跟我做一个Java微服务项目

Week #4 数据处理 / 刘俊强

欢迎关注StuQ公众号

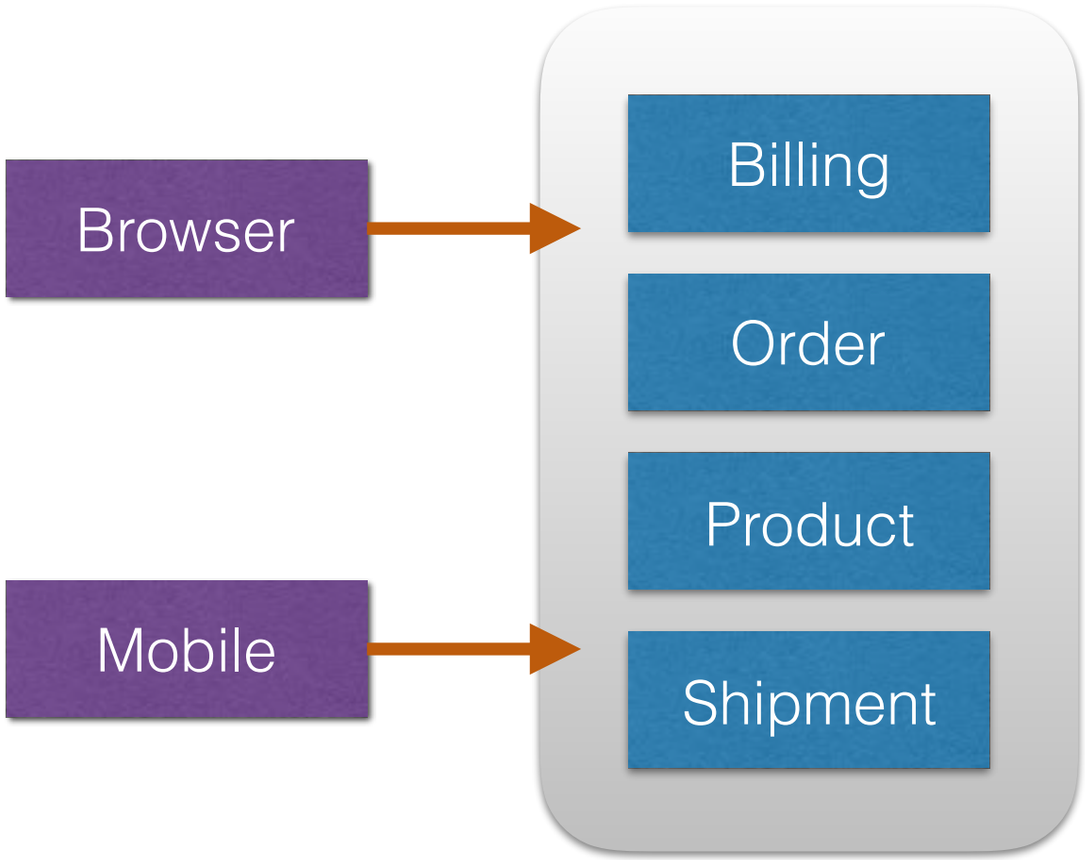
欢迎关注我的微信公众号

大纲

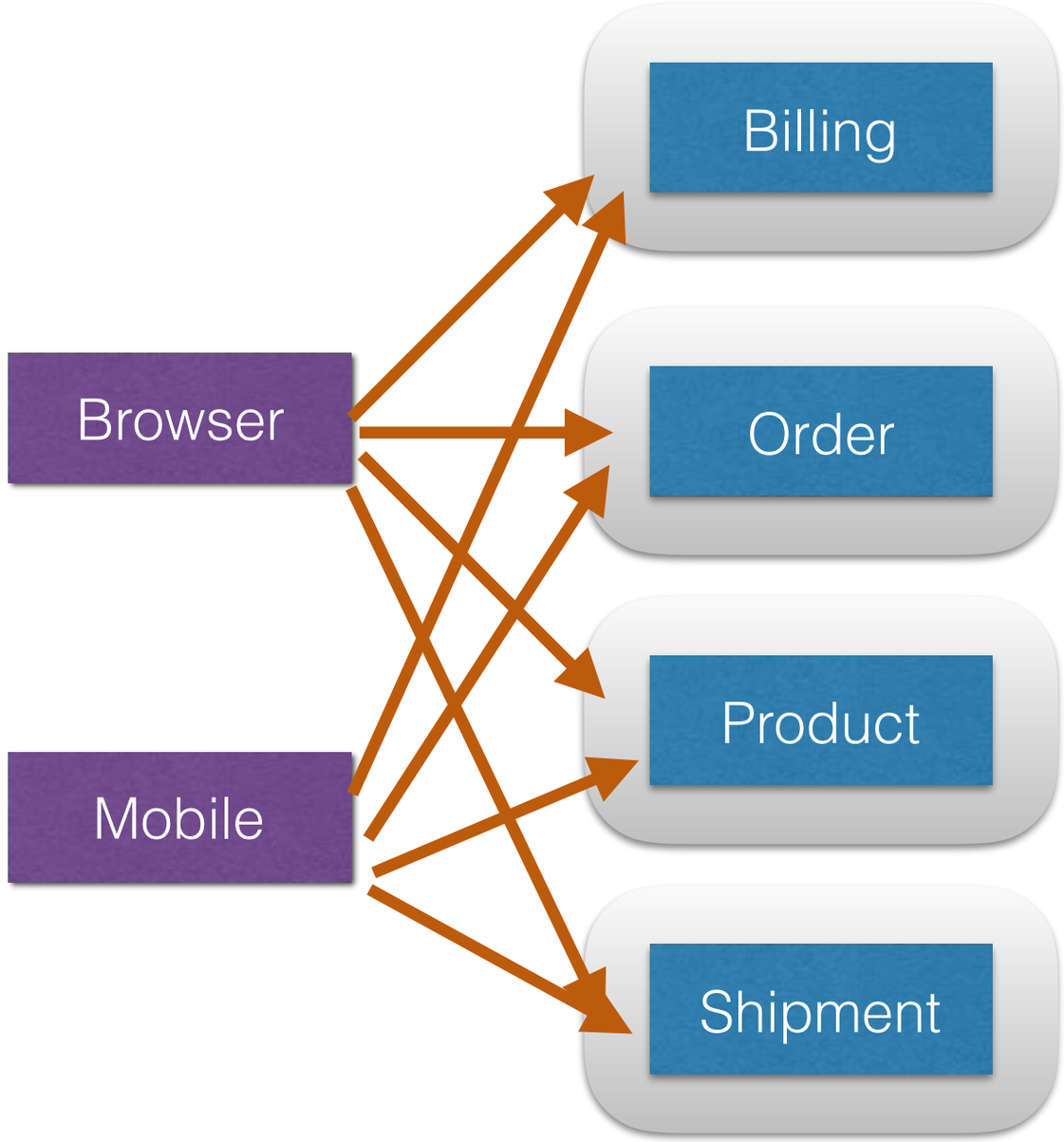
- API 网关
- 数据存储
- 微服务间数据共享
- 命令查询责任分离 & 事件溯源

