

# 消息队列 CMQ 版 实践教学



腾讯云

## 【 版权声明 】

©2013–2026 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

## 【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

## 【 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或 95716。

# 文档目录

## 实践教程

消息去重

选择 Push 还是 Pull

旧版 CMQ 接入地址切换指南

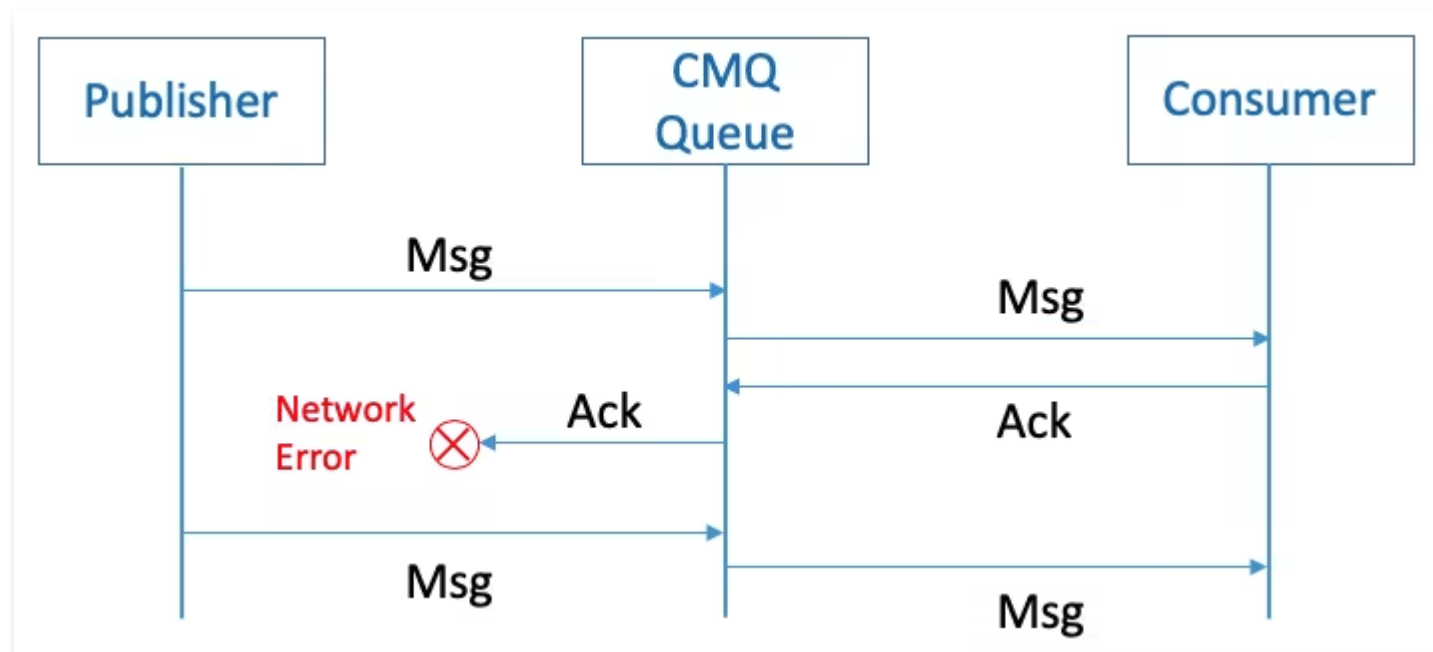
# 实践教程

## 消息去重

最近更新时间：2025-12-29 11:09:21

对于重复消息，最好的方法是消息可重入（消息重复消费对业务无影响）。做不到可重入时，需要在消费端去重。

### 重复消息出现的原因



网络异常、服务器宕机等原因都有可能导致消息丢失。TDMQ CMQ 版为了做到不丢消息、可靠交付，采用了消息生产、消费确认机制。

**生产消息确认：**生产者向 TDMQ CMQ 版发送消息后，等待 TDMQ CMQ 版回复确认；TDMQ CMQ 版将消息持久化到磁盘后，向生产者返回确认成功。否则在生产者请求超时、TDMQ CMQ 版返回失败等情况下，生产者需要向 TDMQ CMQ 版重发消息。

