rabbitmq

why rabbitmq?

RabbitMQ 支持非常灵活的消息传递模式，比如 优先级队列、死信队列、延迟队列 等，能够满足复杂的业务逻辑和需求。适用于需要处理高可靠性的消息传递和复杂的系统集成。

Kafka 和 **RocketMQ** 设计时更加偏向于 日志记录 和 流处理，通常不支持 RabbitMQ 中那么多的消息管理机制。

开源

RabbitMQ 具有非常高的可靠性，支持多种消息确认机制，如生产者确认、消费者确认等

我选择RabbitMQ是因为它具有强大的消息路由能力和灵活的交换机机制，非常适合需要复杂消息路由的场景。而RocketMQ和Kafka虽然性能强大，适合高吞吐量的应用，但对于消息路由和一些特定的应用场景，RabbitMQ提供了更高的灵活性和易用性。

> 结合项目

我们需要处理不同类型的消息，并根据不同的业务需求将消息发送到不同的队列。RabbitMQ的交换机（Exchange）机制非常适合这种情况，它允许我们根据消息的不同属性（比如路由键）将消息灵活地路由到不同的队列。比如，我们可以使用直接交换机（Direct Exchange）来精确路由消息，或者使用主题交换机（Topic Exchange）来根据模式匹配将消息路由到多个队列，这对于我们项目中的复杂消息流转非常有帮助。

延时队列的topic，还有和用户中心交互也是通过mq。包括更新插入es也是用的mq

这个背景是之前有一些单子客服没法区分审批先后，导致有一些申请单审批时间较久导致用户投诉。

后面组长让我做一个申请单超时通知功能

插入申报信息的地方，申报信息入表后返回一个主键id，这个id发消息到mq，然后消费方拿到这个id

反查一次数据库，查到数据后进行es的数据组装，然后插入es

> 交换机机制

交换机（Exchange）机制是其核心特性之一，用于决定消息的路由方式。它接受来自生产者的消息，并将这些消息转发到一个或多个队列中。交换机本身不存储消息，它只是将消息路由到符合条件的队列

如何保证消息不丢失

> 生产者确认机制

消息发送到MQ以后 会返回一个结果给发送者 表示消息是否处理成功

失败了？ 回调方法及时重发 、记录日志、保存到数据库然后定时重发，成功发送后即可删除表中的数据

> 消息持久化

MQ默认存内存

交换机持久化

队列持久化

消息持久化

> 消费者确认

MQ收到 消费者成功消息才会删除该消息

。而SpringAMQP则允许配置三种确认模式：

- manual：手动ack，需要在业务代码结束后，调用api发送ack。
- auto：自动ack，由spring监测listener代码是否出现异常，没有异常则返回ack；抛出异常则返回nack
- none：关闭ack，MQ假定消费者获取消息后会成功处理，因此消息投递后立即被删除

一般自动

失败了会重试 要设置

实在不行就会放到异常交换机里面 交由人工处理

重复消费问题如何解决

网络抖动

消费者挂了

MQ没有收到确认导致重复消费

解决方案：每个消息设置一个标识token 校验业务id是否存在再消费 （消费者拿到后 放redis 再消费）

幂等方案：分布式锁 数据库锁

延迟队列

延迟消费的队列

场景：超时订单、限时优惠、定时发布

延迟队列=死信交换机+TTL

> 死信和*TTL*

TTL 代表消息的生存时间，当消息的 TTL 时间到了还未被消费，就会变成死信（Dead Letter）。

队列级别 **TTL**（所有消息的 TTL 相同）

消息级别 **TTL**（每条消息的 TTL 可不同）

按最低的算

> 死信交换机

死信交换机是专门用来存储过期或被拒绝的消息的交换机。

哪些消息会进入死信队列？

1. **TTL** 过期（消息存活时间到期）
2. 队列已满（队列达到了最大长度）
3. 消息被拒绝（**basic.reject** 或 **basic.nack**） 且 未重新入队