API 网关

单体


Browser
Mobile
Billing
Order
Product
Shipment

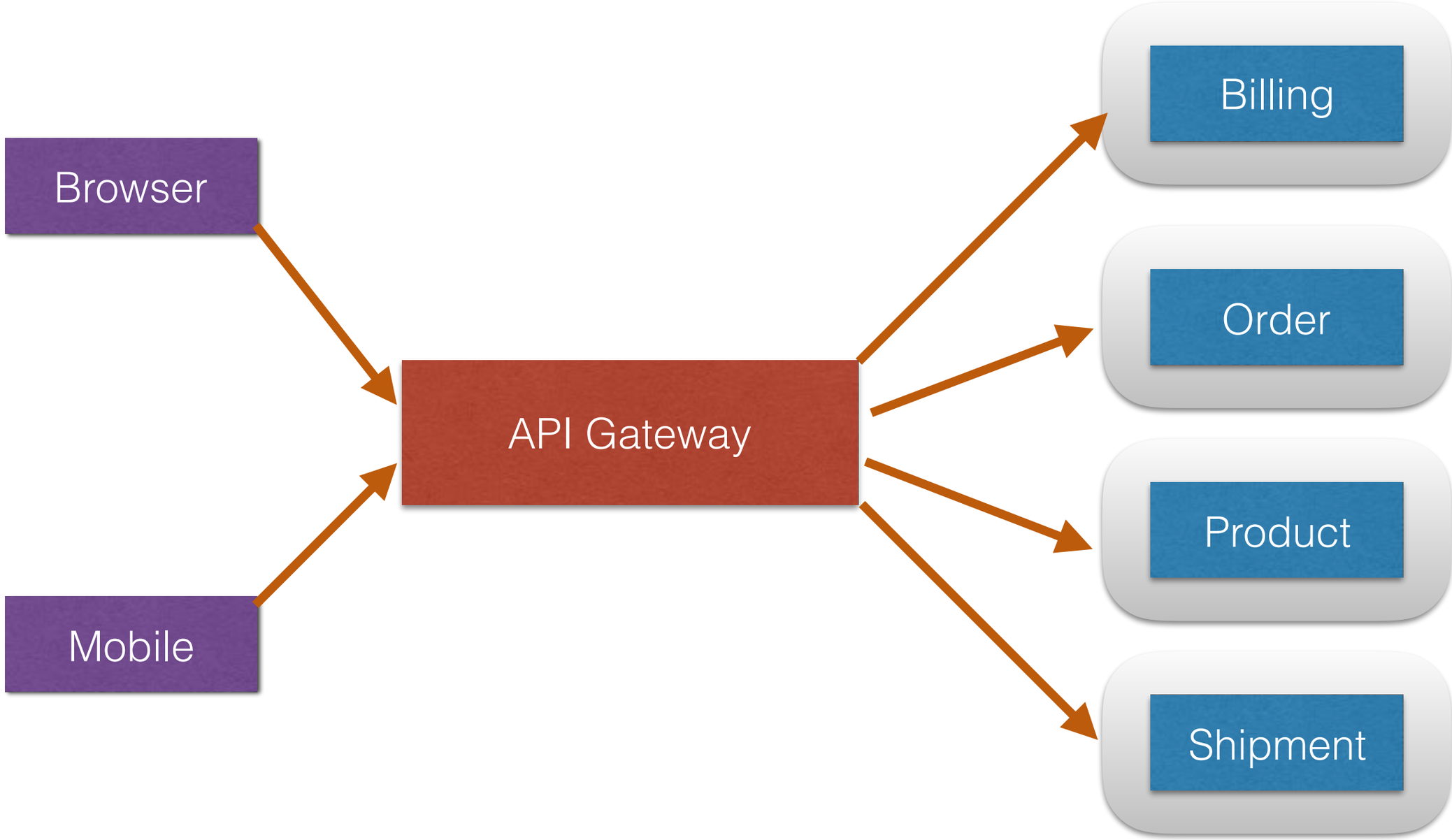
微服务


Browser
Mobile
Billing
Order
Product
Shipment

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


Billing
Browser
Order
API Gateway
Product
Mobile
Shipment

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...