**消费者确认：**TDMQ CMQ 版向消费者交付消息后，将消息置为不可见；在消息不可见时间内，消费者使用句柄删除消息。如果消息未被删除，且不可见时间超时，消息将重新可见。

由于消息确认机制是“至少一次交付（at least once）”，在网络抖动、生产者/消费者异常等情况下，就会出现生产者重复生产、消费者重复消费的情况。

### 去重方案

要去重，先要识别重复消息。通常的做法是在生产消息时，业务方在消息体中插入去重 key，消费时通过该去重 key 来识别重复消息。去重 key 可以由 <生产者 IP + 线程 ID + 时间戳 + 时间内递增值> 组成的唯一值。只有一个消费者时，您可以将消费过的去重 key 缓存（如 KV 等），然后每次消费时检查去重 key 是否已消费过。去重 key 缓存可以根据消息最大有效时间来淘汰。TDMQ CMQ 版提供了队列当前最小未消费消息的时间

(`min_msg_time`)，您可以使用该时间和业务生产消息最大重试时间来确定缓存淘汰时间。  
存在多个消费者时，去重 key 缓存就需要是分布式的。

- 根据消息最大有效时间，计算 key 过期时间点：

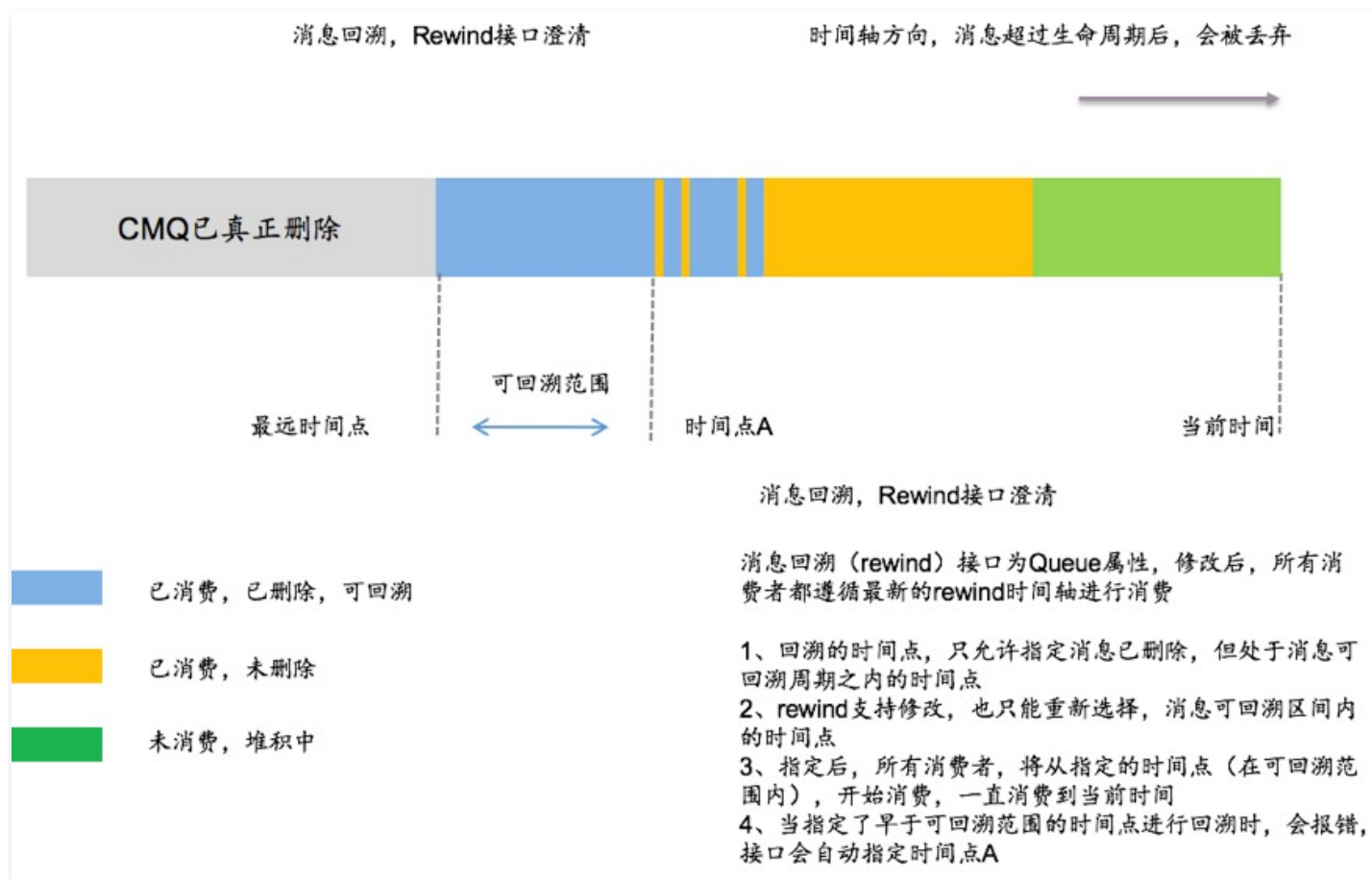
$\text{current\_time} + \text{max\_retention\_time} + \text{max\_retry\_time} + \text{max\_network\_time}$   
(当前时间) + (最大有效时间) + (最大重试时间) + (最大网络时间)

