消息队列 CMQ

迁移说明

产品文档

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

文档目录

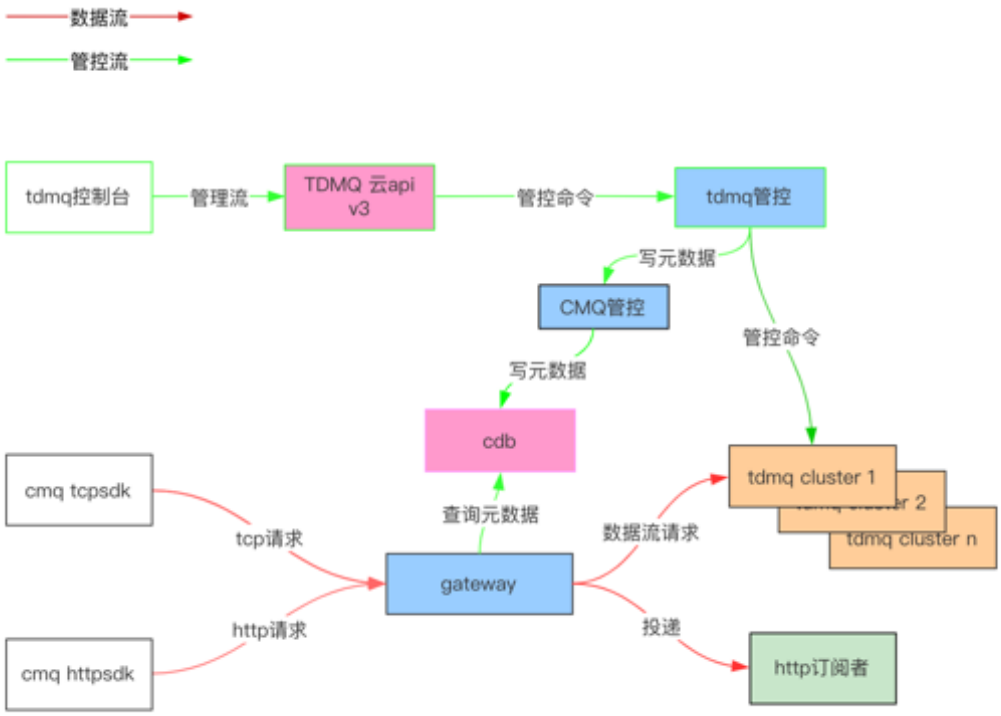
迁移说明
CMQ 队列迁移至 TDMQ CMQ 版

最近更新时间：2021-09-23 10:38:32

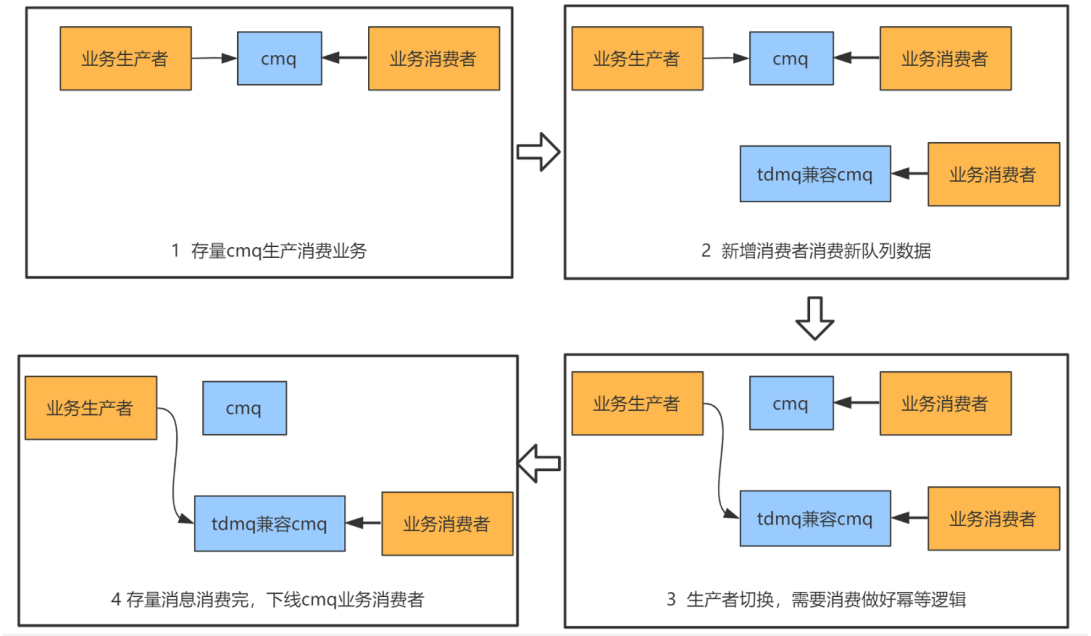
操作场景

本文以一个 Java 客户端为例，为您介绍将 CMQ 队列迁移至 TDMQ CMQ 版的操作步骤。

迁移原理



方案总览



整体流程

1. 在控制台上将 CMQ 的队列和主题元数据迁移至 TDMQ CMQ 版中。
2. 旧的消费者保持不动，消费端新建消费者，接入到 TDMQ CMQ 版的队列中。
3. 生产者停止向原 CMQ 队列生产消息，并切换生产流接入到 TDMQ CMQ 版队列中。
4. 旧的消费者继续消费原 CMQ 队列中的存量消息，消费完成后下线 CMQ 业务消费者。

前提条件

参考 SDK 文档部署好 CMQ 队列的生产端和消费端服务，并且运行正常。以下迁移步骤以 TCP SDK 为例。

操作步骤

步骤1：迁移元数据

1. 登录 CMQ 控制台。
2. 在左侧导航栏选择**队列**，选择好**地域**后，单击页面上方的**同步到 TDMQ**。

3. 在弹出的窗口中单击**启动**，将该地域下的所有队列和主题元数据迁移至 TDMQ CMQ 版中。

4. 迁移完成后，登录 TDMQ 控制台。
5. 在左侧导航栏选择**队列服务**，选择相同的**地域**可看到迁移到 TDMQ CMQ 版的队列。

步骤2：新建消费者

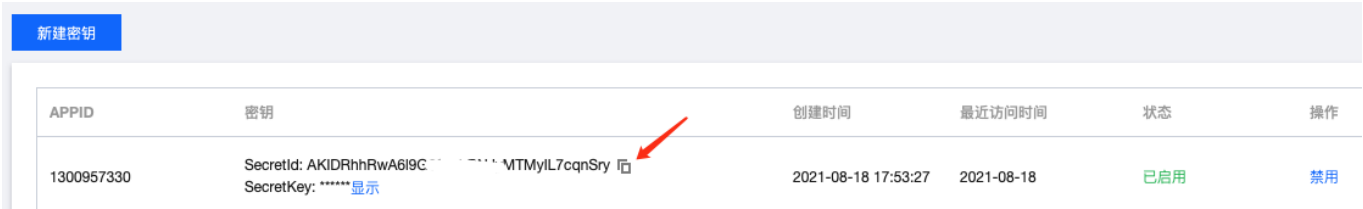
1. 在消费端新建一个消费者，并接入到 TDMQ CMQ 版队列中。

```
Consumer consumer = new Consumer();
// 私有网络地址： http://{region}.mqadapter.cmq.tencentyun.com 支持腾讯云私有网络的云服务器内网访问
// 公网地址： https://cmq-{region}.public.tencenttdmq.com
consumer.setNameServerAddress("http://****.com");
// 设置SecretId，在控制台上获取，必须设置
consumer.setSecretId("****");
// 设置SecretKey，在控制台上获取，必须设置
consumer.setSecretKey("****");
// 设置签名方式，可以不设置，默认为SHA1
```

```
consumer.setSignMethod(ClientConfig.SIGN_METHOD_SHA256);
// 批量拉取时最大拉取消息数量，范围为1-16
consumer.setBatchPullNumber(16);
// 设置没有消息时等待时间，默认10s。可在consumer.receiveMsg等方法中传入具体的等待时间
consumer.setPollingWaitSeconds(6);
// 设置请求超时时间， 默认3000ms
// 如果设置了没有消息时等待时间为6s，超时时间为5000ms，则最终超时时间为(6*1000+5000)ms
consumer.setRequestTimeoutMS(5000);
// 消息拉取的队列名称
final String queue = "****";
```

