

RabbitMQ

月薪上万如此简单~

目 录

1

确认机制

2

死信队列

3

延迟队列

4

RabbitMQ 实战场景

5

RabbitMQ集群



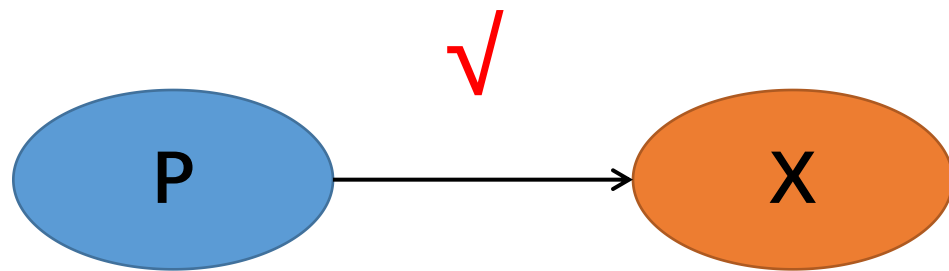
确认机制

- 发布确认机制
- 回退机制
- 备份交换机
- 消费者确认机制

一、确认机制

1.1 生产投递丢失问题

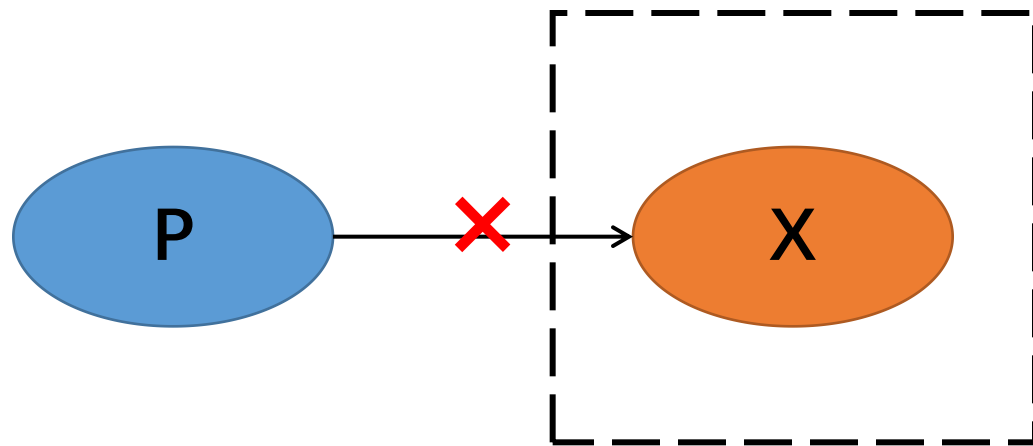
如果生产者 P 投递消息到交换机 X 的过程中，出现了网络延迟，导致消息丢失，怎么保证消息安全？



一、确认机制

1.1 生产投递丢失问题

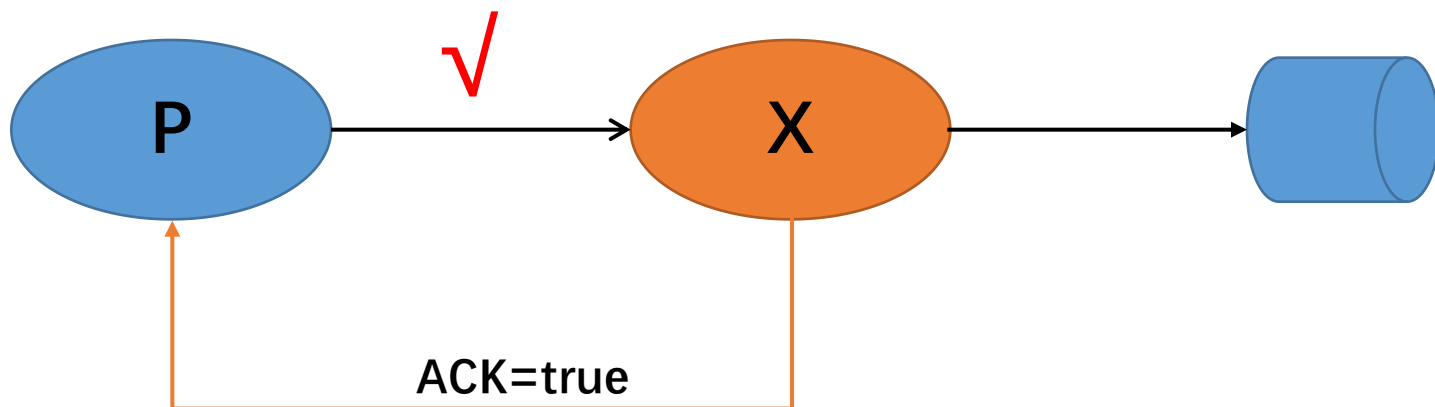
如果生产者 P 投递消息到交换机 X 的过程中，出现了网络延迟，导致消息丢失，怎么保证消息安全？



一、确认机制

1.2 通过发布确认机制解决

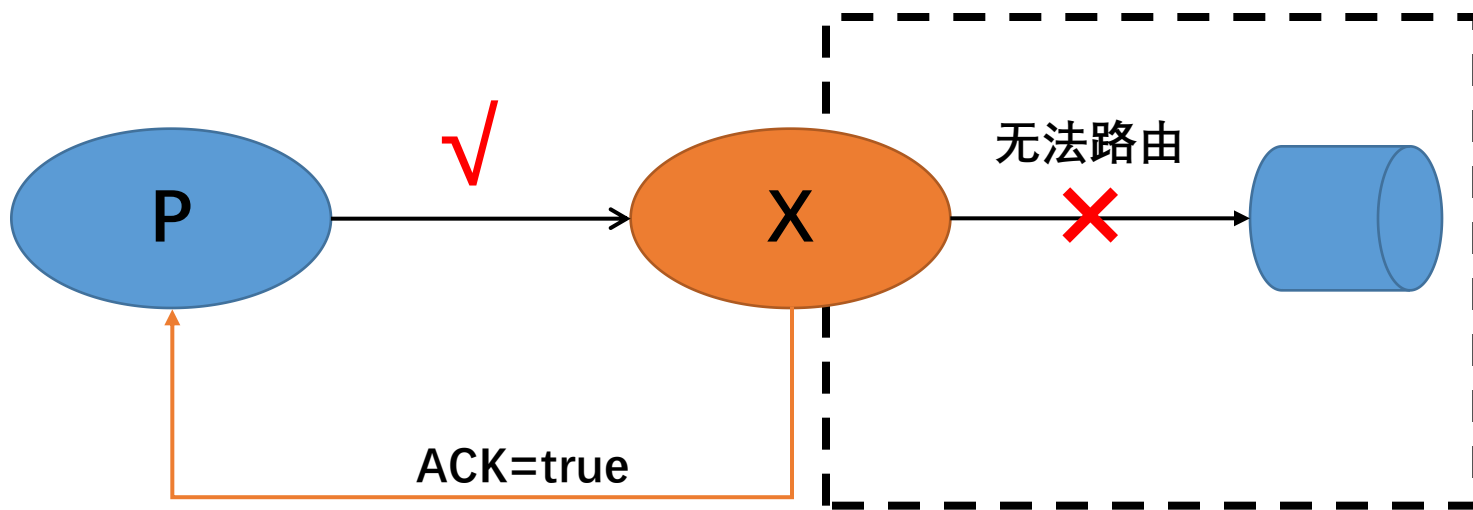
生产者 P 投递消息到交换机 X 的过程中，交换机 X 会给生产者 P 一个 ACK 确认回调，生产者可以根据收到 ACK 值知道是否投递成功。



