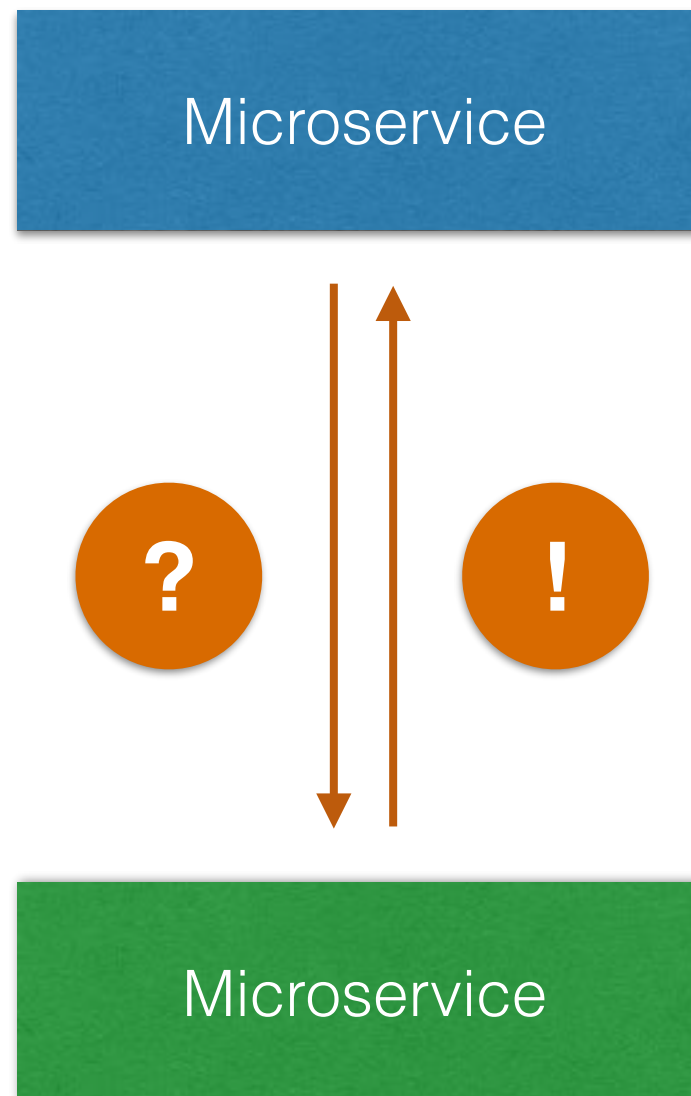
服务间通信方式

	1对1	1对多
同步	请求/响应	—
异步	通知	发布/订阅
	请求/异步响应	发布/异步响应

基于HTTP的同步方式

- HTTP REST
- 简单，“请求/响应”模式
- 防火墙友好，跨网调用方便
- 不支持“发布/订阅”模式
- 调用方、被调用方得同时在线
- 调用方得知道被调用方的主机名和端口