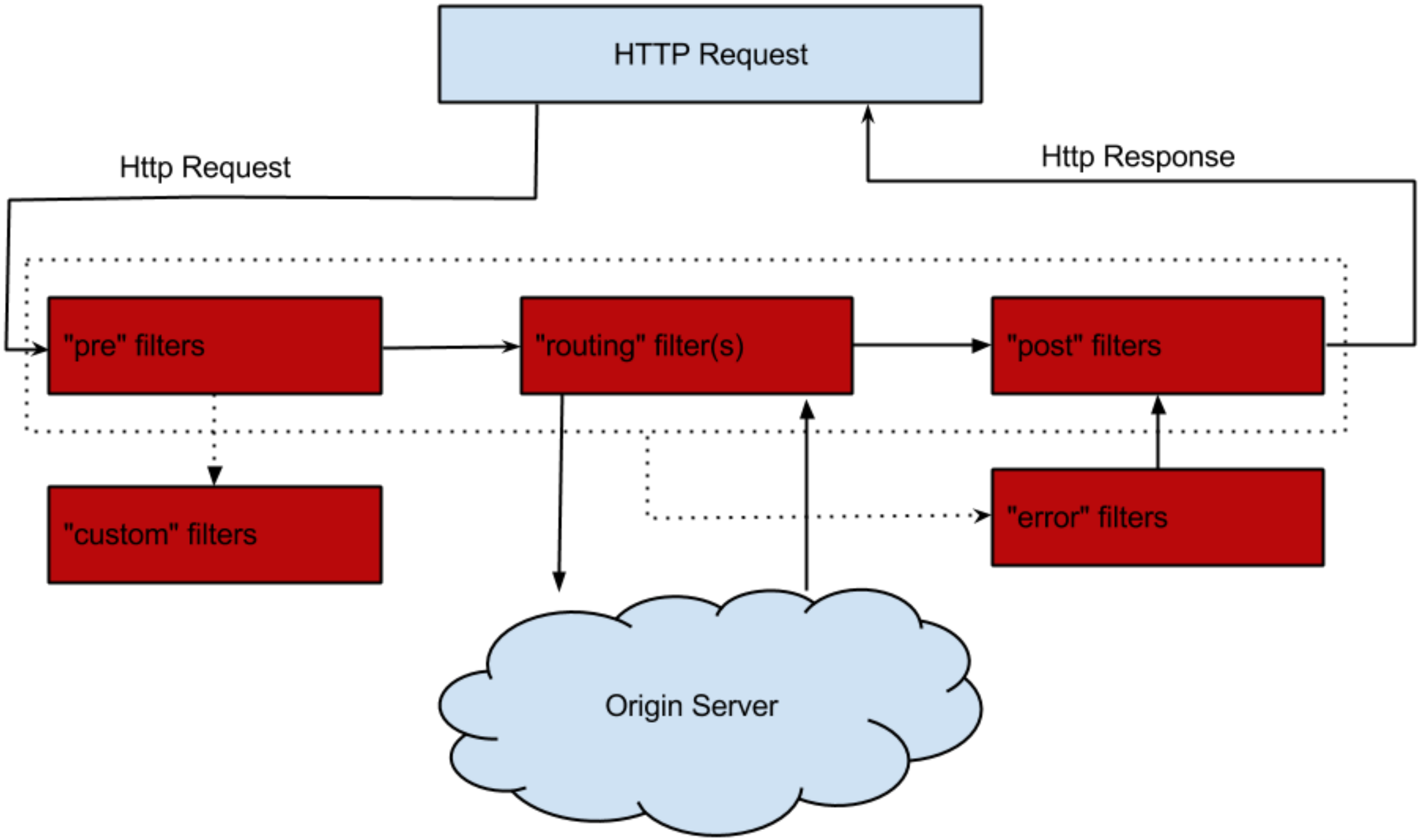
API 网关的缺点

- 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/

Zuul 请求生命周期

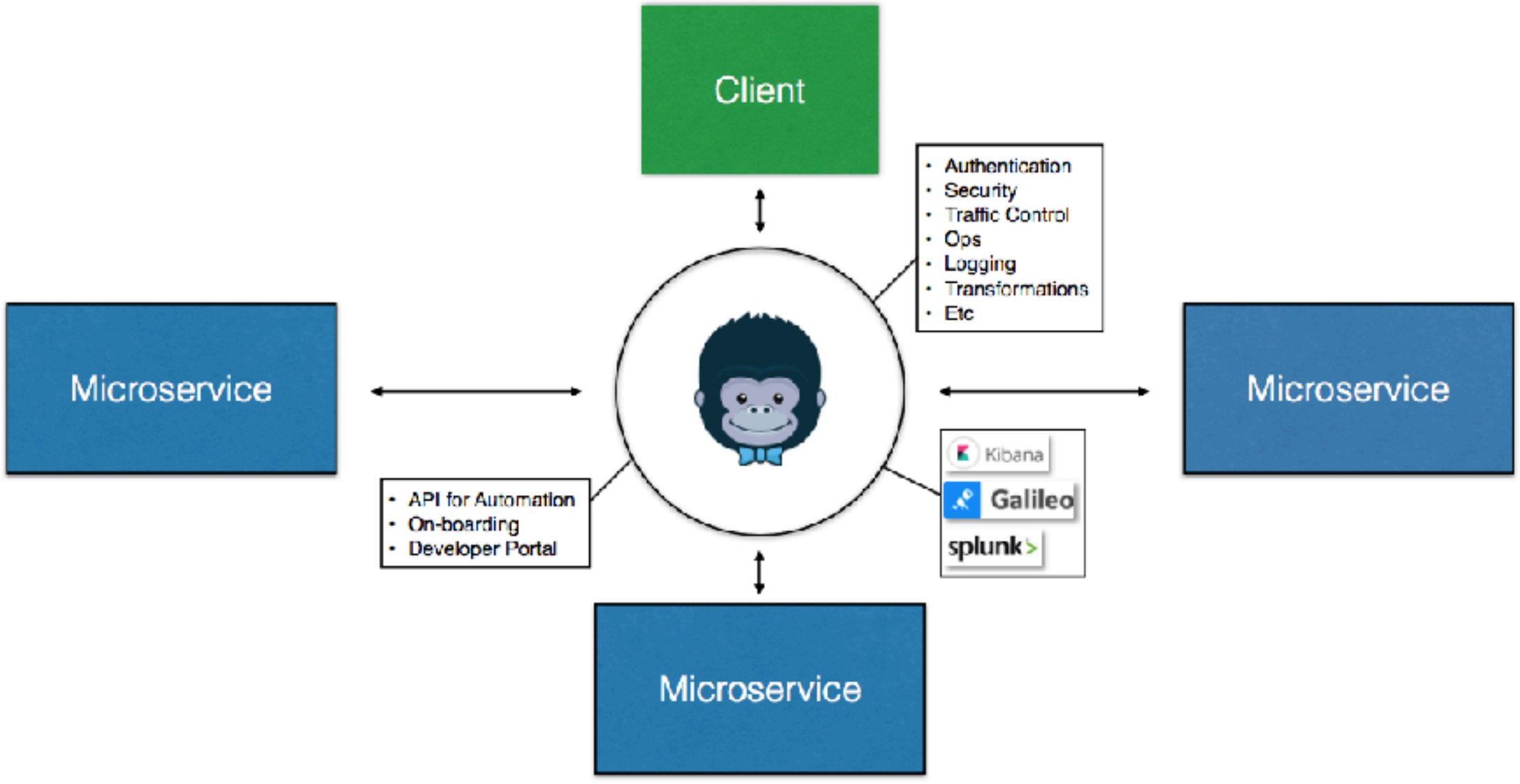

HTTP Request
Http Request
Http Response
"pre" filters
"routing" filter(s)
"post" filters
"custom" filters
"error" filters
Origin Server

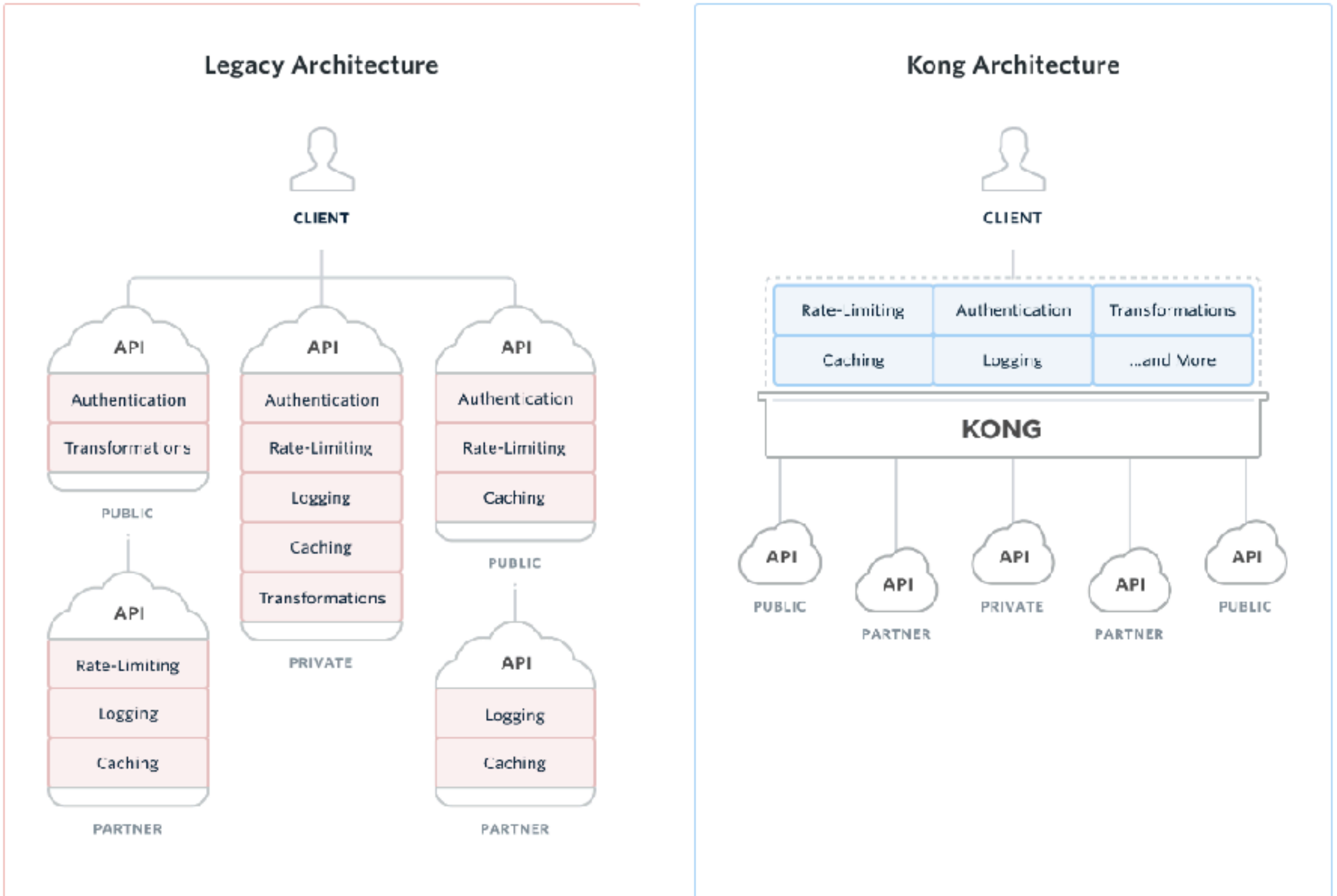
Zuul 请求生命周期

- **PRE** 过滤器，执行于路由至原始服务之前
  - 验证、选取源服务等
- **ROUTING** 过滤器，将请求路由至原始服务
  - 服务发现、入口匹配等，基于HttpClient或Ribbon
- **POST** 过滤器，执行于请求已路由至原始服务后
  - 数据格式转换、度量统计等
- **ERROR** 过滤器，以上阶段出错时执行

Zuul Demo

Kong


Client
Authentication
Security
Traffic Control
Ops
Logging
Transformations
Etc
Microservice
Microservice
API for Automation
On-boarding
Developer Portal
Kibana
Galileo
splunk>
Microservice