一、确认机制

1.3 交换机无法路由问题

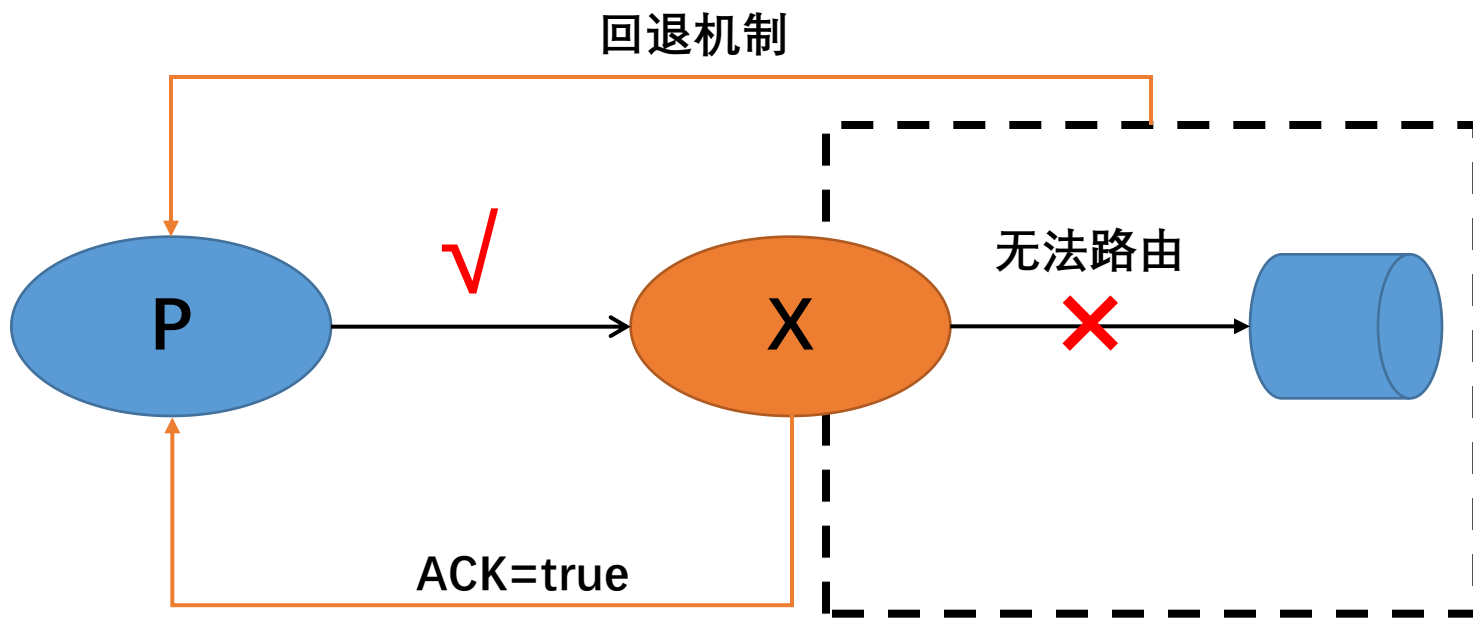
当生产者 P 投递消息到交换机 X 的过程中，消息确定收到了，但是路由配置错误，或者没有绑定队列，此时又如何保证消息安全性？



一、确认机制

1.4.1 回退机制让生产者自行处理

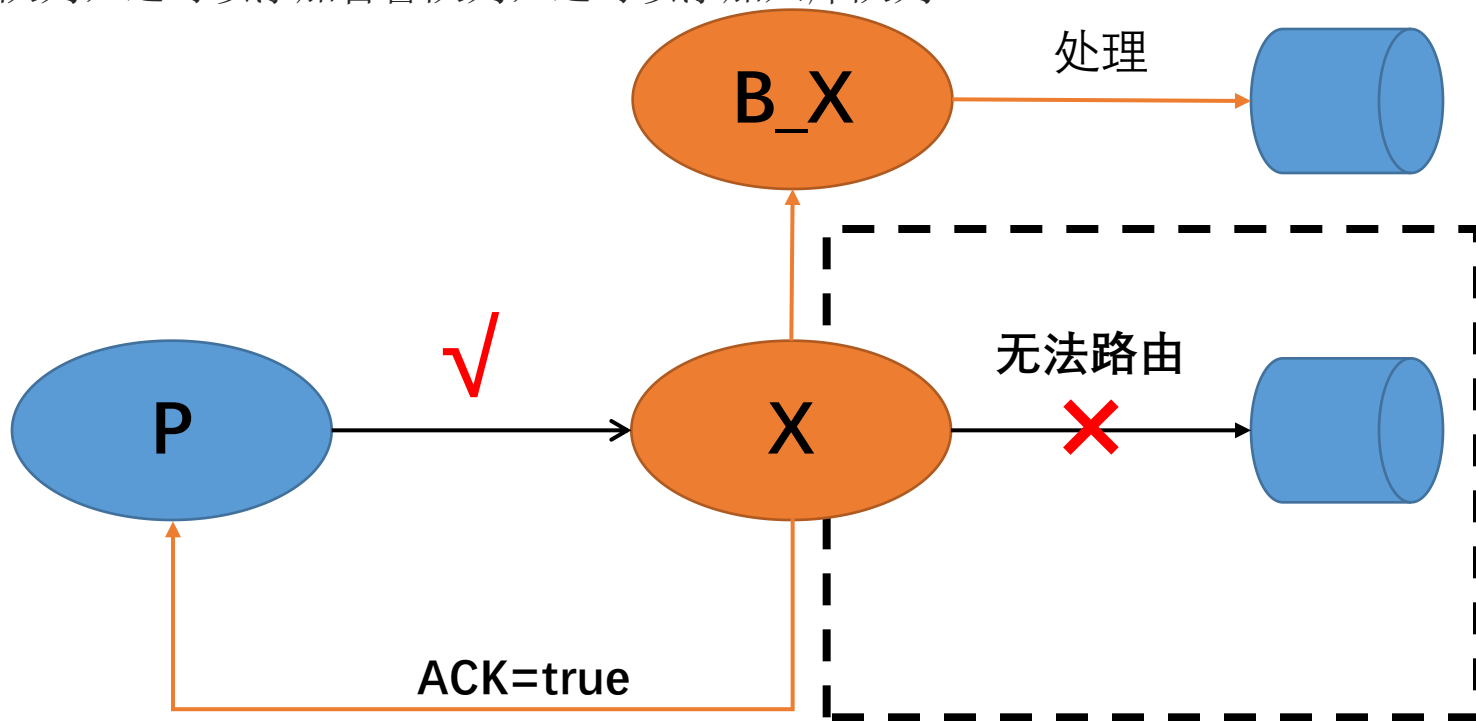
仅仅开启确认机制无法保证消息安全性，可以通过回退机制，通知消费者此条消息无法处理，让消费者自行处理消息