- NameServerAddress：API 调用地址，在 TDMQ CMQ 版控制台 的**队列服务 > API请求地址**处复制。

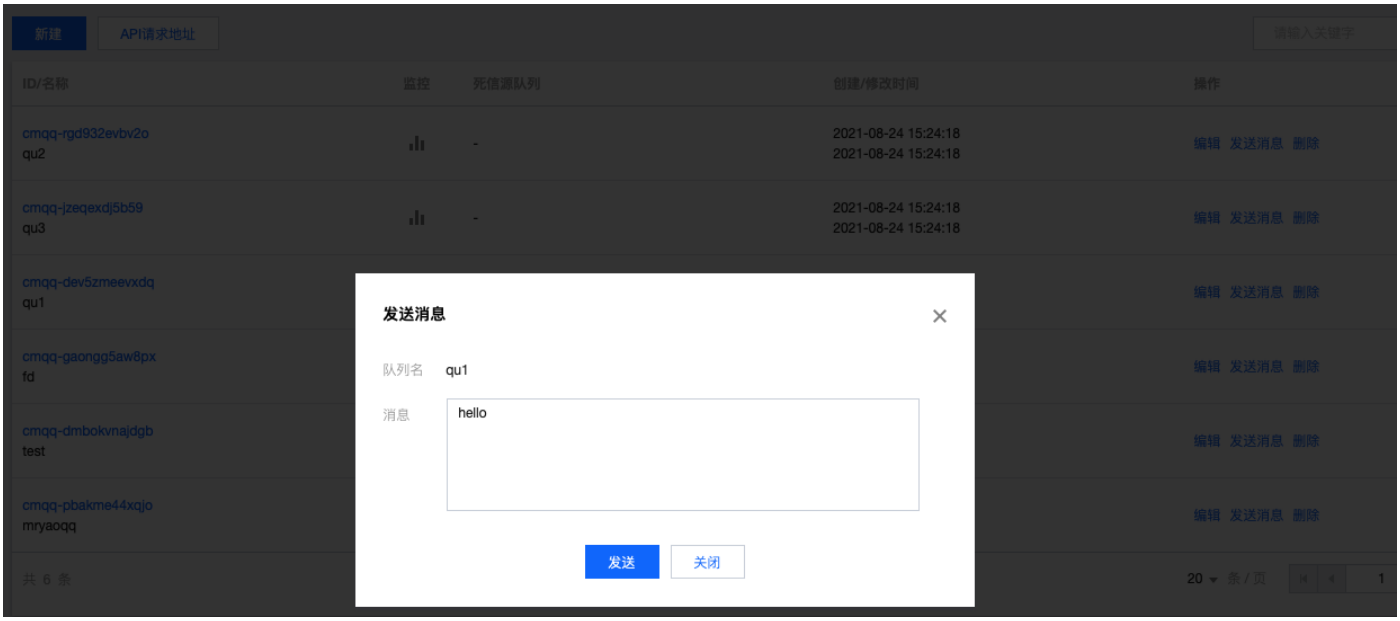
- SecretId、SecretKey：云API密钥，登录 访问管理控制台，在**访问密钥 > API密钥管理**页面复制。