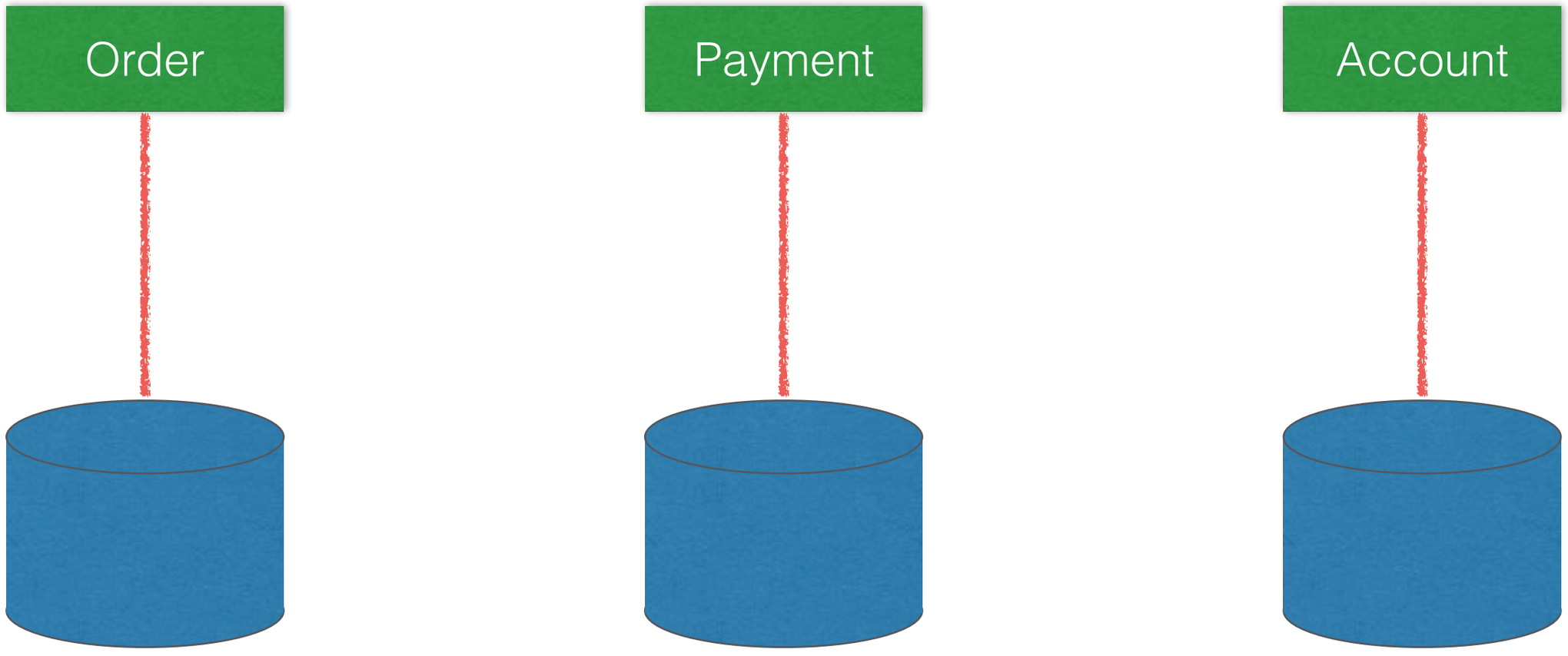


构建于Nginx(OpenResty)之上，集成常用功能成为API Gateway

Kong Demo

数据存储

微服务间数据存储的分离

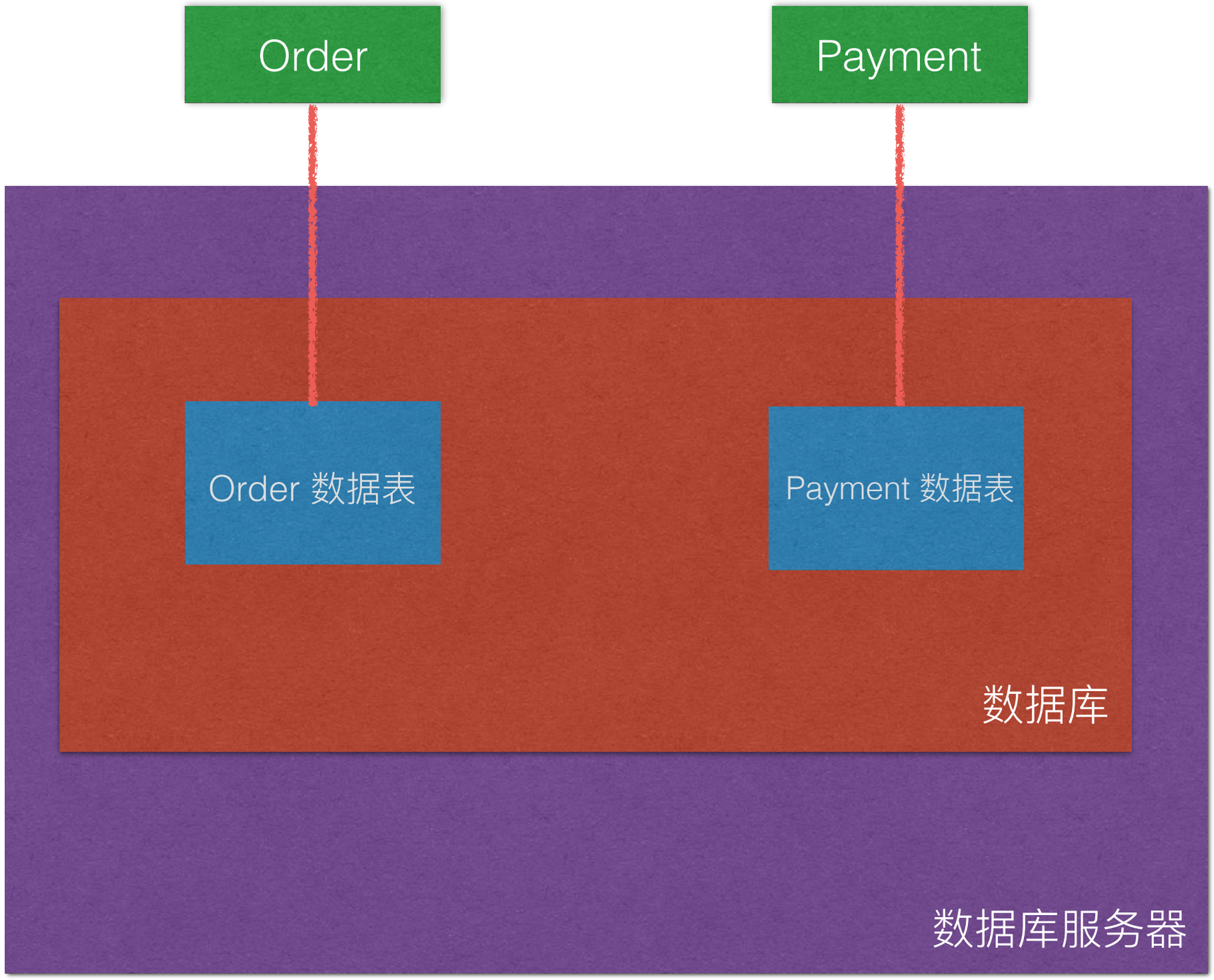


原则：每个微服务数据存储分开，其它微服务不能直接访问其数据存储。

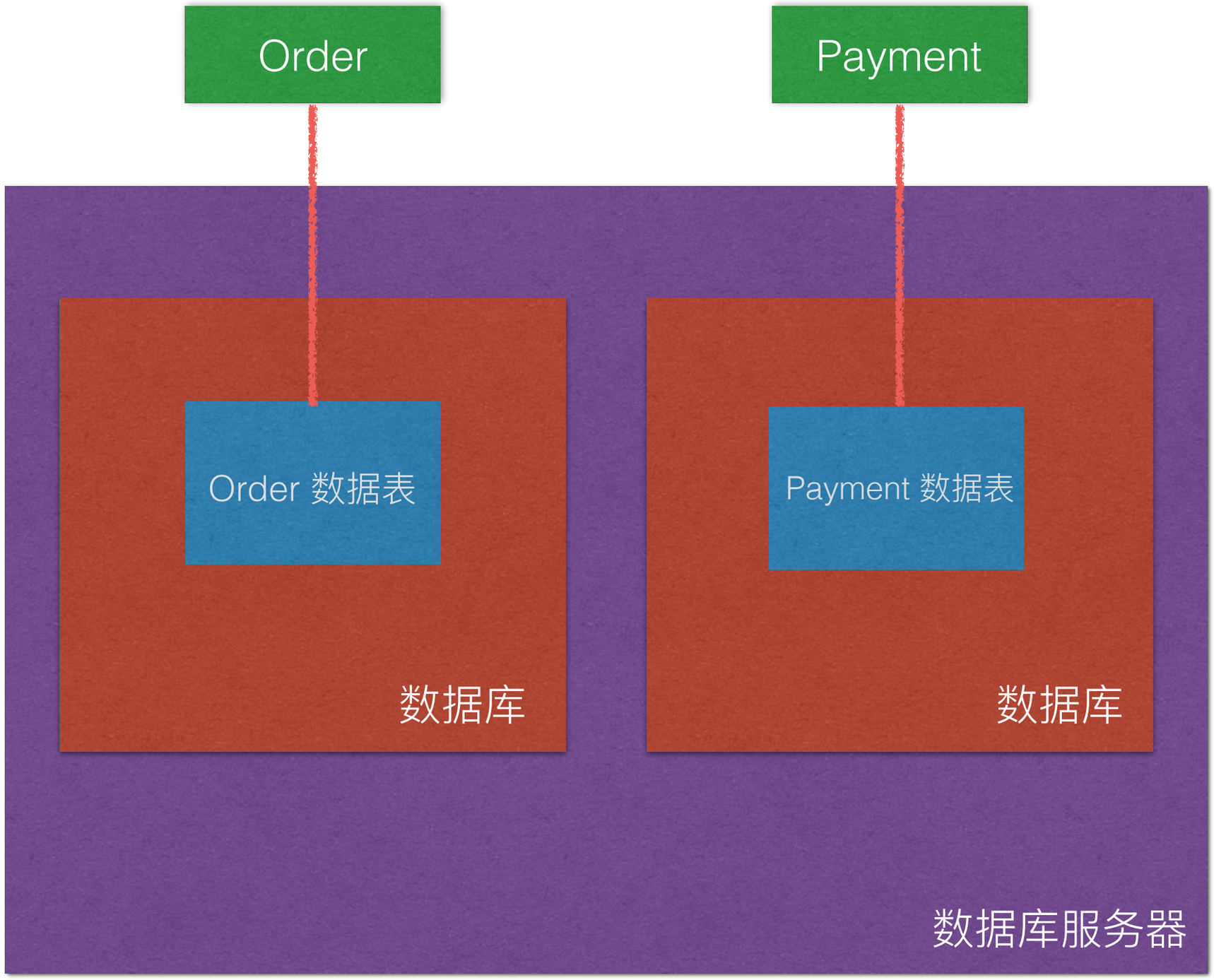
数据存储分离的方式

- 独立数据表
- 独立逻辑数据库
- 独立数据库服务器

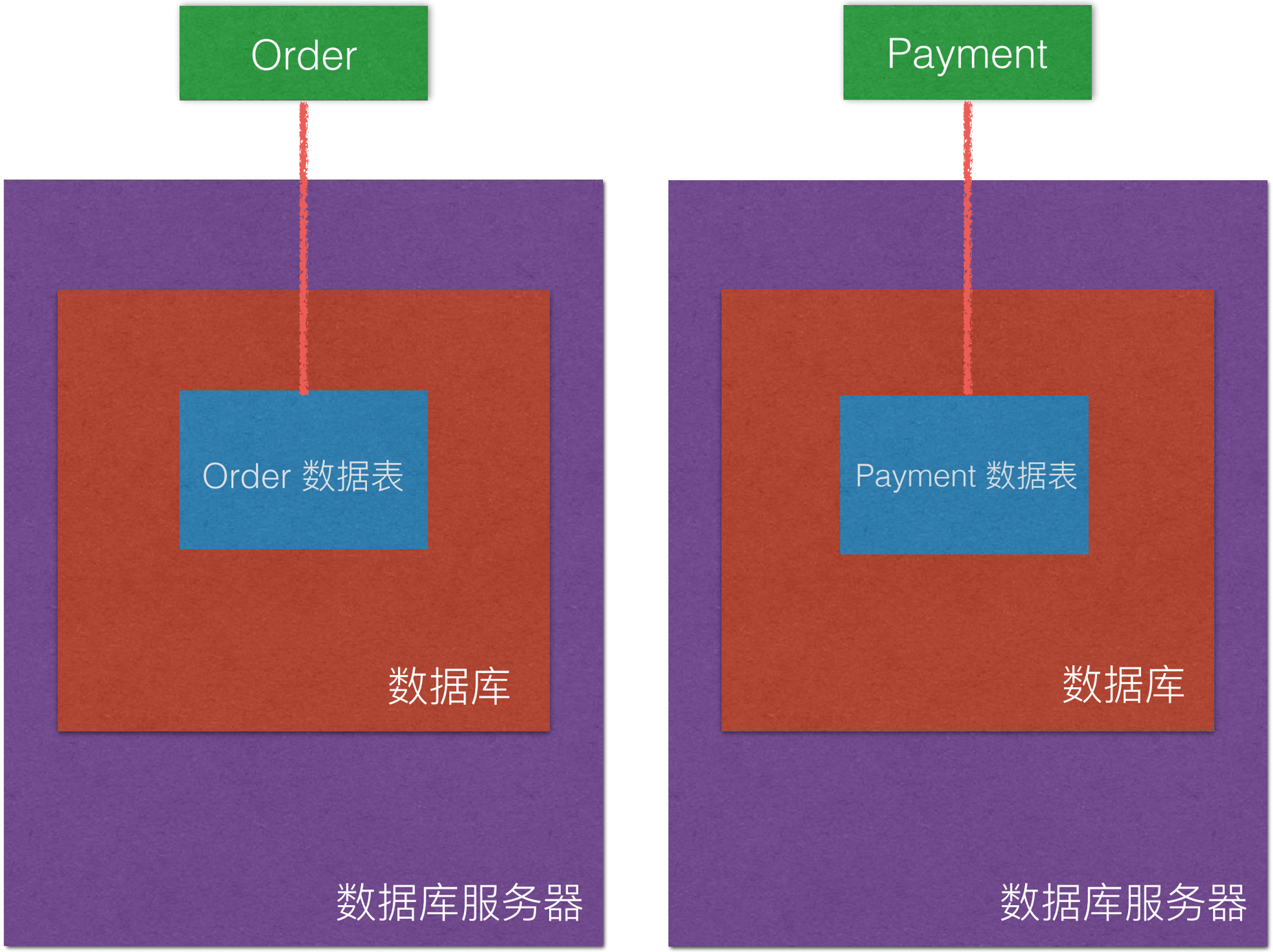
独立数据表



独立逻辑数据库