一、确认机制

1.4.2 备份交换机解决

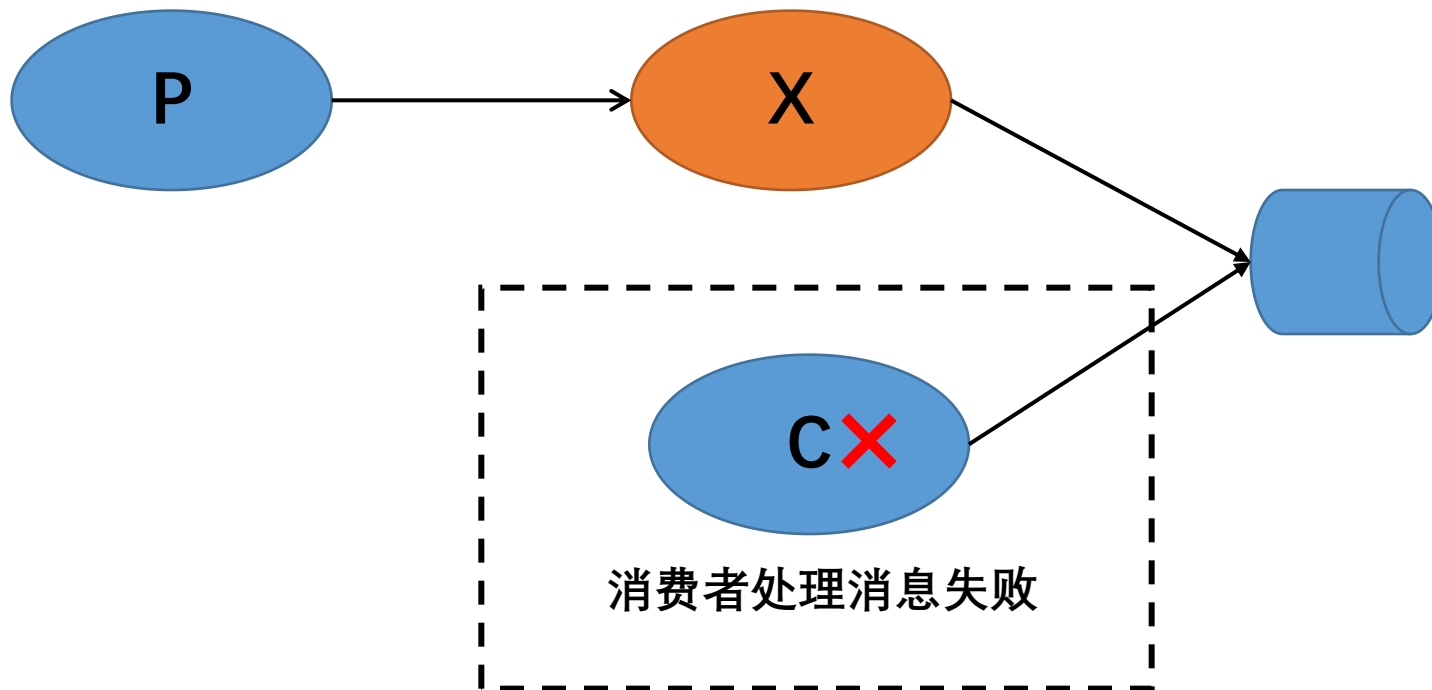
可以通过给交换机设置备份机的方式，来处理交换机无法路由的消息，备份交换机设置 Fanout 类型，可以添加备份队列，还可以添加告警队列，还可以添加入库队列。