- 根据 TDMQ CMQ 版最小未消费时间，计算 key 过期时间点：

$\text{min\_msg\_time} + \text{max\_retry\_time} + \text{max\_network\_time}$   
(最小未消费时间) + (最大重试时间) + (最大网络时间)

TDMQ CMQ 版可配置消息最大有效时间为15天，业务可根据实际情况调整。

TDMQ CMQ 版队列当前最小未消费消息时间，即下图中最远时间点。该时间之前的消息都已经被删除，之后的消息可能未被删除。



## 举例说明

避免重复提交：

- 场景：A 为生产者，B 为消费者，中间是 TDMQ CMQ 版。A 已完成10元转账操作，且将消息发送给 TDMQ CMQ 版，TDMQ CMQ 版也已成功收到。此时网络闪断或者客户端 A 宕机导致服务端应答给客户端 A 失败。A 会认为发送失败，从而再次生产消息。这会造成重复提交。

- 解决方法：A 在生产消息时，加入 time 时间戳等信息，生成唯一的去重 key。若生产者 A 由于网络问题判断当前发送失败，重试时，去重 key 沿用第一次发送的去重 key。此时消费者 B 可通过去重 key 判断并做去重。

该案例也说明了不能使用 TDMQ CMQ 版的 Message ID 进行去重，因为这两条消息有不同的 ID，但却有相同的 body。

- 注意事项：生产者 A，在发送消息之前，要将去重 key 做持久化（写磁盘等，避免掉电后丢失）。

#### **避免多条相同 body 的消息被过滤：**

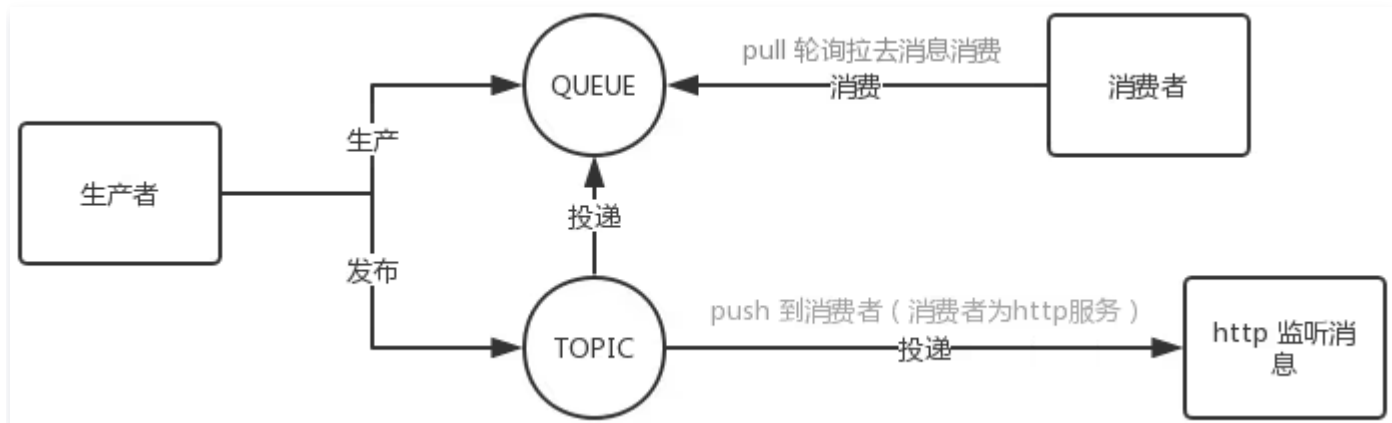
- 场景：A 给 B 转账10元，一共发起5次，每一次提交的 body 内容是一样的。如果消费者粗暴用 body 做去重判断，就会把5次请求，当做1次请求来处理。
- 解决方法：A 在生产消息时，加入 time 时间戳等信息。此时哪怕消息 body 一样，生成的去重 key 都是不同的，这样就满足了多次发送同样内容的需求。

# 选择 Push 还是 Pull

最近更新时间：2025-12-29 11:09:22

TDMQ CMQ 版支持 Pull（队列）和 Push（主题）两种方式：

- Push 模型：当 Producer 发出的消息到达后，服务端马上将这条消息投递给 Consumer。