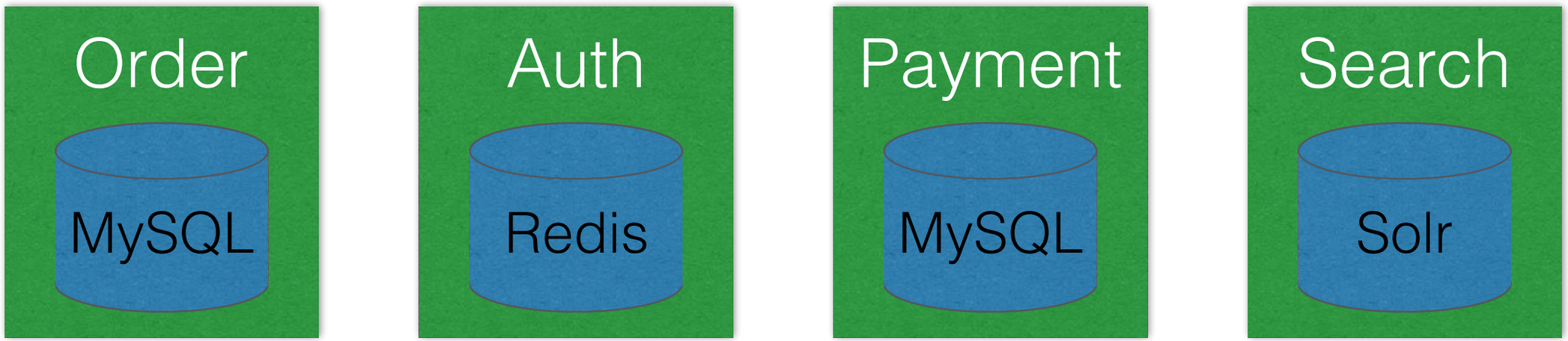
独立数据库服务器



混合存储

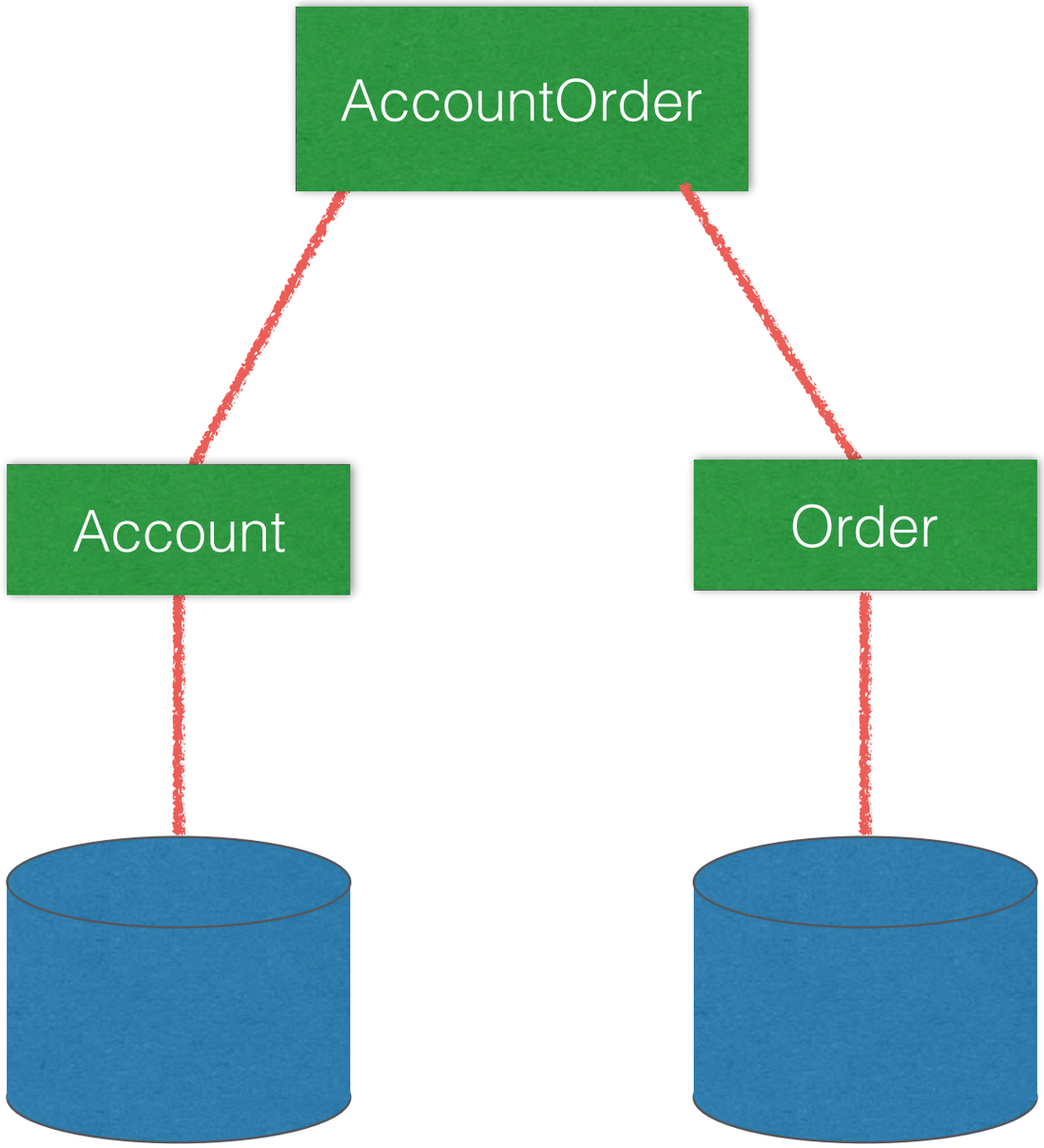
Polyglot Persistence

- 多语言持久化/存储
- 多样持久化/存储
- 多种类持久化/存储
- 混合持久化/存储


Order
MySQL
Auth
Redis
Payment
MySQL
Search
Solr
Cart
Redis
Account
MySQL
Recommendation
Neo4J
Notification
Redis

微服务间数据共享

数据查询



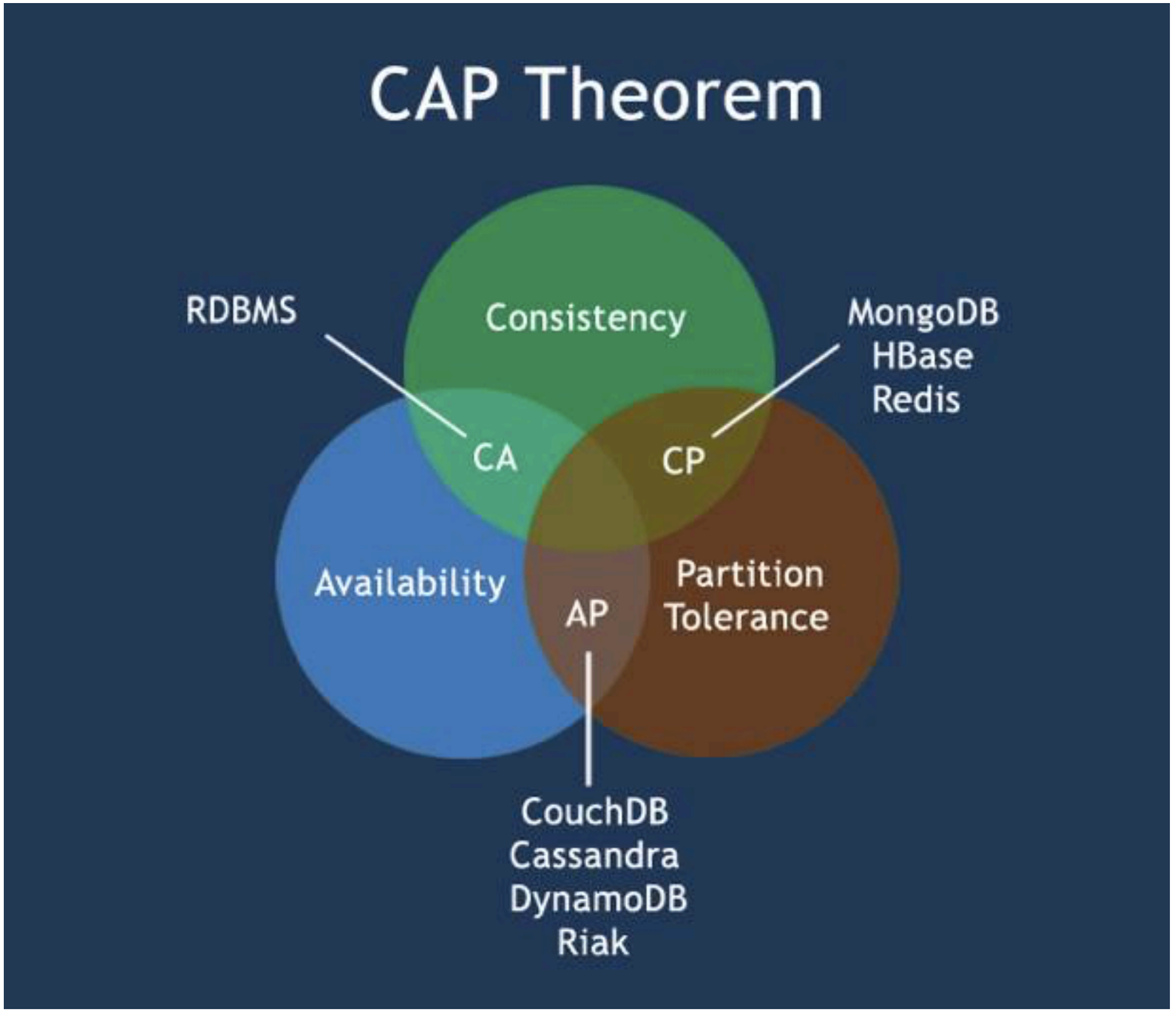
CAP定理

CAP定理

- Consistency 一致性