一、确认机制

1.5.1 消费者异常导致数据丢失

如果消费者处理的过程中发生异常，导致消息丢失怎么办？



一、确认机制

1.5.2 消费者确认机制

通过消费者确认机制避免消息丢失：

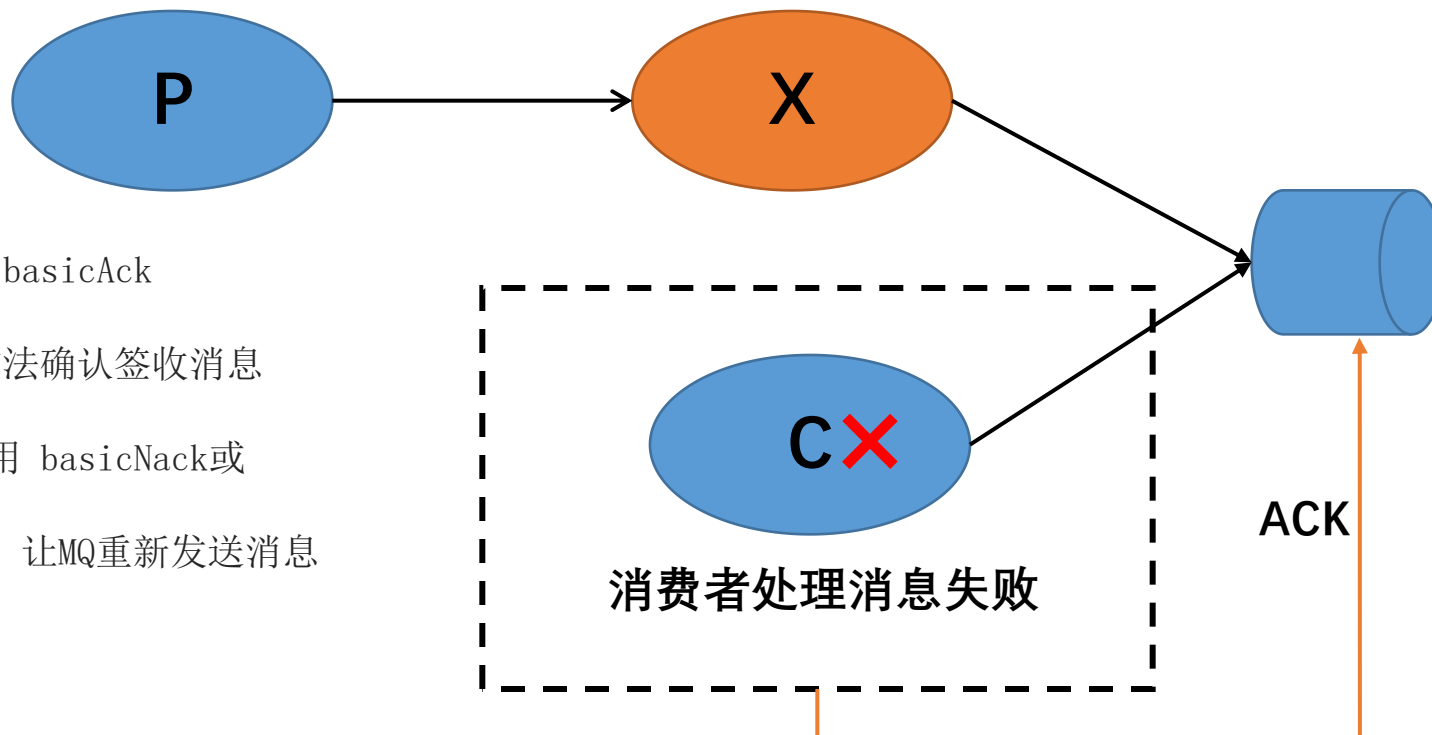
有三种确认方式：①自动确认：acknowledge=none ②手动确认：acknowledge=manual ③根据异常情况确认：acknowledge=auto

手动确认：

正常执行：调用channel.basicAck

(deliveryTag, false);方法确认签收消息

异常执行：在catch中调用 basicNack或
basicReject，拒绝消息，让MQ重新发送消息





死信队列

- 死信队列概述
- 死信队列架构图
- 死信队列实战

二、死信队列

1. 死信队列概述

死信：在官网中对应的单词为“Dead Letter”，可以看出翻译确实非常的简单粗暴。死信是RabbitMQ的一种消息机制，如果出现以下情况消息将变成死信：

1. 消息在队列的存活时间超过设置的生存时间（TTL）时间
2. 消息队列的消息数量已经超过最大队列长度
3. 消息被否定确认，使用basicNack或basicReject并且requeue=false

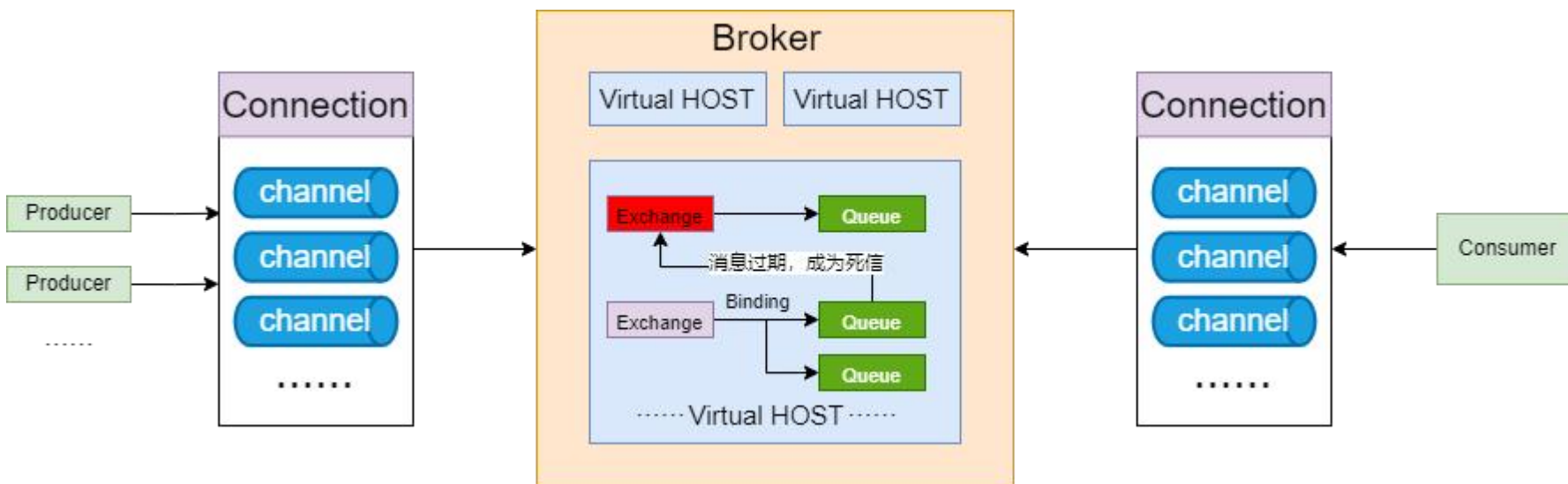
死信队列：Dead Letter Queue用来存放死信消息的队列

死信交换机：Dead Letter Exchange用来路由死信消息的交换机

二、死信队列

2. 死信队列架构图

消息满足刚提到的几个条件之后，消息进入死信队列



二、死信队列

3. TTL

TTL概述：用来控制消息或者是队列的最大存活时间，单位是毫秒。

设置TTL两种方式：

1：给消息设置TTL

2：给队列设置TTL

注意：如果同时配置了队列的 TTL 和消息的 TTL，那么较小的那个值将会被使用。

区别：

1：消息设置TTL的方式是在消费时做消息过期判断，如果是顶端的，到期直接移除

2：队列设置过期是直接丢弃或丢到死信队列

二、死信队列

案例



3. 死信队列实操

1. 达到最大长度进入死信队列
 - 限制队列最大长度为10，发送11条消息给正常队列
2. 到达过期时间进入死信队列
 - 设置10秒过期时间
3. 拒收进入死信队列
 - 模拟异常拒收