- Pull 模型：当服务端收到这条消息后什么也不做，只是等着 Consumer 主动到自己这里来读，即 Consumer 这里有一个“拉取”的动作。



本文将结合不同场景，简要分析 Push 模型和 Pull 模型各自存在的利弊。

## 场景1：Producer 速率大于 Consumer 速率

当 Producer 速率大于 Consumer 速率时，有两种可能性：一种是 Producer 本身的效率就要比 Consumer 高（例如，Consumer 端处理消息的业务逻辑可能很复杂，或者涉及到磁盘、网络等 I/O 操作）；另一种是 Consumer 出现故障，导致短时间内无法消费或消费不畅。

Push 方式由于无法得知当前 Consumer 的状态，所以只要有数据产生，便会不断地进行推送，在以上两种情况下时，可能会导致 Consumer 的负载进一步加重，甚至是崩溃（例如生产者是 flume 疯狂抓日志，消费者是 HDFS+hadoop，处理效率跟不上）。除非 Consumer 有合适的反馈机制能够让服务端知道自己的状况。而采取 Pull 的方式问题就简单了许多，由于 Consumer 是主动到服务端拉取数据，此时只需要降低自己访问频率即可。举例：如前端是 flume 等日志收集业务，不断向 TDMQ CMQ 版生产消息，TDMQ CMQ 版向后端投递，后端业务如数据分析等业务，效率可能低于生产者。

## 场景2：强调消息的实时性

采用 Push 的方式时，一旦消息到达，服务端即可马上将其推送给消费端，这种方式的实时性显然是非常好的；而采用 Pull 方式时，为了不给服务端造成压力（尤其是当数据量不足时，不停的轮询显得毫无意义），需要控制好自己的轮询的间隔时间，但这必然会给实时性带来一定的影响。

## 场景3：Pull 的长轮询

Pull 模式存在的问题：由于主动权在消费方，消费方无法准确地决定何时去拉取最新的消息。如果一次 Pull 取到消息了还可以继续去 Pull，如果没有 Pull 取到消息则需要等待一段时间再重新 Pull。

由于等待时间很难判定。您可能有很多种动态拉取时间调整算法，可能依然会遇到问题，是否有消息到来不是由消费方决定。也许1分钟内连续到来3000条消息，接下来的几分钟或几个小时内却没有新消息产生。

TDMQ CMQ 版提供了长轮询的优化方法，用以平衡 Pull/Push 模型各自的缺点。基本方式是：消费者如果尝试拉取失败，不是直接 return，而是把连接挂在那里 wait，服务端如果有新的消息到来，把连接拉起，返回最新消息。