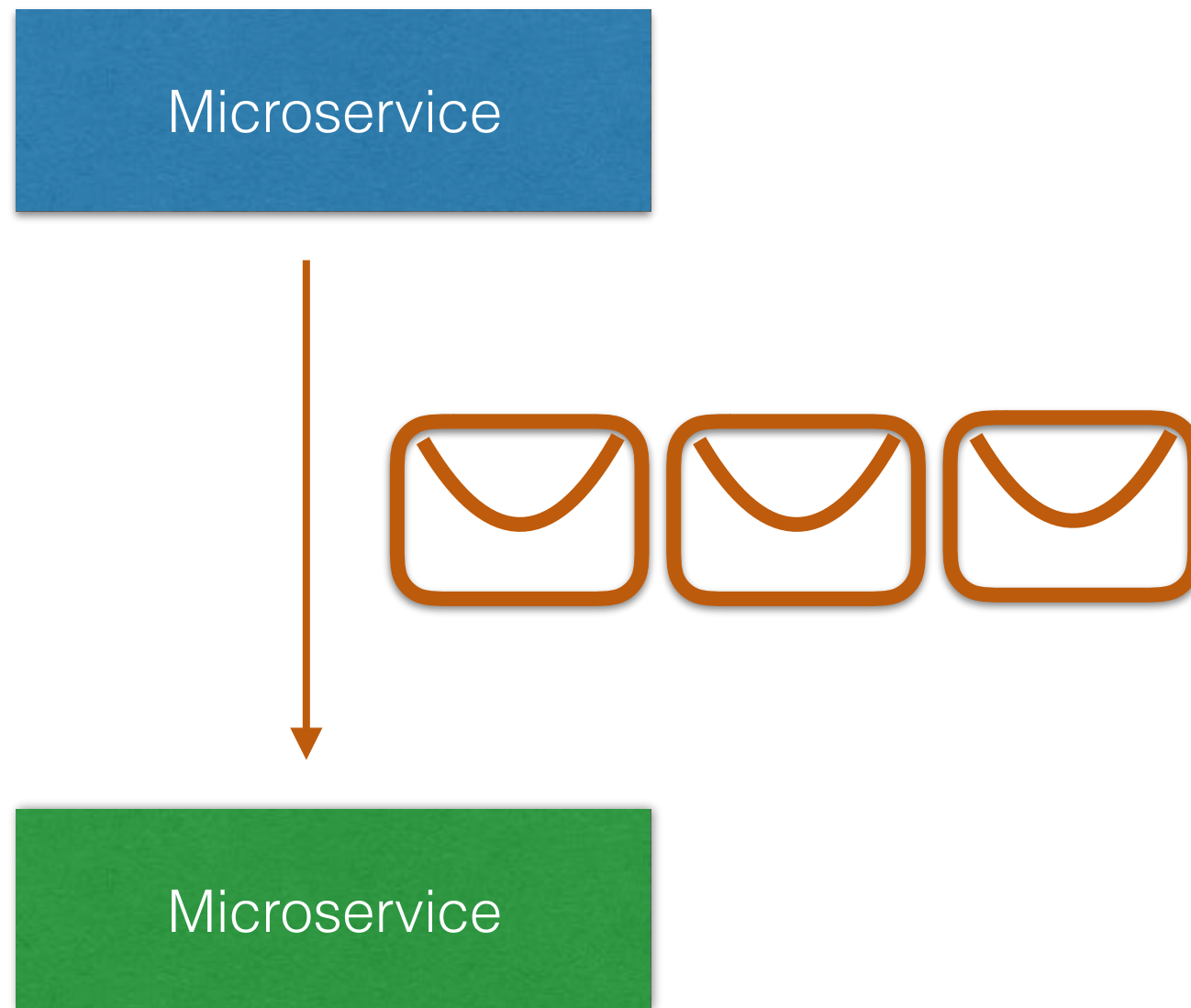
REST



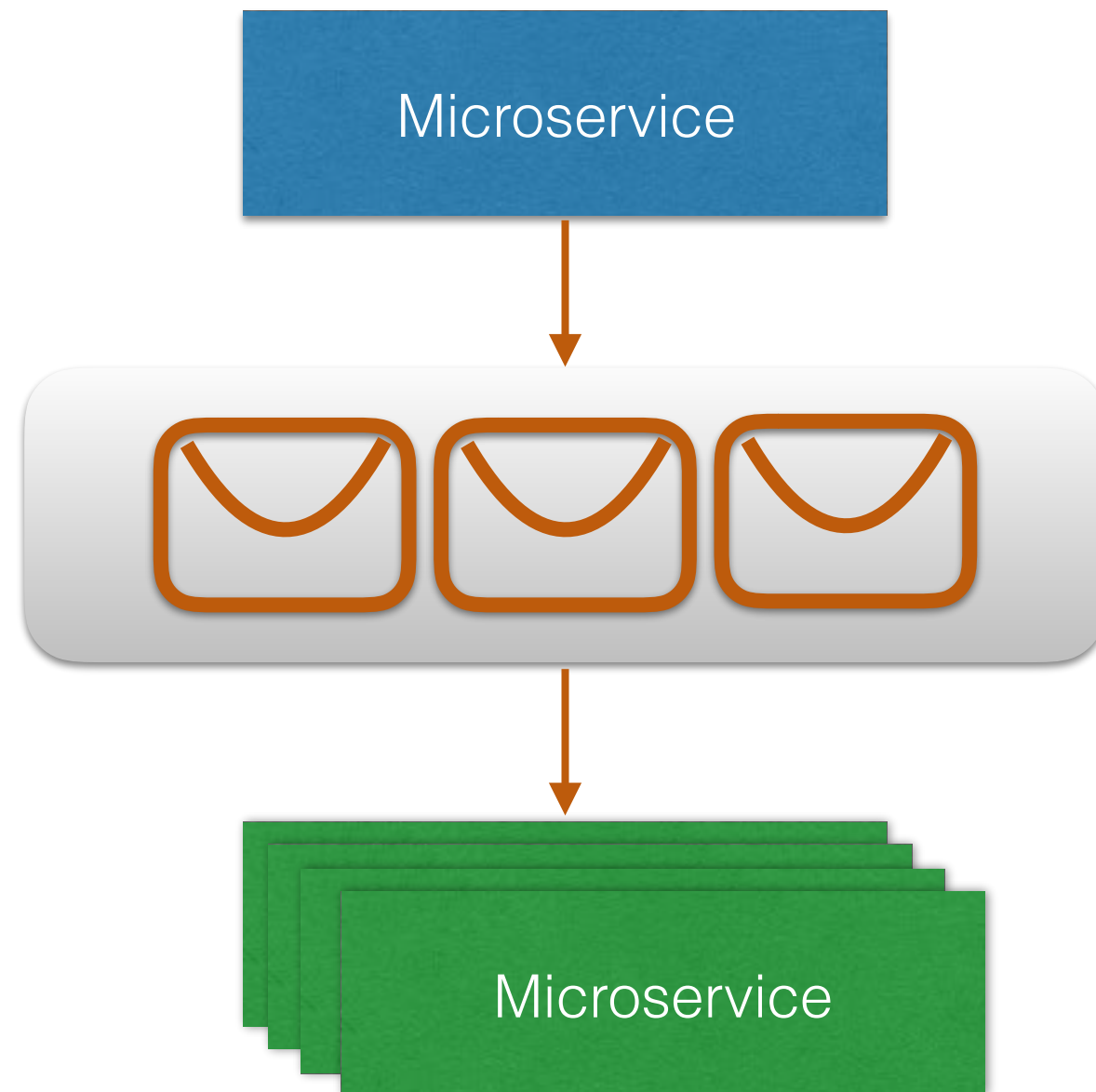
基于消息的异步方式

- 以使用Broker为例
 - 消息生产者与消费者解耦
 - broker可缓存消息
 - 支持多种通信模式
- broker添加了复杂度
- “请求/响应”模式不适用