二、死信队列



小结

1. 死信交换机和死信队列和普通的没有区别
2. 当消息成为死信后，如果该队列绑定了死信交换机，则消息会被死信交换机重新路由到死信队列
3. 消息成为死信的三种情况：
 - 消息在队列的存活时间超过设置的生存时间（TTL）时间
 - 消息队列的消息数量已经超过最大队列长度
 - 消息被否定确认，使用basicNack或basicReject并且requeue=false



延时队列

- 延时队列的概述
- 延时队列的实现
- 延时队列插件
- 案例实现

三、延时队列

1. 延时队列概述

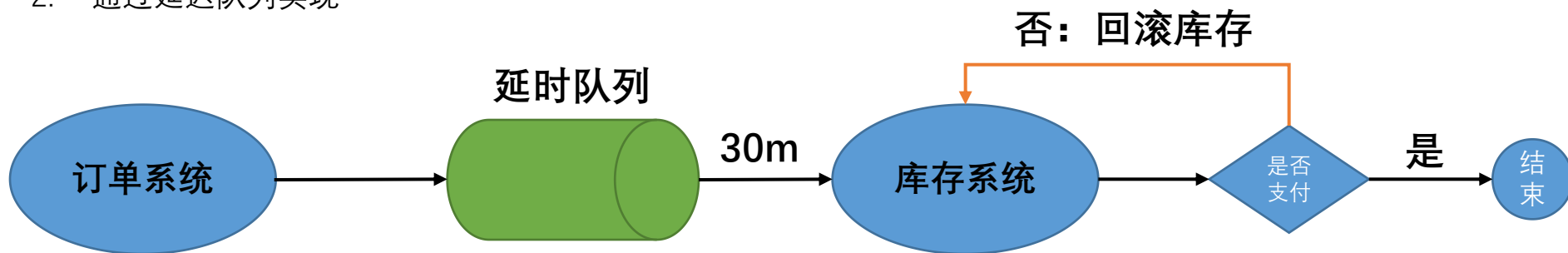
延时队列：消息进入队列后不会立即被消费，只有到达指定时间后，才会被消费，最重要的特征就是延迟上