## 场景4：部分或全部 Consumer 不在线

在消息系统中，Producer 和 Consumer 是完全解耦的，Producer 发送消息时，并不要求 Consumer 一定在线，对于 Consumer 也是同样的道理，这也是消息通信区别于 RPC 通信的主要特点；但是对于 Consumer 不在线的情况，却有很多值得讨论的场景。

首先，在 Consumer 偶然宕机或下线时，Producer 的生产是可以不受影响的，Consumer 上线后，可以继续之前的消费，此时消息数据不会丢失；但是如果 Consumer 长期宕机或是由于机器故障无法再次启动，就会出现問題，即服务端是否需要为 Consumer 保留数据，以及保留多久的数据等。

采用 Push 方式时，因为无法预知 Consumer 的宕机或下线是短暂的还是持久的，如果一直为该 Consumer 保留自宕机开始的所有历史消息，那么即便其他所有的 Consumer 都已经消费完成，数据也无法清理掉，随着时间的积累，队列的长度会越来越大，此时无论消息是暂存于内存还是持久化到磁盘上（采用 Push 模型的系统，一般都是将消息队列维护于内存中，以保证推送的性能和实时性，这一点会在后边详细讨论），都将对 TDMQ CMQ 版服务端造成巨大压力，甚至可能影响到其他 Consumer 的正常消费，尤其当消息的生产速率非常快时更是如此；但是如果不保留数据，那么等该 Consumer 再次起来时，则要面对丢失数据的问题。

折中的方案是：TDMQ CMQ 版给数据设定一个超时时间，当 Consumer 宕机时间超过这个阈值时，则清理数据；但这个时间阈值也并不太容易确定。

在采用 Pull 模型时，情况会有所改善；服务端不再关心 Consumer 的状态，而是采取“你来了我才服务”的方式，Consumer 是否能够及时消费数据，服务端不会做任何保证（也有超时清理时间）。



# 旧版 CMQ 接入地址切换指南

最近更新时间：2025-08-26 11:30:11

## 1. 产品背景

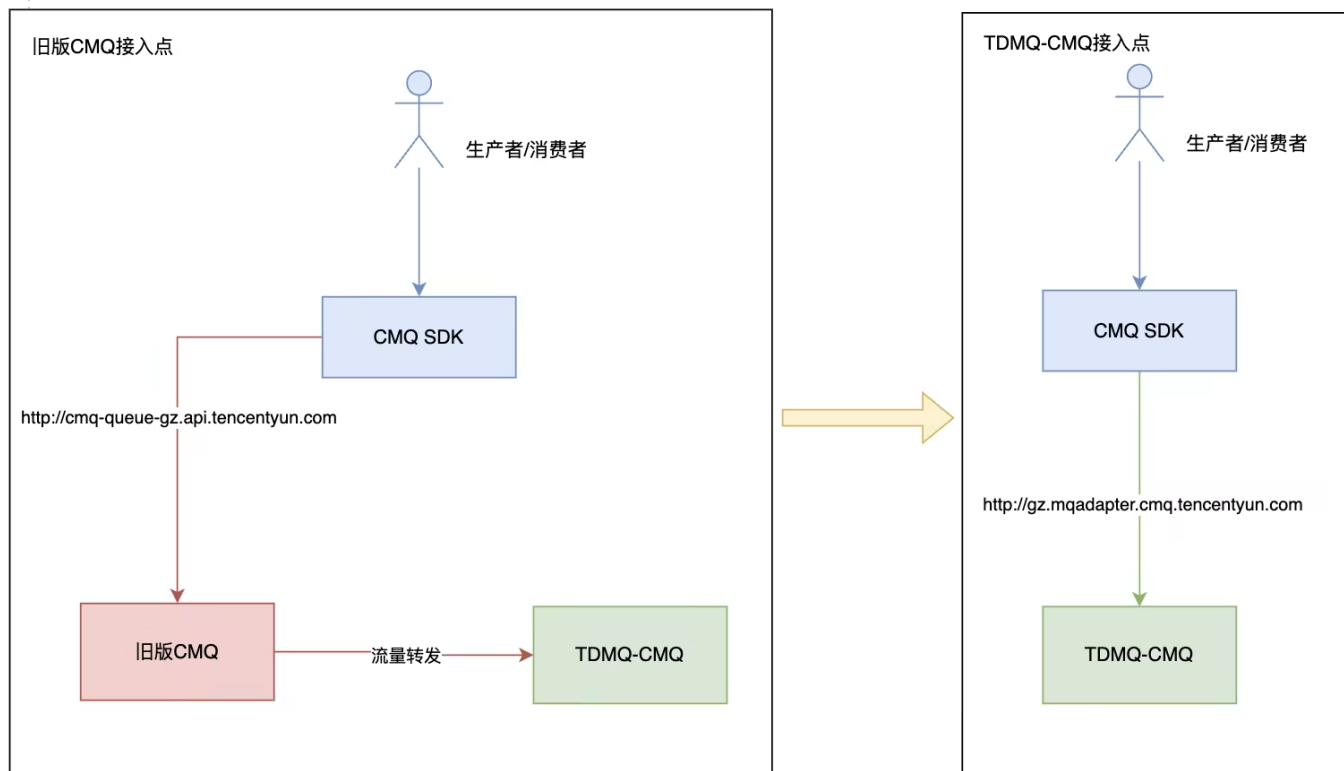
腾讯云消息队列旧版 CMQ 已完成全面升级，旧版 CMQ 服务流量已全部迁移至 TDMQ-CMQ。TDMQ-CMQ 是腾讯云基于 RocketMQ 架构打造的新一代分布式消息队列服务，在兼容旧版 CMQ 功能的同时，提供了更强大的消息处理能力和更稳定的服务保障。

