# 什么是消息队列？

消息队列（Message Queue，简称 MQ）（消息代理，队列服务器）消息中间件。 产品基于高可用分布式集群技术，提供消息订阅和发布、消息轨迹查询、定时（延时）消息、资源统计、监控报警等一系列消息云服务，是企业级互联网架构的核心产品。 MQ 历史超过9年，为分布式应用系统提供异步解耦、削峰填谷的能力，同时具备海量消息堆积、高吞吐、可靠重试等互联网应用所需的特性。

## 主流消息队列有哪些呢？

ActiveMQ

老牌MQ，采用JMS规范，服务性能，数据存储，集群架构

（以前传统企业比较流行，但性能上来说没有其他的高性能MQ好）

Kafka

LinkedIn开源的MQ，目前属于Apache，主要用于日志收集，基于Pull的消费模式

（吞吐量大，追求高吞吐量，但是对数据的一直重复和错误不是太高，所以用来采集数据和日志打印，不支持事务）

比较依赖于Zookeeper

RocketMQ

阿里开源的消息中间件，采用纯JAVA语言编写，高吞吐高可用，采用了Kafka的优点，优化了Kafka的缺点，他优化了数据的可靠性，还有事务

（主要用来交易，充值，计算，息推送，日志流式处理，inglog）

并且在Spring-Cloud-Alibaba中对RockteMQ做了很好的支持

RabbitMQ

使用Erlang语言开发（安装RabbitMQ必须安装Erlang），基于AMQP协议，主要面向消息，队列，路由，可靠，安全，对数据的可靠性和稳定性有很好支持，然后对性能吞吐的要求还要在其次

## 什么是AMQP高级协议？

AMQP全称：Advanced Message Queuing Protocol（高级消息队列协议）

AMQP定义：是具有现代特征的二进制协议，是一个提供统一消息服务的应用层标准高级消息队列协议，是应用层协议的一个开放标准，为面向消息的中间件设计

## 互联网大厂为什么选择RabbitMQ？

1. 开源免费，性能优秀，稳定性保障
2. 提供可靠性消息投递模式（confirm）返回模式（return）
3. 和SpringAMQP完美的整合，API丰富
4. 集群模式丰富，表达式配置，HA模式，镜像队列模型
5. 保证数据不丢失的前提做到高可靠性，可用性

## RabbitMQ的高性能是如何做到的呢？

RabbitMQ是使用Erlang语言开发，而Erlang最初在于交换机领域的架构模式，这样使得RabbitMQ在分布式之间进行数据交互的性能是非常优秀的，而Erlang他有着和Socket一样的延迟（很快）

## 为什么需要消息队列

1. 异步处理
2. 流量削峰
3. 日志处理
4. 应用解耦

# 消息队列核心

## AMQP的核心概念是什么？

Server：又称Broker，接受客户端的连接，实现AMQP实体服务

Connection：连接，应用程序与Broker的网络连接

Channel：网络信道，几乎所有的操作都在Channel中进行，Channel代表一 个回话任务，

Message：消息，服务和应用程序之间传送的数据，由Properties和Body组成，Properties可以对消息进行修饰，比如消息的优先级、延迟等高级特性，Body就是消息的内容

Virtual host：虚拟地址，用于进行逻辑隔离，最上层的消息路由。一个Virtual Host里面可以有若干个Exchange和Queue，同一个Virtual Host

里面不能有相同名称的Exchange或Queue

Exchange：、 交换机，接收消息，根据路由键转发消息绑定的队列

Binding：Exchange和Queue之间的虚拟连接，binding中可以包含routing key

Routing key：一个路由规则，虚拟机可以用它来确定如何路由一个特定消息

Queue：也称为Message Queue，消息队列，保存消息并将他们转发给消费者

## Exchange是什么？

## 如何确保100%的投递成功？

保障消息的成功发出

保障MQ节点的成功接收

发送端收到MQ节点（Broker）确认回答

完善的消息进行补偿机制

消息罗库（保存到数据库），对消息状态进行打标（状态进行标记）

消息的延迟投递，做二次确认，回调检查

## 幂等性

如何结局幂等性的问题呢？

唯一ID+指纹码机制，利用数据库主键去重

SELECT COUNT(1) FROM T\_ORDER WHERE ID = 唯一ID + 指纹码

好处：实现简单

坏处：高并发下有数据库写入的性能瓶颈

解决方案：跟进ID进行分库分表进行算法路由

利用Redis的原子性去实现

使用Redis进行幂等，需要考虑的问题

1. 我们是否要进行数据库落库，如果落库的话，关键解决的问 题是数据库和缓存如何做到原子性（同时成功或失败）
2. 如果不进行落库，那么存储到缓存中，如何设置定时同步的策略？

