消息队列

- 为什么要使用消息队列
  - 解耦
  - 异步
  - 削峰
- 常用的消息队列
  - ActiveMQ
  - RabbitMQ
    - 响应快
    - erlang不易二次开发
  - RocketMQ — Java，易扩展
  - kafka — 大数据实时计算、日志采集
- 消息队列的优缺点
  - 可用性 — 集群
  - 增加复杂性
    - 消息丢失怎么办
    - 重复消息怎么处理
    - 如何保证消息传递的顺序性
  - 一致性问题 — 分布式事务
- 如何保证消息队列的高可用
  - RabbitMQ
    - 普通集群
      - 没有做到高可用
      - 有数据拉取的开销和单实例的瓶颈问题
    - 镜像集群
  - RocketMQ — 双主双从
- 如何保证消息不丢失
  - confirm确认 — 生产者没有成功发送到MQ Broker
  - 消息持久化 — 消息发送给MQ Broker后，Broker宕机导致内存中的消息数据丢失
  - 手动ack确认 — 消费者消费到了消息，但是没有处理完毕就出现异常导致丢失
- 如何保证消息不被重复消费？如何保证消息消费的幂等性
  - 原因：生产者->MQ网络抖动，没有确认
  - 原因:MQ->消费者，应答丢失
  - 消息重复的原因是网络不可达
  - 解决方法：消息幂等性
    - 发送消息时携带全局唯一消息id
    - 消费者获取消费后根据id在redis/db中查询是否存在消费记录
    - 如果没有消费过就正常消费，消费完毕后写入redis/db
    - 如果消息消费过就直接舍弃
- 如何保证消息消费的顺序性
  - 不追求全局有序，追求局部性顺序消费
    - 让需要保持有序的消息放到一个队列中
    - 使用分段锁对队列加锁只有对前一个确认了才能拿下一个
    - 定义 MessageQueueSelector，并根据id等确认应该发送到哪个队列
- 大量消息堆积怎么处理
  - 原因
    - 1.网络故障
    - 2.消费方消费后没进行ack的确认
    - 3.消费的速度跟不上生产的速度
  - 解决方法
    - 1.检查并修复消费方的正常消费速度
    - 2.将堆积消息转存到容量更大的MQ集群
    - 3.增加多个消费者节点并进行消费
    - 4.消费完毕后，恢复原始架构
- 消息过期怎么处理
  - 消息设置了过期时间，如果超时还未被消费，则视为消息过期，过期消息可以转存到死信队列
    - 1.创建一个死信队列消费者
    - 2.将过期消息存放到数据库
    - 3.查询数据库消费日志，重新发送消息到MQ