应用场景：

1. 12306购票30分钟内不支付自动取消订单，回滚库存
2. 会议设置，15分钟前通知参会人员

应用场景：

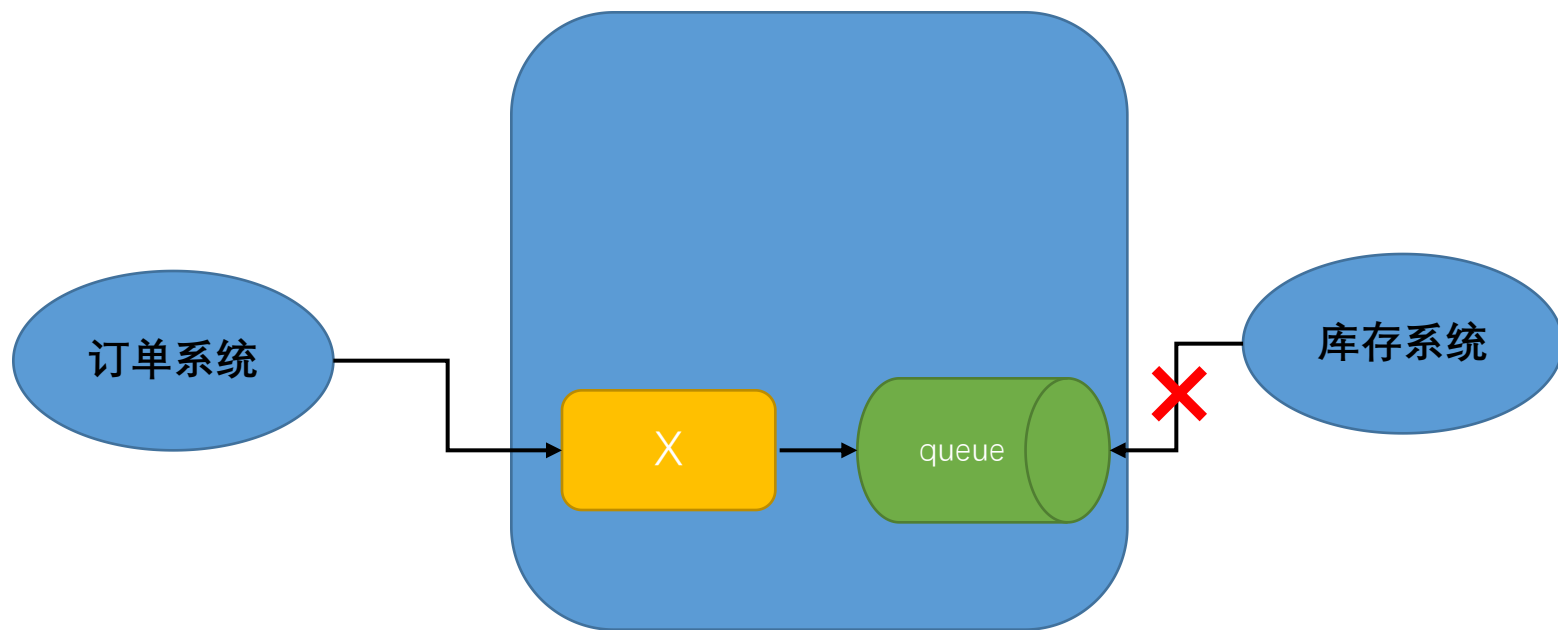
1. 通过轮询数据库查询处理
2. 通过延迟队列实现



三、延时队列

2. 延时队列实现

RabbitMQ未提供延时队列功能，但是我们可以通过 TTL + 死信队列的方式设计出延时队列

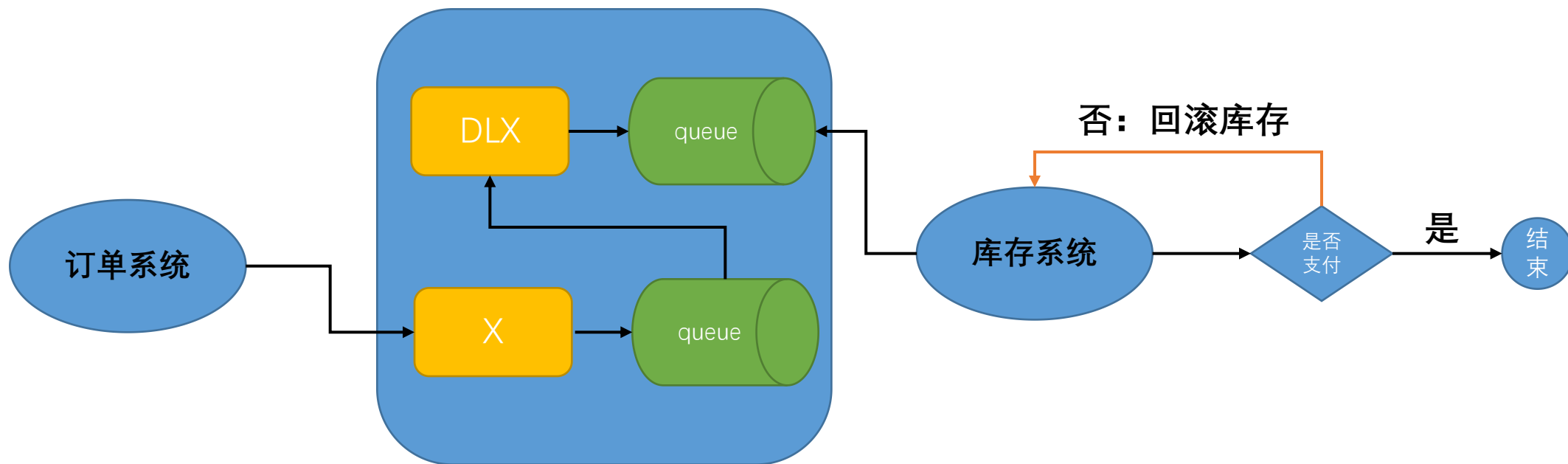


三、延时队列

2. 延时队列实现

RabbitMQ未提供延时队列功能，但是我们可以通过 TTL + 死信队列的方式设计出延时队列

写代码实现（发现消费不能按最近过期时间消费问题）



三、延时队列

3. 安装插件实现延时队列

下载地址 <https://pypi.org/project/rabbitmq-delayed-message-exchange/3.8.0/>

上传位置 <https://pypi.org/project/rabbitmq-delayed-message-exchange/3.8.34/>

启用插件 `rabbitmq-plugins enable rabbitmq_delayed_message_exchange`

重启服务

The image shows two screenshots of the RabbitMQ management console's 'Add a new exchange' dialog. The left screenshot shows the 'Type' dropdown menu open, displaying options: direct, fanout, headers, and topic. The right screenshot shows the same dialog, but the 'Type' dropdown menu is open and 'x-delayed-message' is selected and highlighted with a red box. The 'Name' field is empty in both. The 'Durability' field is set to 'direct'. The 'Auto delete' field has a question mark. The 'Internal' field has a question mark. The 'Arguments' field is empty in both.

三、延时队列

案例



12306实现30分钟内未付款自动取消订单回滚库存

1. 创建订单系统

- 发送订单进延迟队列


2. 创建库存系统

- 接收延时队列中的数据做业务处理，如果已经支付不做任何操作，如果没有支付则取消订单回滚库存

三、延迟队列



小结

1. 延迟队列就是指消息进入队列后需要一定时间才消费
2.  MQ本身是没有提供延迟队列功能，可以通过死信队列+过期时间来实现延迟队列
3. 自己实现的延迟队列不能按时消费，可能存在滞后问题，安装延时队列插件使用



RabbitMQ实战场景

- 日志与可视化监控查看
- 消息追踪
- 幂等性保障

四、RabbitMQ实战场景

1. 日志与可视化监控查看

日志文件：

`/var/log/rabbitmq/rabbitmq.log`

通过如下操作观察日志：

1. 停止服务：`sudo systemctl stop rabbitmq-server`

2. 重新启动：`sudo systemctl start rabbitmq-server`

3. 开启服务：`sudo systemctl enable rabbitmq-server`

四、RabbitMQ实战场景

2. 消息追踪

1. Firehose: 生产者给交换机发送消息时, 会按照指定的格式发送到amq.rabbitmq.trace (Topic) 交换机上。

开启Firehose命令: `rabbitmqctl trace_on`

关闭Firehose命令: `rabbitmqctl trace_off`

注意: 开启Firehose 会影响性能, 不建议一直打开

2. rabbitmq_tracing: 启用插件来实现可视化查看

查看插件: `rabbitmq-plugins list`

启用插件: `rabbitmq-plugins enable rabbitmq_tracing`

四、RabbitMQ实战场景

3. 幂等性保障

幂等性：是分布式中比较重要的一个概念，是指在多作业操作时候避免造成重复影响，其实就是保证同一个消息不被消费者重复消费两次。但是实际开发中可能存在网络波动等问题，生产者无法接受消费者发送的ack信息，因此这条消息将会被重复发送给其他消费者进行消费，实际上这条消息已经被消费过了，这就是重复消费的问题。

如何去避免重复消费问题：

1. 数据库乐观锁机制

```
update items set count=count-1 where count= #{count} and id = #{id}
```

```
update items set count=count-1,version=version+1 where version=#{version} and id = #{id}
```