- Availability 可用性
- Partition Tolerance 分区容错性

分布式系统在给定时间内，不能同时满足3项，最多满足2项

数据库与CAP定理


CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak

不同的数据需要不同的保证策略

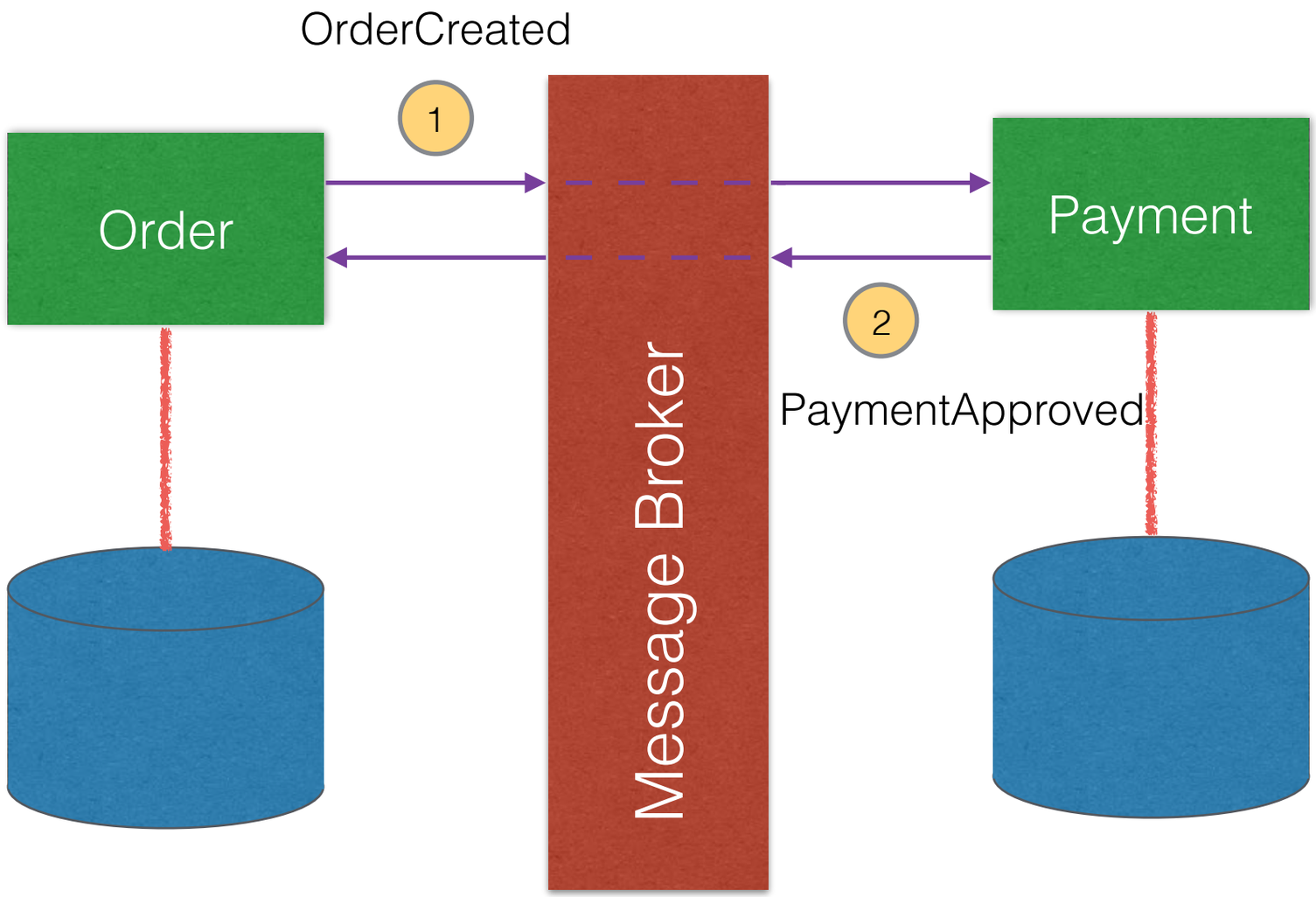
微服务间数据共享

- 事件驱动型架构 Event-Driven Architecture
- 最终一致性 Eventual Consistency
- 事件溯源 Event Souring
- 命令查询责任分离 Command Query Responsibility Segregation