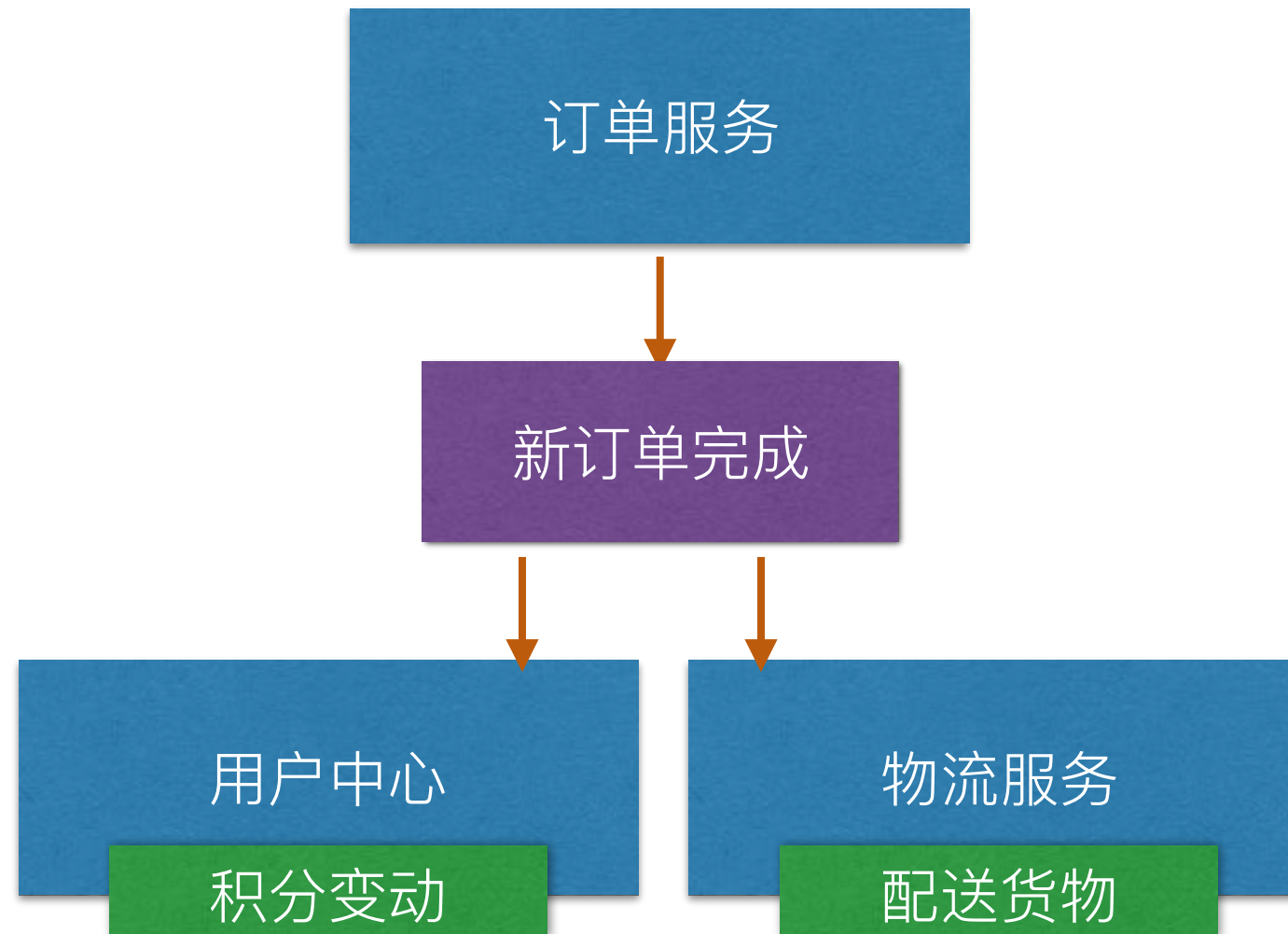
消息



消息

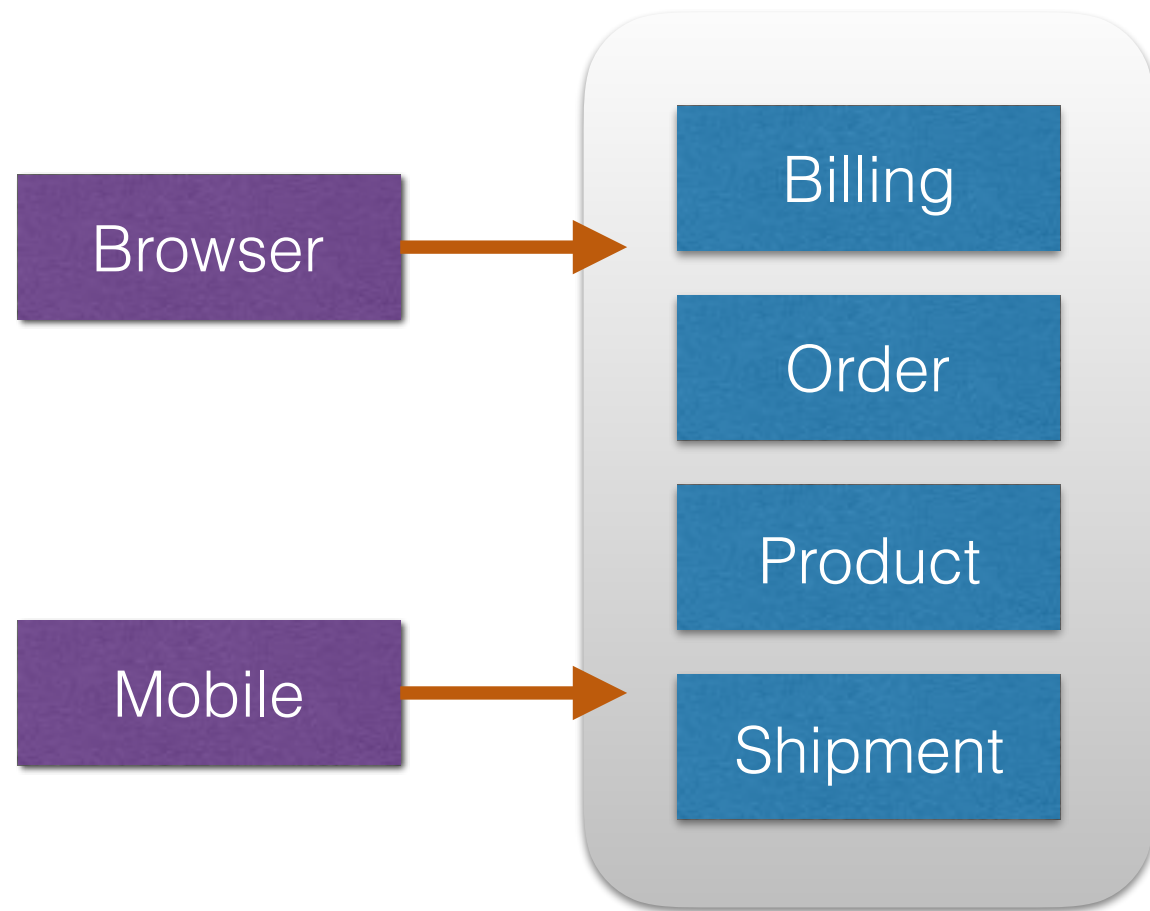


事件驱动

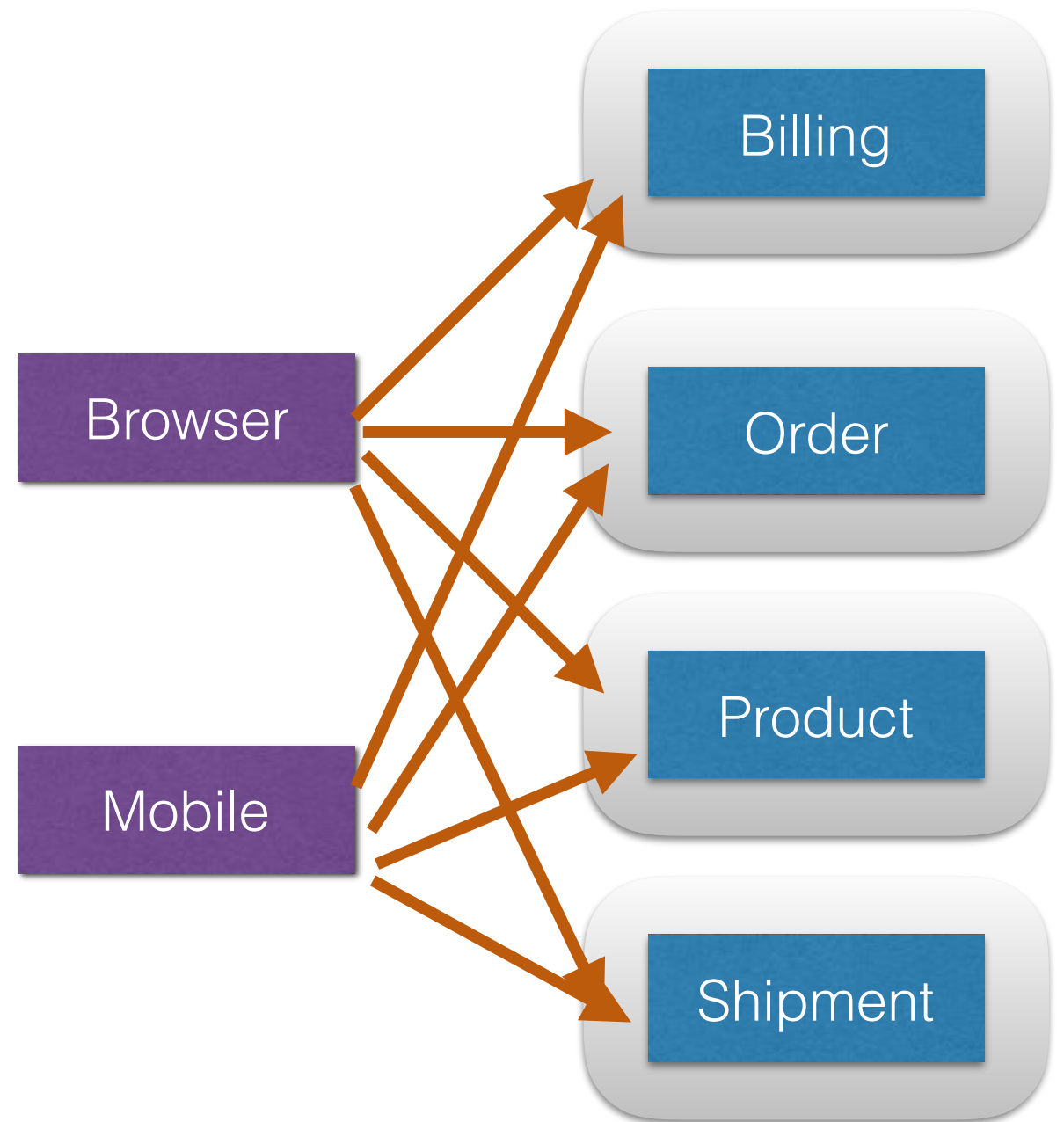


API 网关

单体



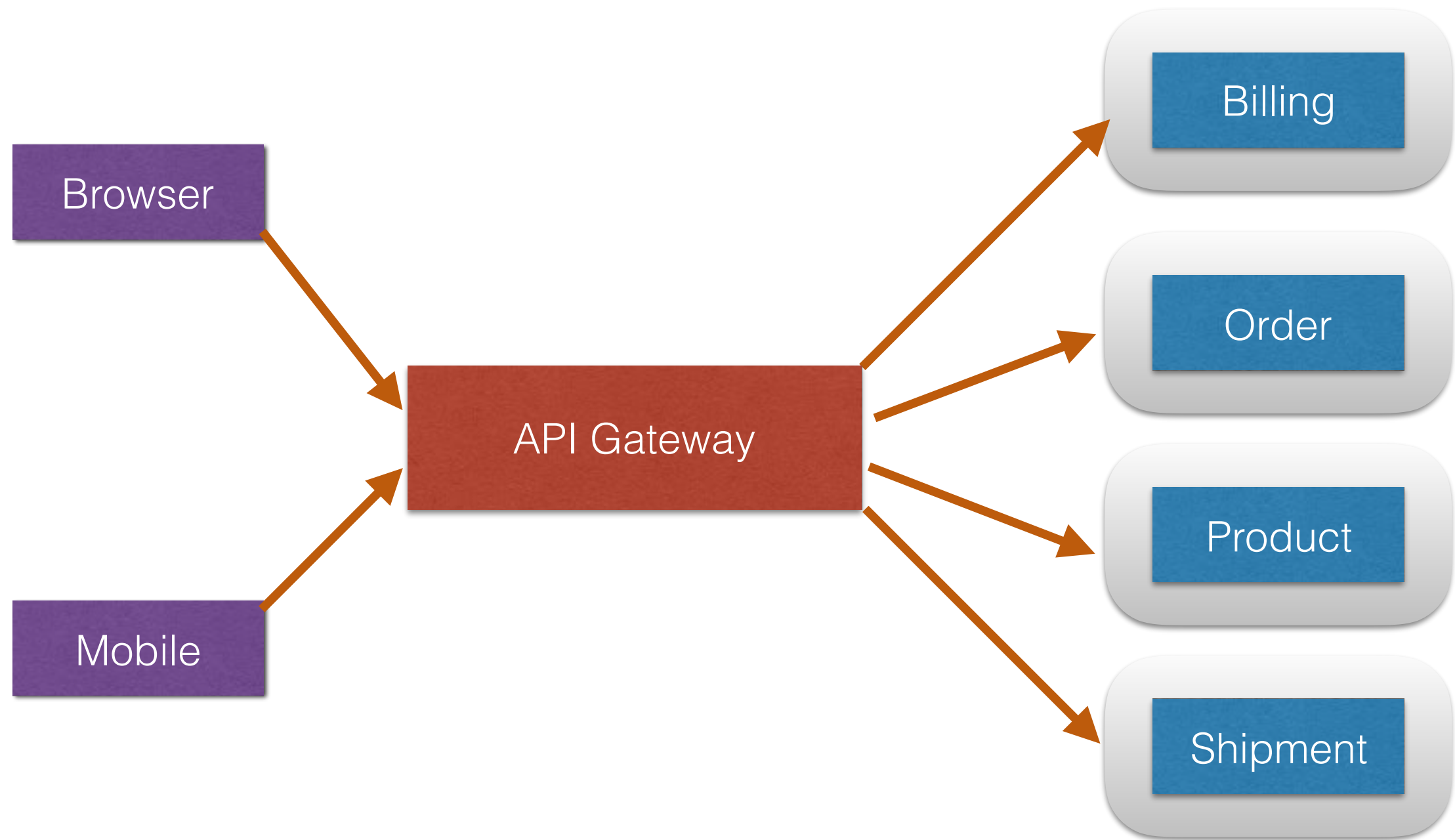
微服务



带来的问题

- 入口(Entry Point)变动需消费者(Client)跟随变动
- 后端服务架构调整对外可见
- 服务接口版本一致性问题
- 请求数量增多
- 协议支持友好度，如AMQP、Thrift

API 网关



API 网关的优点

- 封装了系统内部架构，简化消费者使用
- 请求组合，给消费者灵活定制API，减少请求量
- 单入口，协议转换为Web-Friendly
- 服务端变化带来的影响降到最低
- 负载均衡、请求路由、身份验证...

API 网关的缺点

- 带来了额外的开发量
- 需要管理API路由规则
- 额外的硬件、网络和运营成本
- 隐含的系统瓶颈

常用API 网关

- **Zuul** <https://github.com/netflix/zuul>
- **AWS API Gateway** <https://aws.amazon.com/api-gateway/>
- **Nginx** <https://www.nginx.com/>
- **Kong** <https://getkong.org/>



THANKS!



— 扫码了解更多 —