2. 生成全局唯一id+redis锁机制:操作之前先判断是否抢占了分布式锁 setNx 命名



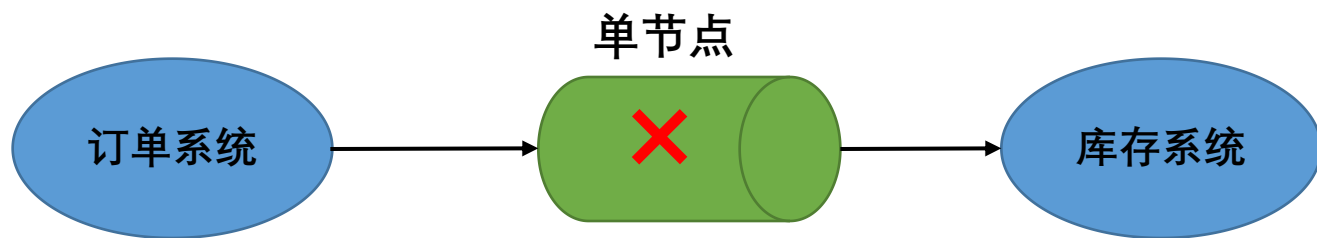
RabbitMQ 集群

- 搭建集群
- 搭建haproxy
- Springboot访问集群

五、RabbitMQ集群

1. 目前存在的问题

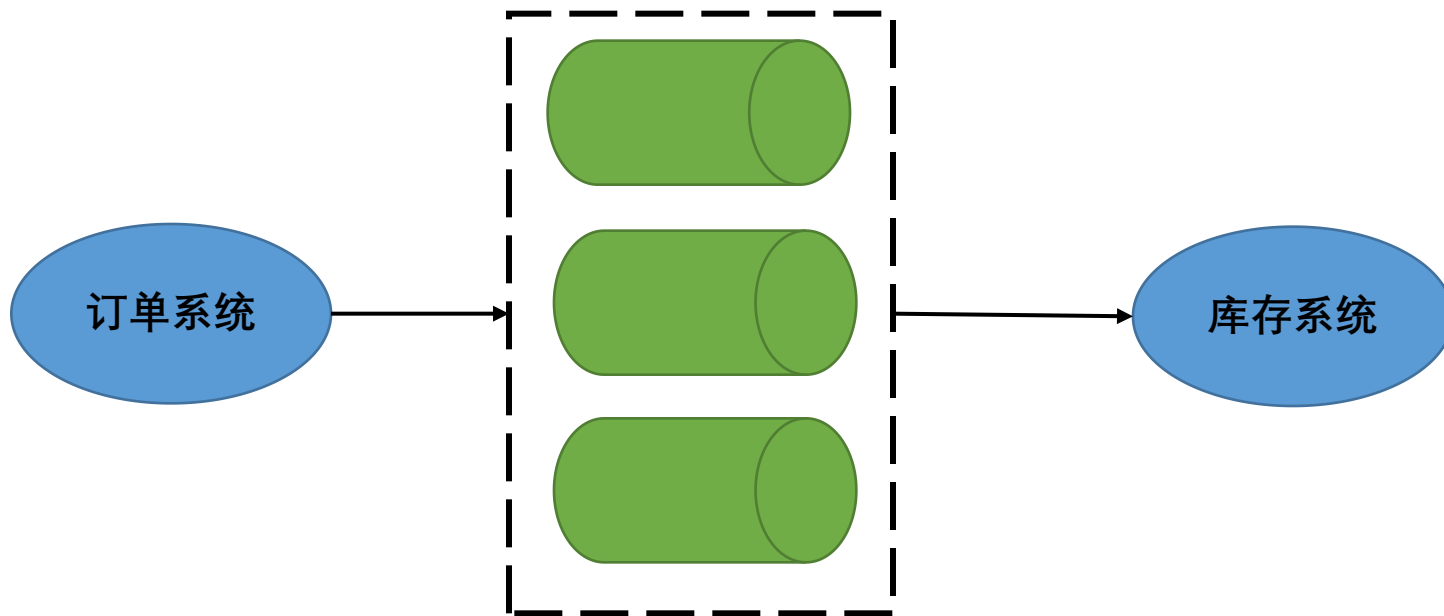
- 单节点的RabbitMQ如果内存崩溃、机器掉电或者主板故障，会影响整个业务线正常使用
- 单机吞吐性能会受内存、带宽大小限制



五、RabbitMQ集群

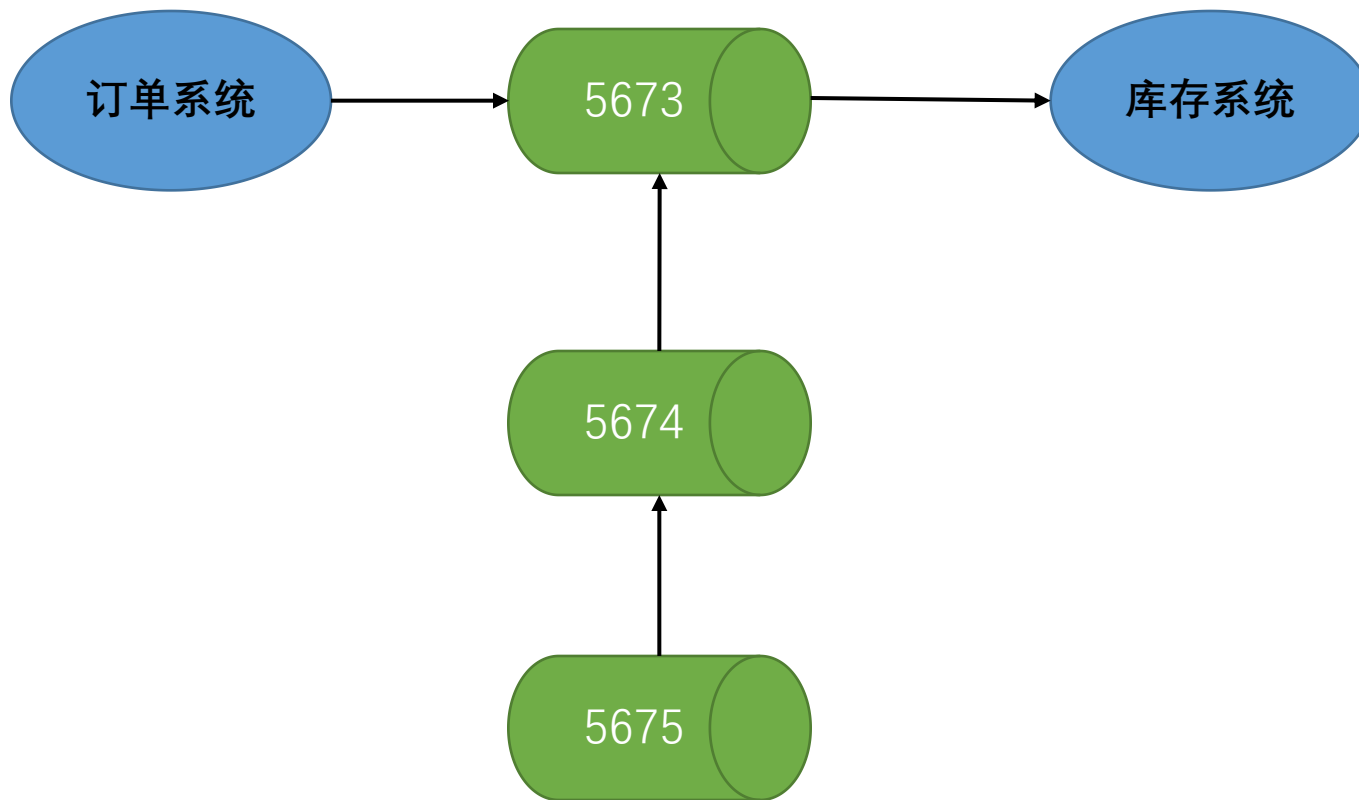
1. 目前存在的问题

- 单节点的RabbitMQ如果内存崩溃、机器掉电或者主板故障，会影响整个业务线正常使用
- 单机吞吐性能会受内存、带宽大小限制
- 解决方案：使用集群模式来解决



