# **[RabbitMQ几个常用面试题](https://www.cnblogs.com/woadmin/p/10537174.html)**

以下观点，仅为个人理解的总结，如有错漏，欢迎指正！

---------------------------------------------------------------------------------------------------------------------

1、什么是RabbitMQ？为什么使用RabbitMQ？

答：RabbitMQ是一款开源的，Erlang编写的，基于AMQP协议的，消息中间件；

可以用它来：解耦、异步、削峰。

2、RabbitMQ有什么优缺点？

答：优点：解耦、异步、削峰；

缺点：降低了系统的稳定性：本来系统运行好好的，现在你非要加入个消息队列进去，那消息队列挂了，你的系统不是呵呵了。因此，系统可用性会降低；

增加了系统的复杂性：加入了消息队列，要多考虑很多方面的问题，比如：一致性问题、如何保证消息不被重复消费、如何保证消息可靠性传输等。因此，需要考虑的东西更多，复杂性增大。

3、如何保证RabbitMQ的高可用？

答：没有哪个项目会只用一搭建一台RabbitMQ服务器提供服务，风险太大；

4、如何保证RabbitMQ不被重复消费？

答：先说为什么会重复消费：正常情况下，消费者在消费消息的时候，消费完毕后，会发送一个确认消息给消息队列，消息队列就知道该消息被消费了，就会将该消息从消息队列中删除；

但是因为网络传输等等故障，确认信息没有传送到消息队列，导致消息队列不知道自己已经消费过该消息了，再次将消息分发给其他的消费者。

针对以上问题，一个解决思路是：保证消息的唯一性，就算是多次传输，不要让消息的多次消费带来影响；保证消息等幂性；

比如：在写入消息队列的数据做唯一标示，消费消息时，根据唯一标识判断是否消费过；

5、如何保证RabbitMQ消息的可靠传输？

答：消息不可靠的情况可能是消息丢失，劫持等原因；

丢失又分为：生产者丢失消息、消息列表丢失消息、消费者丢失消息；

生产者丢失消息：从生产者弄丢数据这个角度来看，RabbitMQ提供transaction和confirm模式来确保生产者不丢消息；

transaction机制就是说：发送消息前，开启事务（channel.txSelect()）,然后发送消息，如果发送过程中出现什么异常，事务就会回滚（channel.txRollback()）,如果发送成功则提交事务（channel.txCommit()）。然而，这种方式有个缺点：吞吐量下降；

confirm模式用的居多：一旦channel进入confirm模式，所有在该信道上发布的消息都将会被指派一个唯一的ID（从1开始），一旦消息被投递到所有匹配的队列之后；

rabbitMQ就会发送一个ACK给生产者（包含消息的唯一ID），这就使得生产者知道消息已经正确到达目的队列了；

如果rabbitMQ没能处理该消息，则会发送一个Nack消息给你，你可以进行重试操作。

消息队列丢数据：消息持久化。

处理消息队列丢数据的情况，一般是开启持久化磁盘的配置。

这个持久化配置可以和confirm机制配合使用，你可以在消息持久化磁盘后，再给生产者发送一个Ack信号。

这样，如果消息持久化磁盘之前，rabbitMQ阵亡了，那么生产者收不到Ack信号，生产者会自动重发。

那么如何持久化呢？

这里顺便说一下吧，其实也很容易，就下面两步

1. 将queue的持久化标识durable设置为true,则代表是一个持久的队列
2. 发送消息的时候将deliveryMode=2

这样设置以后，即使rabbitMQ挂了，重启后也能恢复数据

消费者丢失消息：消费者丢数据一般是因为采用了自动确认消息模式，改为手动确认消息即可！

消费者在收到消息之后，处理消息之前，会自动回复RabbitMQ已收到消息；

如果这时处理消息失败，就会丢失该消息；

解决方案：处理消息成功后，手动回复确认消息。

6、如何保证RabbitMQ消息的顺序性？

答：单线程消费保证消息的顺序性；对消息进行编号，消费者处理消息是根据编号处理消息；