- queue：填写队列名称。

2. 运行代码，查看消费端服务是否能正常运行无报错。

3. 通过 TDMQ CMQ 版控制台 的**队列服务** > **发送消息**向消息接收侧发送测试消息，验证消费者服务是否可以正常消费。



如图则为正常消费：

```
2021-08-27 14:29:02,842 INFO  NettyClient.java(686) : createChannel: begin to connect remote host[119.2      .12000] asynchronously
2021-08-27 14:29:02,856 INFO  NettyClient.java(780) : NETTY CLIENT PIPELINE: CONNECT  UNKNOWN => 119.      3:12000
queue:[qu1] push msg:[msgId=281474976710700, receiptHandle=8070800181945072642, data=tre张测试]
queue:[qu1] push msg:[msgId=281474976710701, receiptHandle=8070812129948242365, data=tre张测试]
queue:[qu1] push msg:[msgId=281474976710702, receiptHandle=8070939274614765827, data=tre张测试]
queue:[qu1] push msg:[msgId=281474976710703, receiptHandle=8070744555126003411, data=hello]
queue:[qu1] push msg:[msgId=281474976710704, receiptHandle=8070732823471456636, data=world]
queue:[qu1] push msg:[msgId=281474976710705, receiptHandle=8070801844779515656, data=hello]
queue:[qu1] push msg:[msgId=281474976710706, receiptHandle=8070951493814685501, data=world]
```

步骤3：切换生产流

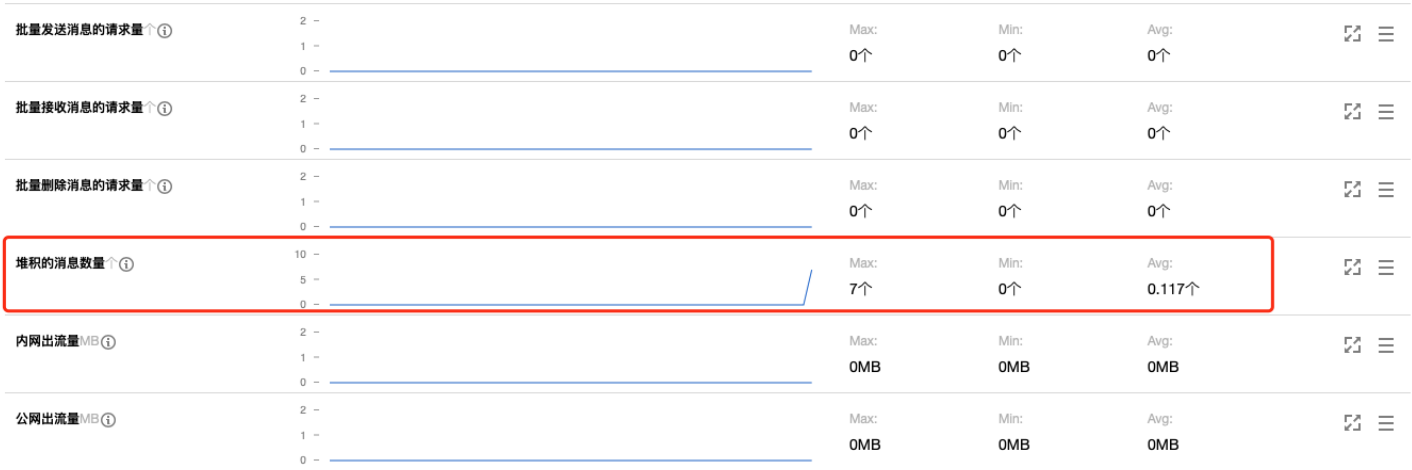
1. 将原生产者的 NameServer 修改为 TDMQ CMQ 版队列的接入地址，在 TDMQ CMQ 版控制台的**队列服务** > **API请求地址**处复制。
2. 运行生产消息程序，验证生产者服务是否可以正常发送消息。

```
2021-08-24 15:09:16,064 INFO  NettyClient.java(686) : createChannel: begin to connect remote host[119.27.176.143:12000] asynchronously
2021-08-24 15:09:16,078 INFO  NettyClient.java(780) : NETTY CLIENT PIPELINE: CONNECT  UNKNOWN => 119.27.176.143:12000
==> send success! msg_id:281474976710657 request_id:1
==> send success! msg_id:281474976710658 request_id:3
==> send success! msg_id:281474976710659 request_id:4
==> send success! msg_id:[281474976710660, 281474976710661] request_id:3
==> send success! msg_id:[281474976710662, 281474976710663] request_id:6
2021-08-24 15:09:21,646 INFO  NettyClient.java(593) : closeChannel: begin close the channel[119.27.176.143:12000] Found: true
2021-08-24 15:09:21,647 INFO  NettyClient.java(606) : closeChannel: the channel[119.27.176.143:12000] was removed from channel table
2021-08-24 15:09:21,648 INFO  NettyClient.java(795) : NETTY CLIENT PIPELINE: CLOSE 119.27.176.143:12000
2021-08-24 15:09:21,649 INFO  MQClientInstance.java(113) : the client instance [10.91.78.131@20037] shutdown OK
```