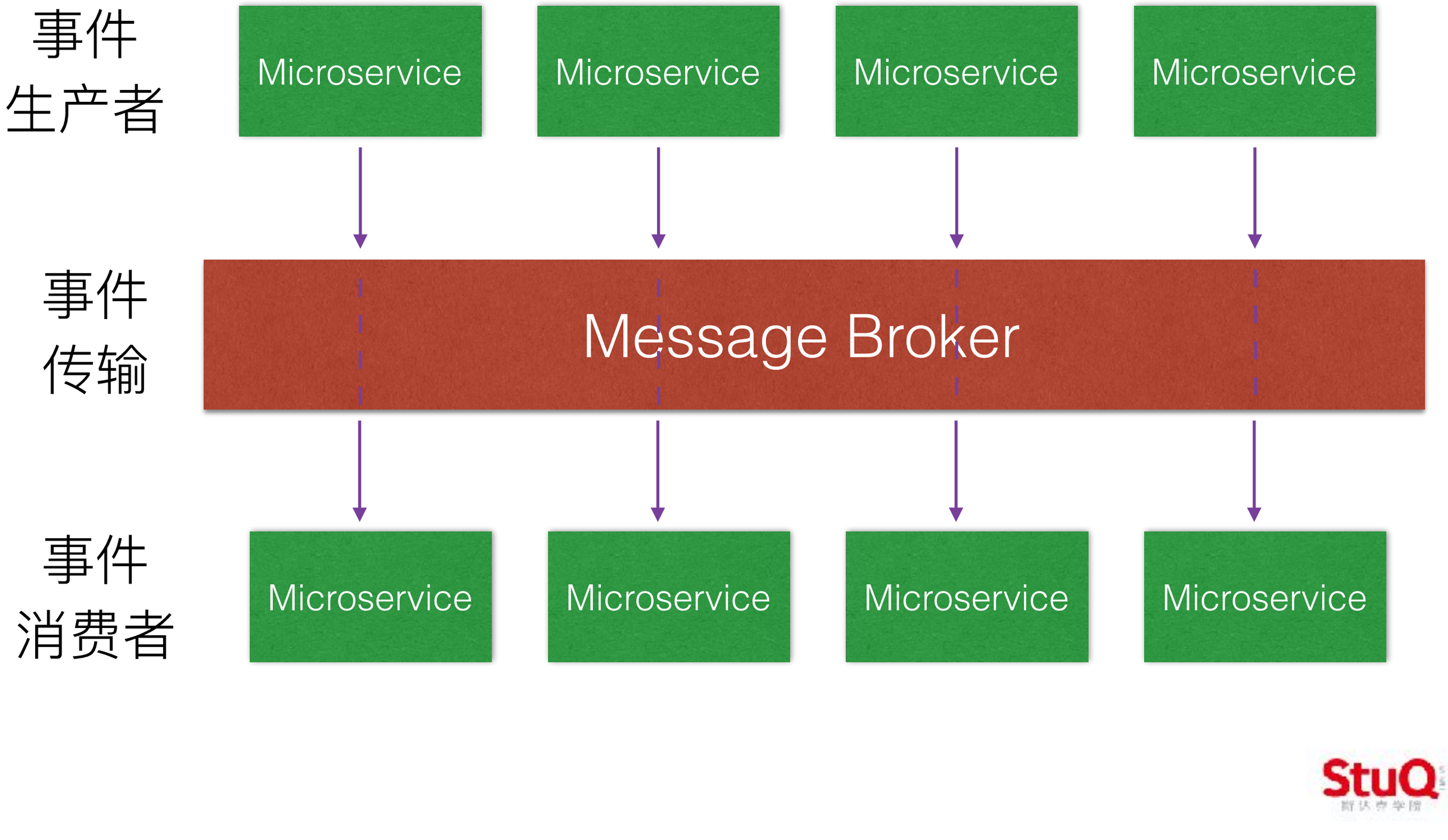
事件驱动型架构

- 使用事件来实现跨多个服务的业务逻辑
- 通过发布-订阅模型，微服务间耦合宽松
- 无需使用分布式事务，且微服务间互不影响

微服务间事件


OrderCreated
1
Order
Message Broker
Payment
2
PaymentApproved

事件驱动型架构


事件
生产者
Microservice
Microservice
Microservice
Microservice
事件
传输
Message Broker
事件
消费者
Microservice
Microservice
Microservice
Microservice

一致性

- 弱一致性，如cache
- 最终一致性，如Amazon S3、DNS
- 强一致性，如关系型数据库

命令查询责任分离

Command Query Responsibility Segregation

CQRS 命令查询责任分离

将查询、命令操作分离为两种独立子系统。

CRUD

CRUD

- C: Create
- R: Read
- U: Update
- D: Delete

CRUD示例

```
public interface ProductService {

    Product createProduct(final Product product);

    Product findProductById(final long id);

    Product updateProduct(final Product product);

    boolean deleteProductById(final long id);
}
```

CQRS

```
public interface ProductReadService {
    Product findProductById(final long id);
}
```

不影响数据的

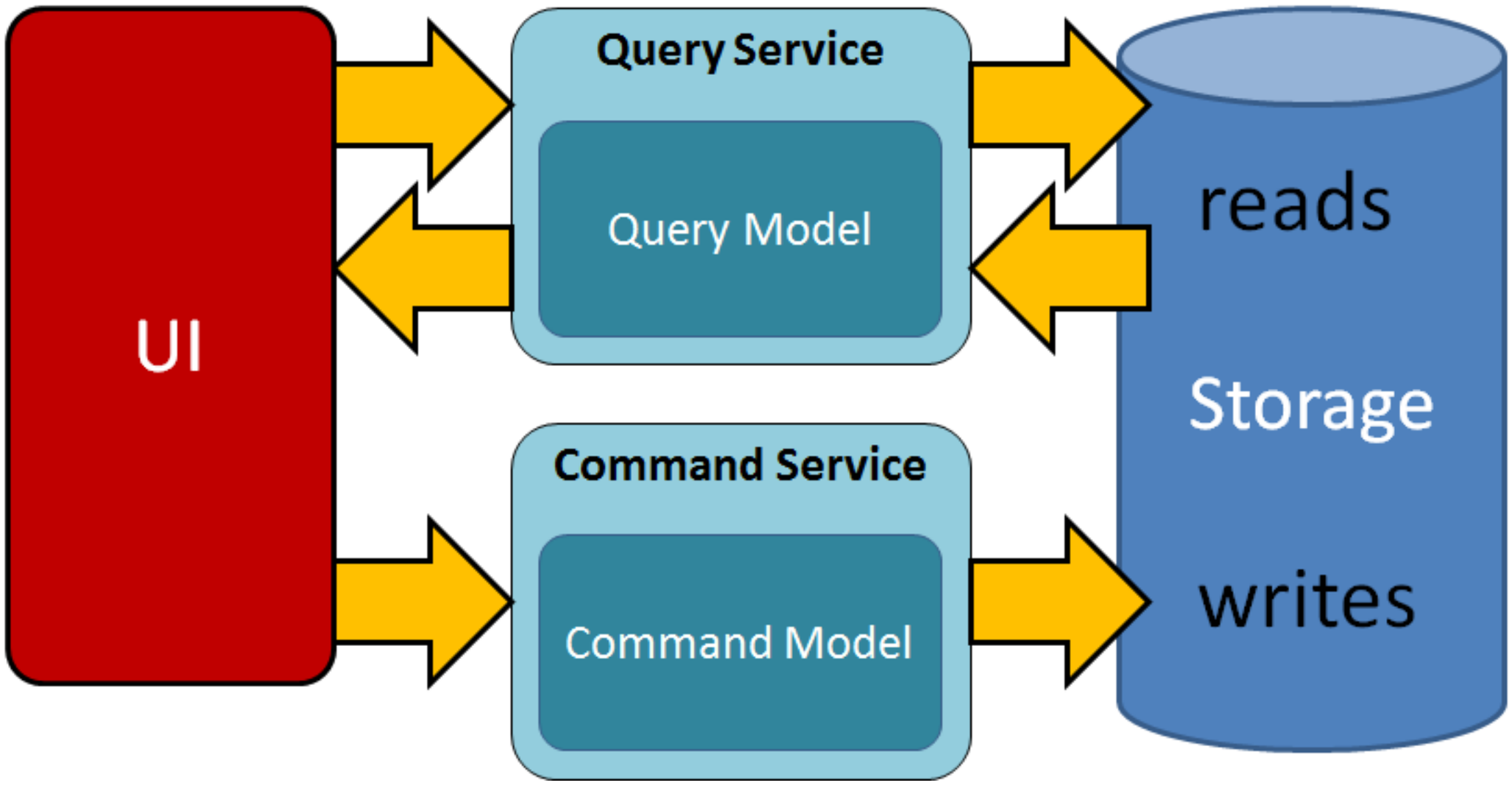
查询

```
public interface ProductWriteService {

    void createProduct(final Product product);

    void updateProduct(final Product product);

    void deleteProductById(final long id);
}
```