为了进一步给您提供稳定的服务和SLA保障，请您将旧版 CMQ 的接入地址切换为新版 TDMQ-CMQ 的接入地址。

## 2. 接入点切换背景

- **服务终止**：旧版 CMQ 服务已全面停止运营，所有流量已切换至 TDMQ-CMQ，旧版 CMQ 机器需全部回收处理。
- **兼容性保障**：TDMQ-CMQ 完全兼容原有 CMQ SDK 接口，业务逻辑代码无需修改，只需要修改接入点配置。
- **性能提升**：接入点替换后，可以提供更低的服务请求耗时和更好的服务 SLA 保障。

## 3. 接入点切换优势



接入点架构图

如上图所示，旧版 CMQ 服务流量通过无感升级的方式已迁移到了 TDMQ-CMQ，仍然使用旧版 CMQ 接入点进行生产消费，请求都会由旧版 CMQ 网关层转发到 TDMQ-CMQ，如果直接使用 TDMQ-CMQ 接入点，请求则会直接路由到 TDMQ-CMQ。

使用 TDMQ-CMQ 接入点，少一层网关转发，网络链路更短，请求延迟更低。同时服务链路不受旧版 CMQ 网关影响，稳定性更高，服务可用性更高。

## 4. 切换步骤

### 4.1 获取新接入点

登录腾讯云控制台，在 [TDMQ CMQ 版控制台](#) 的队列服务 > API 请求地址处复制获取新的接入点。

接入点切换对应列表：

地域	旧版 CMQ 接入点	TDMQ-CMQ 接入点
广州	<p><b>公网地址：</b></p> <p><code>https://cmq-queue-gz.api.qcloud.com</code></p> <p><code>https://cmq-topic-gz.api.qcloud.com</code></p> <p><b>内网地址：</b></p> <p><code>http://cmq-queue-gz.api.tencentyun.com</code></p> <p><code>http://cmq-topic-gz.api.tencentyun.com</code></p>	<p><b>公网地址：</b> <code>https://cmq-gz.public.tencenttdmq.com</code></p> <p><b>内网地址：</b> <code>http://gz.mqadapter.cmq.tencentyun.com</code></p>
上海	<p><b>公网地址：</b></p> <p><code>https://cmq-queue-sh.api.qcloud.com</code></p> <p><code>https://cmq-topic-sh.api.qcloud.com</code></p> <p><b>内网地址：</b></p> <p><code>http://cmq-queue-sh.api.tencentyun.com</code></p> <p><code>http://cmq-topic-sh.api.tencentyun.com</code></p>	<p><b>公网地址：</b> <code>https://cmq-sh.public.tencenttdmq.com</code></p> <p><b>内网地址：</b> <code>http://sh.mqadapter.cmq.tencentyun.com</code></p>
北京	<p><b>公网地址：</b></p> <p><code>https://cmq-queue-bj.api.qcloud.com</code></p> <p><code>https://cmq-topic-bj.api.qcloud.com</code></p> <p><b>内网地址：</b></p>	<p><b>公网地址：</b> <code>https://cmq-bj.public.tencenttdmq.com</code></p> <p><b>内网地址：</b> <code>http://bj.mqadapter.cmq.tencentyun.com</code></p>