步骤4：下线旧消费者

等待旧的消费者继续消费完原 CMQ 队列中的存量消息后，下线 CMQ 业务消费者。

在 CMQ 控制台 的**队列服务** > **队列** > **监控**页面可查看 CMQ 队列中堆积的消息数量，当堆积消息数量为0时，代表原 CMQ 队列中的存量消息已被消费完成。


批量发送消息的请求量 个
Max: 0个
Min: 0个
Avg: 0个
批量接收消息的请求量 个
Max: 0个
Min: 0个
Avg: 0个
批量删除消息的请求量 个
Max: 0个
Min: 0个
Avg: 0个
堆积的消息数量 个
Max: 7个
Min: 0个
Avg: 0.117个
内网出流量 MB
Max: 0MB
Min: 0MB
Avg: 0MB
公网出流量 MB
Max: 0MB
Min: 0MB
Avg: 0MB

迁移常见问题

最近更新时间：2022-05-31 17:53:30

以下 TDMQ CMQ 版简称为新版 CMQ，当前的消息队列 CMQ 简称为原 CMQ。

基础数据迁移相关

问题场景：

单击 CMQ 界面上的基础数据后发现新 CMQ 订阅的数量和原 CMQ 的订阅数量不符。

问题说明：

其实相差的这部分是曾经在原 CMQ 已经被人为删除了的队列，而原 CMQ 在查看订阅时不会严格过滤已经删除的队列，实际上这些“多余”的订阅者早已失去了相关的消费逻辑。但是在新 CMQ 的订阅逻辑里，只要队列删除了，相关的主题订阅关系也一并会被删除，因此不会展示这部分订阅者。

解决方案：

如果您希望验证这个问题，可以将原 CMQ 里“没有迁移过来”的订阅队列名输入到原 CMQ 队列列表的搜索框中搜索一下，没有结果则证明该订阅关系在新 CMQ 的缺失属于上述情况。

消费时延明显增长

问题场景：

切换到新版 CMQ 后发现消息拉取的时间明显变长（可能达到10s – 20s）。此类问题属于已知问题，常出现在消息量不大，且消费者不多的场景。原因在于新版的分布式架构中，拉取消息轮询各节点的随机性在消息数量少的时候体现得更为明显，使得底层会有一些反复轮询的现象，导致时延增加。

解决方案：

1. 队列属性中的“消息接收长轮询等待时间”改小，推荐调整为3秒。
2. 我们准备了专门针对这种场景特化的集群，降低轮询耗时，**用户将客户端的接入点切换为这些集群即可明显降低消息拉取的时耗**（无需重建队列）。

目前已支持特化集群的有以下地域：

地域	接入点
广州	• http://gz.mqadapter.cmq.tencentyun.com:8080(内网) • https://cmq-gz.public.tencenttdmq.com:9443
上海	• http://sh.mqadapter.cmq.tencentyun.com:8080(内网) • https://cmq-sh.public.tencenttdmq.com:9443(公网)