> 为什么说是死信交换机+*TTL*

生产者将消息发送到普通队列，并设置 TTL（例如 10s）。

消息 10s 后过期，进入死信队列。

消费者监听死信队列，执行实际业务逻辑。

普通队列不绑定消费者，这样消息只能在 TTL 过期后进入死信队列（DLQ）

死信队列才绑定消费者，确保消息延迟到 TTL 之后才会被消费。

> 使用 ***RabbitMQ*** 的 ***Delayed Exchange*** 插件
>
> - 这个方式是 官方推荐的真正的延迟队列方案，它支持直接在交换机层面控制延迟，而不是利用 *TTL*

发送消息设置x-delay头和超时时间

通过一个 交换机 来暂存消息，然后在设定的 延迟时间到期后，将消息发送到 实际的队列中进行消费。这个方案利用了 **RabbitMQ** 的延迟交换机插件，通过交换机将消息“存储”一定时间，之后将消息转发到目标队列。

消息堆积问题

太多了 都成为死信

三种思路

1、more消费者

2、消费者拉满速度（线程池）

3、给队列扩容

> 惰性队列

直接持久化 从磁盘中拿 磁盘比内存大 （lazy加载）

高可用机制

普通集群 镜像集群 仲裁队列

> 普通集群

各个节点共享部分数据 包括交换机、队列元信息 (引用信息)

访问集群某节点，如果队列不在该节点，会从数据所在节点传递到当前节点并返回

队列所在节点宕机，那就寄了

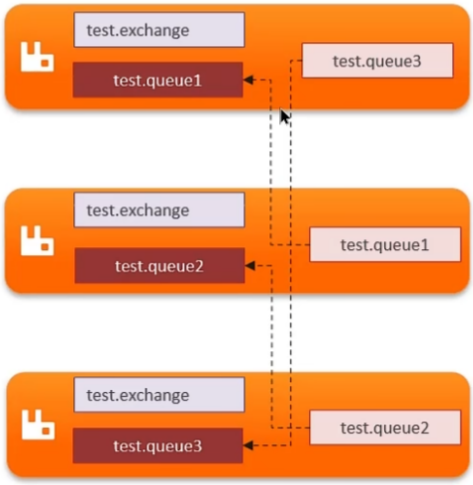
> 镜像集群

本质是主从模式

跟redis差不多

镜像集群：本质是主从模式，具备下面的特征：

- 交换机、队列、队列中的消息会在各个mq的镜像节点之间同步备份。
- 创建队列的节点被称为该队列的**主节点**，备份到的其它节点叫做该队列的**镜像**节点。
- 一个队列的主节点可能是另一个队列的镜像节点
- 所有操作都是主节点完成，然后同步给镜像节点
- 主宕机后，镜像节点会替代成新的主



> 仲裁队列

主从模式　同步基于raft协议，强一致

> *raft*协议

Raft协议是一种用于分布式系统的共识算法，它能够帮助集群中的多个节点就某个操作达成一致，保证在一些节点发生故障的情况下，系统依然能够正确工作。Raft协议的目标是为分布式系统提供一种易于理解且强一致性的协议，解决分布式系统中如何确保一致性和数据可靠性的问题。

Raft协议提供的是强一致性，即所有节点的日志必须一致，客户端只能在数据被多数节点确认提交后才能返回响应。

存日志

当大多数节点都确认接收到该日志条目并且日志已被提交后，领导者才会向客户端返回响应，表示操作已经成功执行。

顺序性

多个消费者和多线程问题

同一个队列是有序的 先进先出

生产有序

消费有序

一个队列 一个消费者

用hashkey分到不同queue里面 （例如同一个用户的消息在同一个队列）

怎么说 其实就是同一个用户的相关消息要完整执行 不能被分开 细究下去 还是没办法保证的