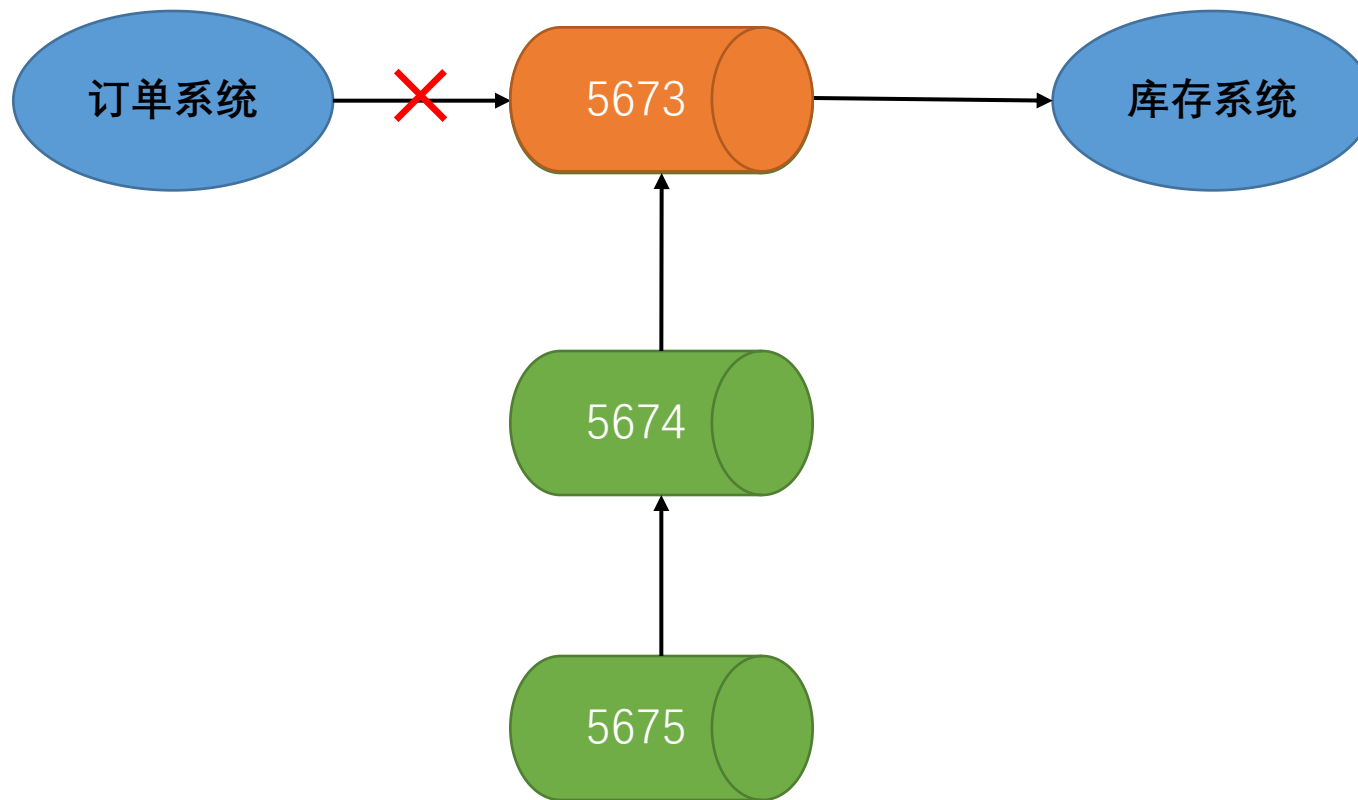
五、RabbitMQ集群

2. 搭建RabbitMQ集群



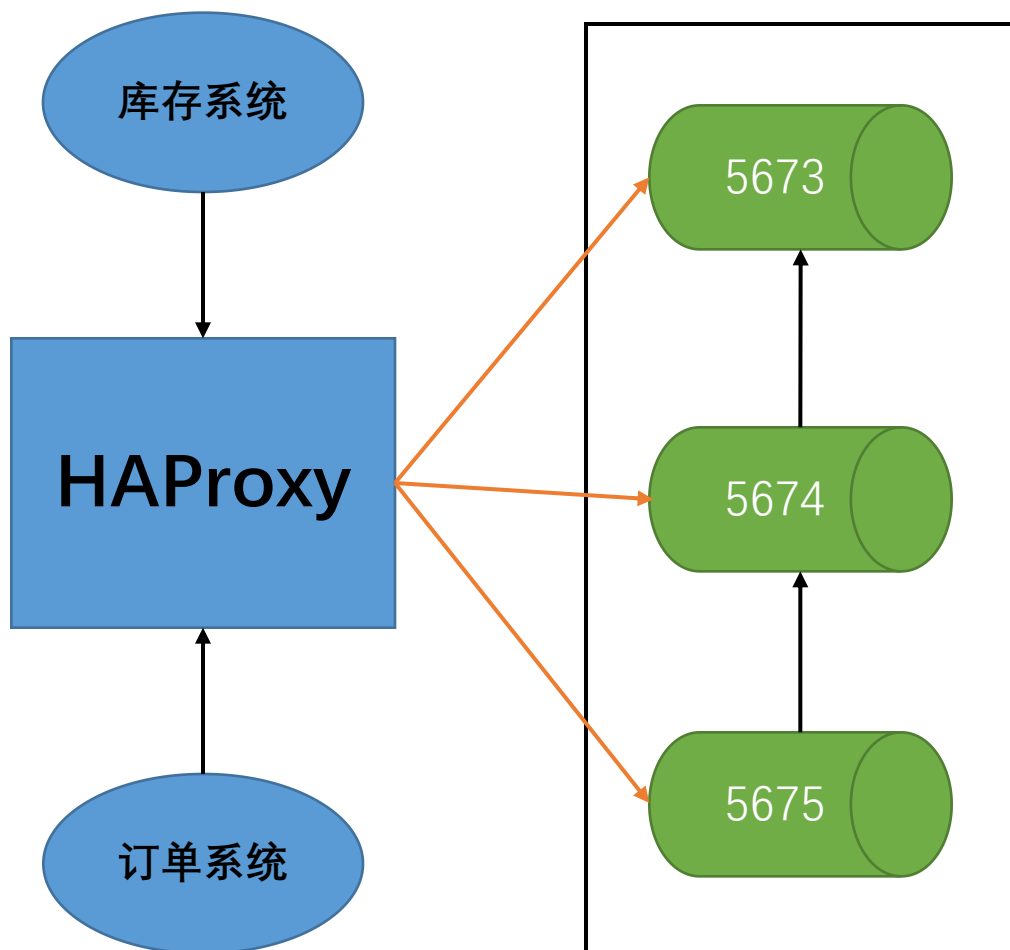
五、RabbitMQ集群

3. RabbitMQ集群存在问题



五、RabbitMQ集群

4. RabbitMQ集群解决





月薪上万如此简单