	<p><code>http://cmq-queue-bj.api.tencentyun.com</code></p> <p><code>http://cmq-topic-bj.api.tencentyun.com</code></p>	
成都	<p><b>公网地址:</b></p> <p><code>https://cmq-queue-cd.api.qcloud.com</code></p> <p><code>https://cmq-topic-cd.api.qcloud.com</code></p> <p><b>内网地址:</b></p> <p><code>http://cmq-queue-cd.api.tencentyun.com</code></p> <p><code>http://cmq-topic-cd.api.tencentyun.com</code></p>	<p><b>公网地址:</b> <code>https://cmq-cd.public.tencentttdmq.com</code></p> <p><b>内网地址:</b> <code>http://cd.mqadapter.cmq.tencentyun.com</code></p>
上海金融	<p><b>公网地址:</b></p> <p><code>https://cmq-queue-shjr.api.qcloud.com</code></p> <p><code>https://cmq-topic-shjr.api.qcloud.com</code></p> <p><b>内网地址:</b></p> <p><code>http://cmq-queue-shjr.api.tencentyun.com</code></p> <p><code>http://cmq-topic-shjrapi.tencentyun.com</code></p>	<p><b>公网地址:</b> <code>https://cmq-shjr.public.tencentttdmq.com</code></p> <p><b>内网地址:</b> <code>http://shjr.mqadapter.cmq.tencentyun.com</code></p>
深圳金融	<p><b>公网地址:</b></p> <p><code>https://cmq-queue-szjr.api.qcloud.com</code></p> <p><code>https://cmq-topic-szjr.api.qcloud.com</code></p> <p><b>内网地址:</b></p> <p><code>http://cmq-queue-szjr.api.tencentyun.com</code></p> <p><code>http://cmq-topic-szjrapi.tencentyun.com</code></p>	<p><b>公网地址:</b> <code>https://cmq-szjr.public.tencentttdmq.com</code></p> <p><b>内网地址:</b> <code>http://szjr.mqadapter.cmq.tencentyun.com</code></p>

## 4.2 配置变更

修改应用程序配置文件中 endpoint 地址。

```
Account account = new Account (SERVER_ENDPOINT, SECRET_ID, SECRET_KEY);
```

#### # 旧配置示例

```
SERVER_ENDPOINT= "http://cmq-queue-gz.api.qcloud.com"
```

#### # 新配置示例

```
SERVER_ENDPOINT= "http://gz.mqadapter.cmq.tencentyun.com"
```

## 4.3 验证测试

4.3.1 发送测试消息；

4.3.2 消费测试消息，并删除消息；

4.3.3 验证消息是否生产消费删除成功。

## 4.4 正式切换

可采用灰度发布策略，先切换部分资源接入点观察业务指标，灰度验证通过后全量切换。

## 5. 常见问题

**Q1: 切换过程中原有消息是否会丢失？**

A: 不会丢失。用户切换接入点不影响消息的生产消费，且新接入点完全兼容旧版 CMQ 协议接口。

**Q2: 新接入点的鉴权方式是否有变化？**

A: 鉴权机制保持不变，继续使用 SecretId 和 SecretKey 进行鉴权。

如需进一步帮助，请[提交工单](#)联系我们，我们将为您提供专业的技术支持服务，解答您在切换过程中遇到的任何问题。