地域	接入点
北京	• http://bj.mqadapter.cmq.tencentyun.com:8080(内网) • https://cmq-bj.public.tencenttdmq.com:8443(公网)
中国香港	• http://hk.mqadapter.cmq.tencentyun.com:8080(内网) • https://cmq-hk.public.tencenttdmq.com:8443(公网)

管控类 API(新增/查看队列等)不兼容

问题场景:

切换过来后 SDK 中 createQueue 等管控类的方法报错。

问题说明:

新版 CMQ 对于原来 SDK 中管控流的操作是不兼容的（创建队列、查看队列列表等接口）。腾讯云对控制台整体的云 API 做了一次升级（V2 到 V3），不再提供原来 V2 协议的接口，因此新版 CMQ 管控流相关的接口需要遵照最新云 API 的协议进行改造，改造后性能和开发友好性更高。具体可以参见 API 文档。

新版 CMQ 管控流和数据流做了分离，原则上是这两类的调用场景和性质不同，区分可以更好提供服务，因此调用的域名也不相同:
数据流的调用地址请在控制台获取；管控流的调用地址请参考具体的云 API 的文档，例如 创建 CMQ 队列 API 文档。

原先使用的 TCP 协议的 SDK，新版 CMQ 出错

新版 CMQ 默认仅支持 HTTP 协议，如果原先有使用 TCP 协议的 SDK 且希望无缝迁移，我们提供了 TCP SDK 的专属接入点，如下所示:

地域	接入点
广州	• http://gz.mqadapter.cmq.tencentyun.com:12000 (内网) • https://cmq-gz.public.tencenttdmq.com:12000 (公网)
上海	• http://sh.mqadapter.cmq.tencentyun.com:12000 (内网) • https://cmq-sh.public.tencenttdmq.com:12000 (公网)
北京	• http://bj.mqadapter.cmq.tencentyun.com:12000 (内网) • https://cmq.bj.public.tencenttdmq.com:12000 (公网)

TCP 的客户端配置成 TCP 专用的接入点即可解决。

存量系统复杂迁移困难

如果您发现您正在使用 CMQ 的系统较为复杂或由于各类原因难以维护，导致迁移困难，请及时通过 工单 与我们联系，我们将提供后台切换的方案协助您进行迁移。

> ⚠ 注意
>
> - 后台切换会有一定的延时增长
> - 后台切换完成后，新增或者编辑的操作请勿在原版 CMQ 进行

参数差异说明

最近更新时间：2022-06-09 15:32:50

新版 CMQ 与原CMQ参数差异说明

新版 CMQ 在数据流（消息收发）SDK 的用法和语法上与原 CMQ 一致，但有些参数与特性会和原 CMQ 有一定的差异。这些差异新版 CMQ 会通过特殊设置这些参数来保证在您迁移之后不会改变原有的生产消费逻辑，但如果是新建的队列或主题则尽可能参考新 CMQ 的逻辑进行设置。

消息生命周期

新版 CMQ 采用了业界通用的消息生命周期模型，即通过 TTL(Time to Live)和 Retention 配合来避免产生过量的堆积导致消息队列负载过高（消息堆积容量达到100%以上时会触发不可写入）。

相较于原版 CMQ，新版 CMQ 增设了消息最大未确认时间，范围30秒到12小时，如果消费客户端在获取到消息后超过此时间仍未进行消息的确认，则服务端会自动确认该消息。

