## AMQP是什么?

RabbitMQ就是 AMQP 协议的 `Erlang` 的实现 。规定了交换器、队列、绑定的协议。

- 交换器 (Exchange)：消息代理服务器中用于把消息路由到队列的组件。
- 队列 (Queue)：用来存储消息的数据结构，位于硬盘或内存中。
- 绑定 (Binding)：一套规则，告知交换器消息应该将消息投递给哪个队列。

# 如何保证消息的可靠性?

消息到MQ的过程中搞丢，MQ自己搞丢，MQ到消费过程中搞丢。

- 生产者到RabbitMQ：事务机制和Confirm机制，注意：事务机制和 Confirm 机制是互斥的，两者不能共存，会导致 RabbitMQ 报错。
- RabbitMQ自身：持久化、集群、普通模式、镜像模式。
- RabbitMQ到消费者：basicAck机制、死信队列、消息补偿机制。

# 事务机制?

RabbitMQ 客户端中与事务机制相关的方法有三个:

- `channel.txSelect` 用于将当前的信道设置成事务模式。
- `channel . txCommit` 用于提交事务 。
- `channel . txRollback` 用于事务回滚,如果在事务提交执行之前由于 RabbitMQ 异常崩溃或者其他原因抛出异常,通过txRollback来回滚。

# 发送确认机制?

生产者把信道设置为 `confirm` 确认模式,设置后，所有再改信道发布的消息都会被指定一个唯一的ID，一旦消息被投递到所有匹配的队列之后，RabbitMQ就会发送一个确认（`Basic.Ack`)给生产者（包含消息的唯一ID)，这样生产者就知道消息到达对应的目的地了。

# 消息传输保证层级?

- At most once:最多一次。消息可能会丢失，但不会重复传输。
- At least once：最少一次。消息绝不会丢失，但可能会重复传输。
- Exactly once：恰好一次，每条消息肯定仅传输一次。

# 集群中的节点类型?

- 内存节点：ram,将变更写入内存。
- 磁盘节点：disc,磁盘写入操作。
- RabbitMQ要求最少有一个磁盘节点。

# 生产者如何将消息可靠投递到MQ?

- 1.Client发送消息给MQ
- 2.MQ将消息持久化后，发送Ack消息给Client，此处有可能因为网络问题导致Ack消息无法发送到Client，那么Client在等待超时后，会重传消息；
- 3.Client收到Ack消息后，认为消息已经投递成功。

# MQ如何将消息可靠投递到消费者?

- 1.MQ将消息push给Client（或Client来pull消息）
- 2.Client得到消息并做完业务逻辑
- 3.Client发送Ack消息给MQ，通知MQ删除该消息，此处有可能因为网络问题导致Ack失败，那么Client会重复消息，这里就引出消费幂等的问题；
- 4.MQ将已消费的消息删除

# 如何保证RabbitMQ消息队列的高可用?

RabbitMQ 有三种模式：`单机模式`，`普通集群模式`，`镜像集群模式`。

- **单机模式**：就是demo级别的，一般就是你本地启动了玩玩儿的，没人生产用单机模式
- **普通集群模式**：意思就是在多台机器上启动多个RabbitMQ实例，每个机器启动一个。
- **镜像集群模式**：这种模式，才是所谓的RabbitMQ的高可用模式，跟普通集群模式不一样的是，你创建的queue，无论元数据(元数据指RabbitMQ的配置数据)还是queue里的消息都会存在于多个实例上，然后每次你写消息到queue的时候，都会自动把消息到多个实例的queue里进行消息同步。

# 对于消息中间机，你们是怎么做技术选型的?

| 特性 | ActiveMQ | RabbitMQ | RocketMQ | Kafka |
| --- | --- | --- | --- | --- |
| 单机吞吐量 | 万级，比 RocketMQ、Kafka 低一个数量级 | 同 ActiveMQ | 10 万级，支撑高吞吐 | 10 万级，高吞吐，一般配合大数据类的系统来进行实时数据计算、日志采集等场景 |
| topic 数量对吞吐量的影响 | | | topic 可以达到几百/几千的级别，吞吐量会有较小幅度的下降，这是 RocketMQ 的一大优势，在同等机器下，可以支撑大量的 topic | topic 从几十到几百个时候，吞吐量会大幅度下降，在同等机器下，Kafka 尽量保证 topic 数量不要过多，如果要支撑大规模的 topic，需要增加更多的机器资源 |
| 时效性 | ms 级 | 微秒级，这是 RabbitMQ 的一大特点，延迟最低 | ms 级 | 延迟在 ms 级以内 |
| 可用性 | 高，基于主从架构实现高可用 | 同 ActiveMQ | 非常高，分布式架构 | 非常高，分布式，一个数据多个副本，少数机器宕机，不会丢失数据，不会导致不可用 |
| 消息可靠性 | 有较低的概率丢失数据 | 基本不丢 | 经过参数优化配置，可以做到 0 丢失 | 同 RocketMQ |
| 功能支持 | MQ 领域的功能极其完备 | 基于 erlang 开发，并发能力很强，性能极好，延时很低 | MQ 功能较为完善，还是分布式的，扩展性好 | 功能较为简单，主要支持简单的 MQ 功能，在大数据领域的实时计算以及日志采集被大规模使用 |
| 社区活跃度 | 低 | 中 | 高 | 高 |



## 如何确保消息正确地发送至 RabbitMQ？如何确保消息接收方消费了消息?

- 发送方确认模式
- 接收方确认机制

## RabbitMQ保障消息 100% 投递成功方案

- 消息落库，对消息状态进行打标
- 消息的延迟投递，做二次确认，回调检查

## 常见用来保证幂等的手段

- MVCC方案

多版本并发控制。该策略主要使用update with condition（更新带条件来防止）来保证多次外部请求调用对系统的影响是一致的。在系统设计的过程中，合理的使用乐观锁，通过version或者updateTime（timestamp）等其他条件，来做乐观锁的判断条件，这样保证更新操作即使在并发的情况下，也不会有太大的问题。

- 去重表

在插入数据的时候，插入去重表，利用数据库的唯一索引特性，保证唯一的逻辑

- token机制，防止页面重复提交
    - 业务要求：页面的数据只能被点击提交一次
      发生原因：由于重复点击或者网络重发，或者nginx重发等情况会导致数据被重复提交
    - 解决办法：
        - 集群环境：采用token加redis（redis单线程的，处理需要排队）
        - 单JVM环境：采用token加redis或token加jvm内存
    - 处理流程：
        - 数据提交前要向服务的申请token，token放到redis或jvm内存，token有效时间
        - 提交后后台校验token，同时删除token，生成新的token返回
    - token特点:要申请，一次有效性，可以限流
- 利用redis原子性

## Elastic-Job——分布式定时任务框架

elastic-job + zookeeper + mysql