修改数据的

命令

CQRS Pattern



CQRS的好处

- 数据一致性：最终一致性
- 可扩展性：耦合性低
- 高可用性：分开解决高可用挑战
- 伸缩性：合理分配计算资源

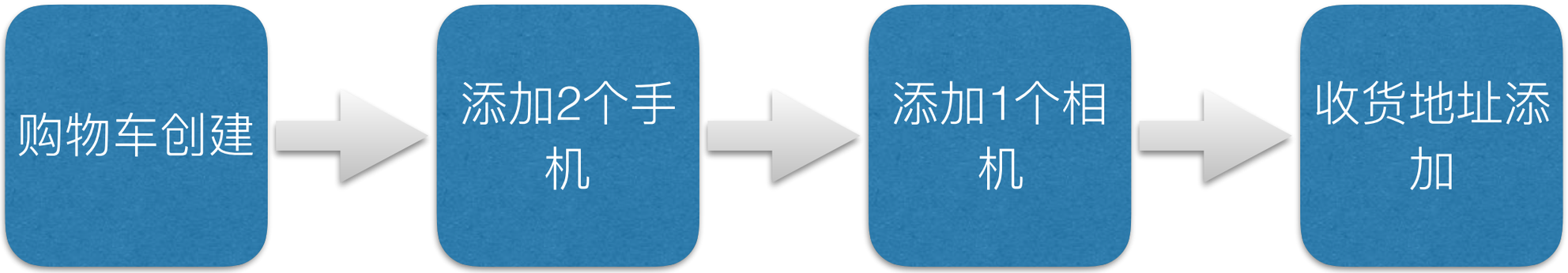
事件溯源 Event Souring

事件溯源 Event Souring

- 以事件为中心的方式来保存业务实体
- 不存储业务实体当前状态
- 存储实体状态改变的事件序列
- 事件举例：
  - Cart：ItemAddedEvent, ItemRemovedEvent, OrderPlacedEvent
  - Order：OrderCreated, OrderCancelled, OrderApproved,

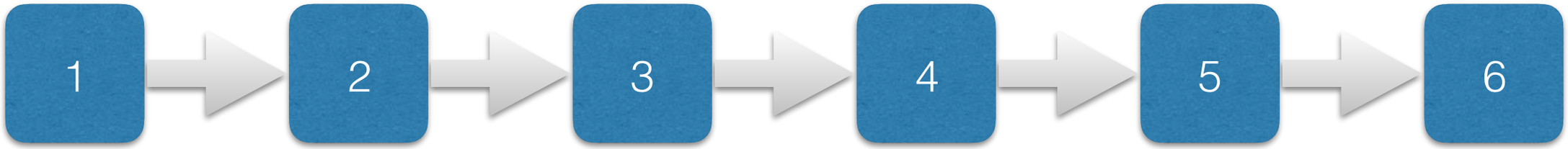
    OrderRejected, OrderShipped

事件溯源 Event Souring



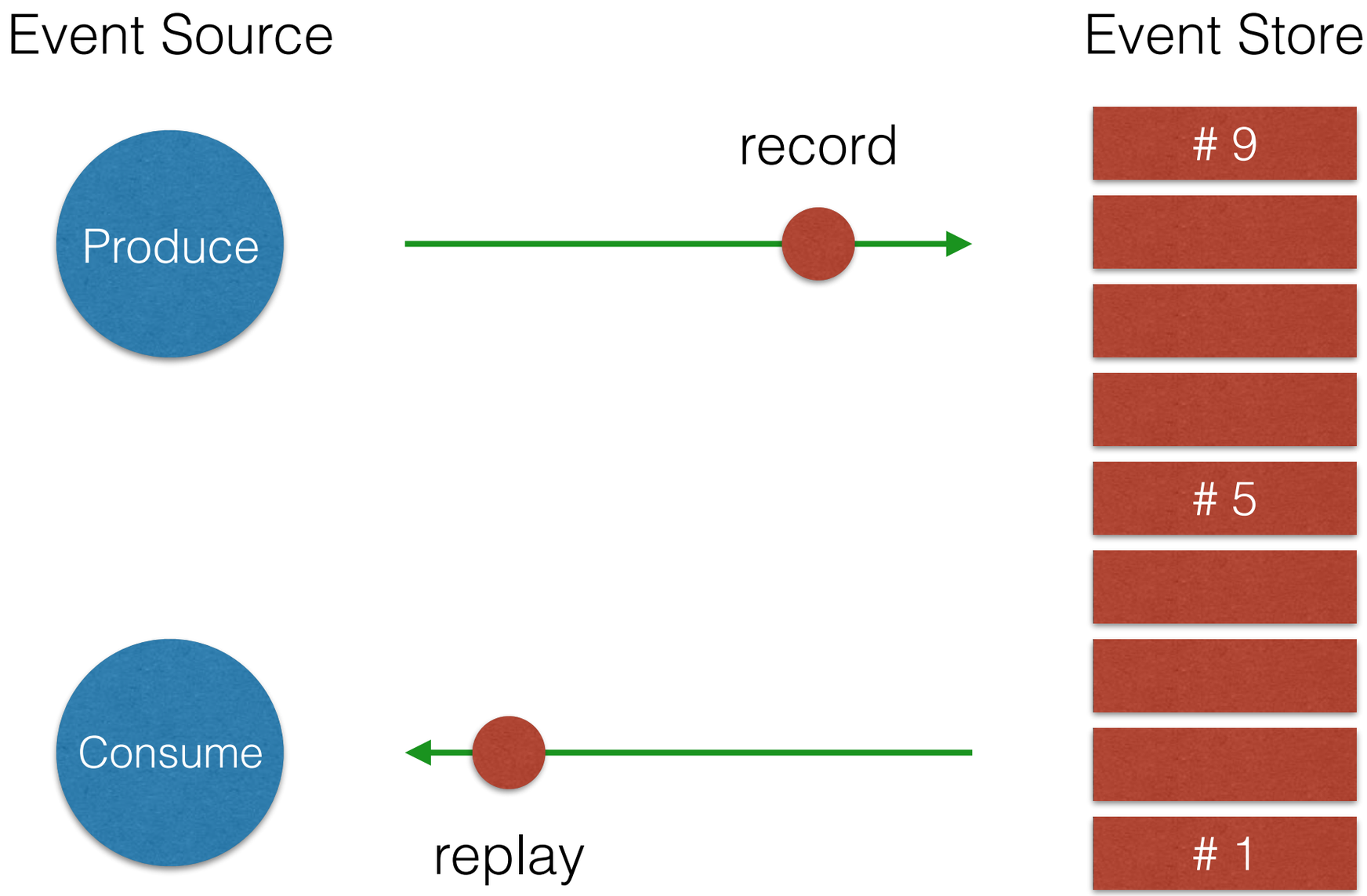
存储状态改变序列 Storing Deltas

事件溯源 Event Souring



通过事件回放，实体对象状态能够回到源头

事件存储 Event Store



currentState = fx(initialState, entity, events)

Event Store = Message Broker + Database

事件溯源的好处

- 解决微服务间数据一致性的问题
- 记录所有状态变更
- 实现了业务事务：带时序的业务历史记录
- 可靠的事件发布：推送通知、数据分析、日志审计等

Event Store方案

- Apache Kafka http://kafka.apache.org/

- akka persistance http://akka.io/

- Event Store https://geteventstore.com/

StuQ 斯达克学院

THANKS!

— 扫码了解更多 —