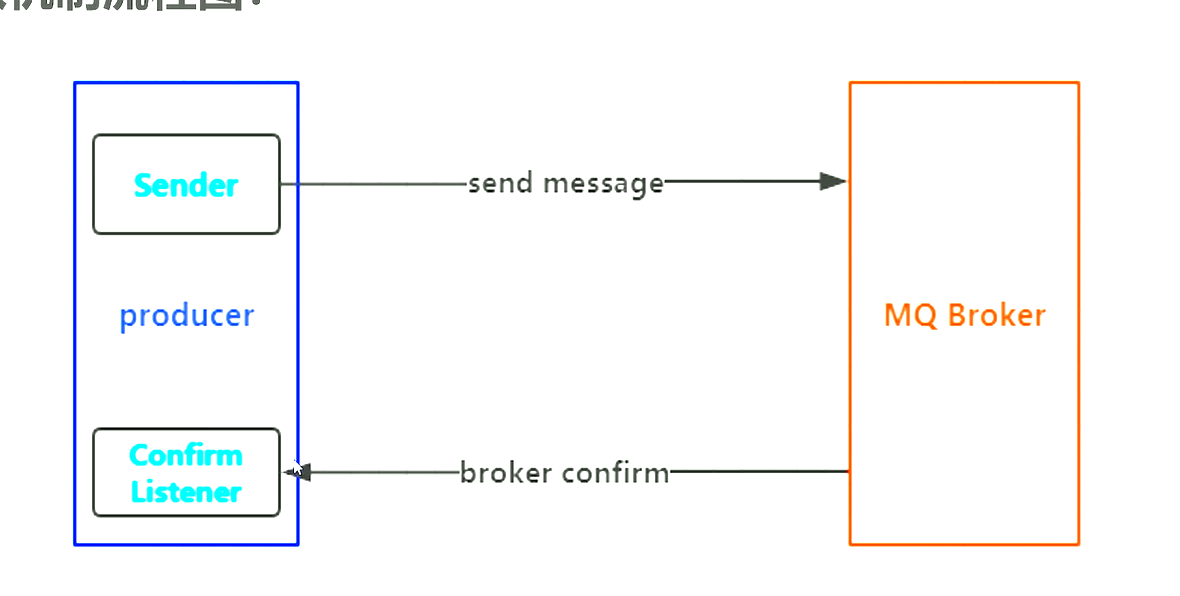
## 在海量订单的高峰期中，如何避免重复消费问题？

## Confirm确认消息

Confirm消息确认的机制：

消息确认，是指生产者投递消息后，如果Broker收到消息，则会给我们生产者一个应答

生产者进行接收应答，用来确定这条消息是否正常的发送到Broker。这种方式也是消息的可靠性投递的核心保障



类似于这张图中的一样，我们发送的一个请求，但是我们还是回去等他的响应，并且获取他的返回结果

通过Confirm Listener来监听他

## 自定义消费者

## 消息的ACK与重回队列

## 消息的限流

什么是消费端的限流？

假设一个场景，首先我们的RabbitMQ有上万条未处理的消息，我们随便打开一个消费者客户端会出现，巨量的消息瞬间全部推送过来，但是我们单个客户端无法同时处理这么多数据，就会引发崩溃，所以RabbbitMQ提供了一种qos（服务质量保证）功能，即在非自动确认消息的前提下，如果一定数目的消息未被缺人前，不进行新的消费

## TTl消息

## 死信队列

# 使用RabbitMQ

## 快速开始使用RabbitMQ

首先我们要先整理用到的东西

ConnectionFactory：获取连接工厂

Connnection：一个连接对象

Channel：数据通信通道，可发送和接受消息

Queue：具体的消息存储队列

Producer&Consumen：生产者和消费者