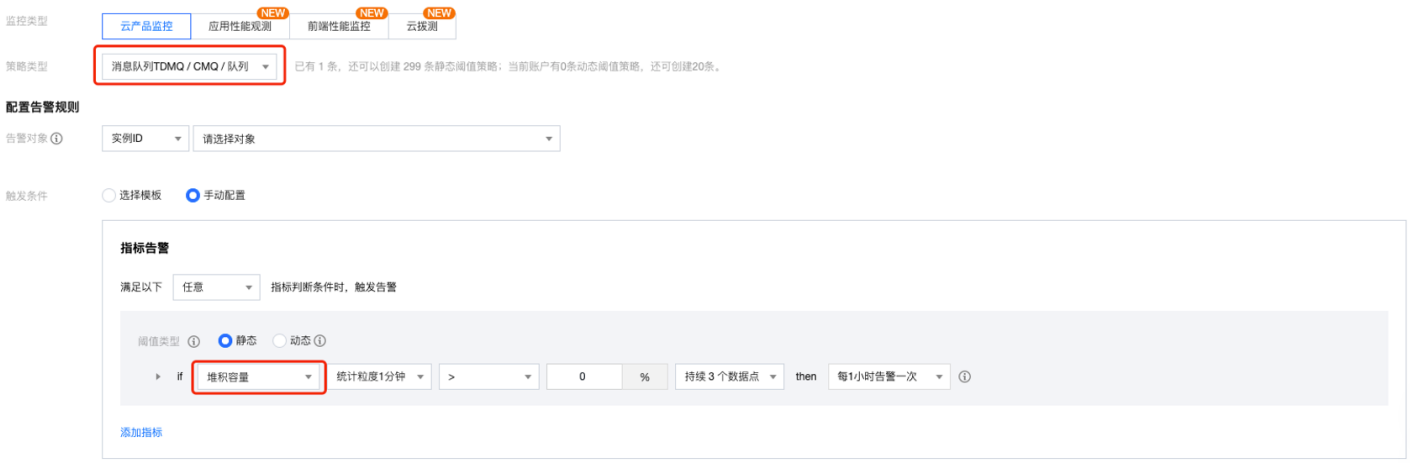
未确认的消息将持续保存在 MQ 中不会删除，已确认的消息受到消息可回溯时间大小和可回溯磁盘空间的作用（如果没消息回溯开则会在确认后直接删除）

新版 CMQ 取消了消息生命周期的限制。默认不再支持消息长时间在队列中堆积，如有特殊需要可以开启消息回溯并设置可回溯的时间范围，只有开启了消息回溯的消息才允许在消息队列中保存超过12小时，开启后会产生一定的存储费用，具体计费请参考新版 CMQ 计费说明。

> 说明：
> 从原 CMQ 迁移到新 CMQ 的队列，如果设置了生命周期超过12小时的，会将消息最大未确认时间调整到12小时，其余情况，消息最大未确认时间会继承原 CMQ 消息生命周期的值。

消息堆积上限

新版 CMQ 取消了原版 CMQ 关于消息堆积条数的限制，理论上只要存储资源满足，可以无限堆积。但实际从硬件层面出发，我们会给每个队列分配10GB的最大堆积存储资源，并支持您通过该值配置对应的云监控告警。



一般平均1KB大小的消息可以堆积大约1千万条，可以依据此进行简单换算，如发现迁移后此处可能存在堆积超过上限的风险，可以及时通过 工单 与我们联系。

消息大小

新版 CMQ 不再支持设置消息大小上限，为不影响业务从原 CMQ 迁移，新增队列统一设定为原 CMQ 的消息大小上限，即1024KB。

> 说明：
> 从原 CMQ 迁移过来的队列不再支持设置消息大小，如需增加限制，请重新创建队列使用。

消息接收长轮询等待时间

参数意义完全相同，但是其作用效果在新版 CMQ 和原 CMQ 上有所不同，新版中该参数推荐设置到3s以下。

在新版 CMQ 中，如果消息接收长轮询等待时间设置的过大，由于底层需要保证“至少一次”的语义，可能导致消息投递的重复率显著增高，从而对于一些未做消息去重的下游业务系统产生较大影响。因此，如果希望减少消息重复的概率，可以尽量设置的小一些，推荐3秒，3秒以下基本不会产生重复投递。

未确认消息容量

新版 CMQ 增设了未确认消息容量的限制，这样可以保障 MQ 服务端的内存消耗得到控制从而确保稳定性。不可见消息过多一般是客户端未及时 ACK 导致的，该指标有对应的监控图表进行监控，如有明显突增，请尽快检查消费者的确认删除逻辑是否正常运行，如突发性容量不足请尽快 提交工单 申请。