天翼云

分布式 RocketMQ

用户指南

目 录

1 简介

1.1 RocketMQ 介绍

分布式消息服务 RocketMQ，是在开源消息中间件 RocketMQ 基础上，进行问题修复与优化，实现低成本、高可靠、高性能和具备监控运维能力的消息中间件产品。提供高效可靠的消息传递服务，解决分布式应用系统之间的消息数据通信难题，用于系统间的解耦。主要应用于：

1）分布式事务：基于消息有序、不重、不丢失的特性，通过发送事务型消息，实现数据的最终一致性。

2）削峰填谷：对于突发流量，数据可先堆积在 MQ，消费者按照实际能力来处理，防止雪崩效应和消息丢失。

3）系统解耦：通过 MQ 解耦系统间的依赖关系，将调用进行异步化处理，提升整体性能。

4）消息复用：数据生产一次并堆积到 MQ，不同消费场景可多次复用。

1.2 基本术语

Broker

消息中转角色，负责存储消息，转发消息，一般也称 Server。在 JMS 规范中称为 Provider。RocketMQ 一般在多个服务器部署 broker 集群，从而达到分布式、高可用、可横向扩展的目的。

Name Server

Name Server 是一个几乎无状态节点，可集群部署，节点之间无同步信息。它主要提供 broker 注册、Topic 路由管理等功能

Topic

在 RocketMQ 中，topic 类似于 JMS 规范中的队列，所有消息都是存放在不同的 topic 中，生产都与消费者都以 topic 名字进行生产与消费。(注：RocketMQ 的 topic 并不是 jms 规范中广播消费 topic 的概念)

一个 Topic 可以存在多个 broker 之中，这样 topic 就可以分布在不同的 broker 从而达到分布式的目的。

一个 Topic 下面，可以有多个队列，可以理解成分区，Topic 的消息是放在不同队列下的。

Queue

在 RocketMQ 中，queue 是存放数据的最小单位，queue 是存在于 topic 下面的。在 RocketMQ 中，queue 不同于 JMS 规范中的队列，可以理解为 topic 的分区。

生产组

一类 Producer 的集合名称，这类 Producer 通常发送一类消息，且发送逻辑一致，一般由业务系统负责产生消息。

消费组

一类 Consumer 的集合名称，这类 Consumer 通常消费一类消息，且消费逻辑一致，一般是后台系统负责异步消费。每个消费组对应一个消费进度，集群消费的消费进度存储在 broker 上。

消费者实例

一个消费者实例代表消费组的一员，不同的消费者用不同的实例名字创建。

生产者实例

一个生产者实例代表生产者的一员，不同的生产者用不同的实例名字创建。

PUSH 消费

Consumer 的一种，应用通常向 Consumer 对象注册一个 Listener 接口，一旦收到消息，Consumer 对象立刻回调 Listener 接口方法。在 RocketMQ 中，客户端会自动起线程消费消息，线程数是 5-64，意味着，listener 里面的方法，会被多线程执行。客户端内部可以根据堆积量进行调整，使用者不需要新启、管理消费线程。并有流控机制，当客户端缓存一定量消费不及时，会停止推送新消息。

PULL 消费

Consumer 的一种，应用通常主动调用 Consumer 的拉消息方法从 Broker 拉消息，主动权由应用控制，但实时性取决于应用主动拉取的频率。在 PULL 消费中，线程由应用自主决定。

广播消费

注意：使用消费模式，在很多使用场景都会带来影响或限制，在 RocketMQ 中，应尽量避免使用此消费模式。

在 RocketMQ 中，消费者有两种不同的方式消费 topic 中的消息，其中一种是广播消费。在广播消费模式下，一条消息被多个 Consumer 消费，即使这些 Consumer 属于同一个 Consumer Group，消息也会被 Consumer Group 中的每个 Consumer 都消费一次，广播消费中的 Consumer Group 概念可以认为在消息划分方面无意义。

V1.x 版本由于广播消费的消费进度，是保存在客户端的，对于很多使用场景会带来影响，在 RocketMQ 中，并不推荐使用此消费模式。

集群消费

一个 Topic 可以被一个或多个 Consumer Group 消费，每个 Consumer Group 有自己独立的消费进度，消费进度是保存在服务端的。

一个 Consumer Group 中的消费者实例可以平均分摊消费消息，做到负载均衡。例如某个 Topic 有 9 条消息，其中一个 Consumer Group 有 3 个不同的消费者实例（可能是 3 个进程，或者 3 台机器），那么每个实例只消费其中的 3 条消息。

在此消费模式下，可以做到 Point-To-Point 的消费，也可以做到 JMS 里面广播消费，能满足绝大部分场景，推荐使用此消费模式。

有序消息

消费消息的顺序要同发送消息的顺序一致，在 RocketMQ 中，主要有两种有序消息。

普通有序消息

有序消息的一种，在正常情况下可以保证完全的顺序消息，但是一旦发生通信异常，Broker 重启，由于队列总数发生变化，哈希取模后定位的队列会变化，产生短暂的消息顺序不一致。

如果业务能容忍在集群异常情况（如某个 Broker 宕机或者重启）下，消息短暂的乱序，使用普通顺序方式比较合适。

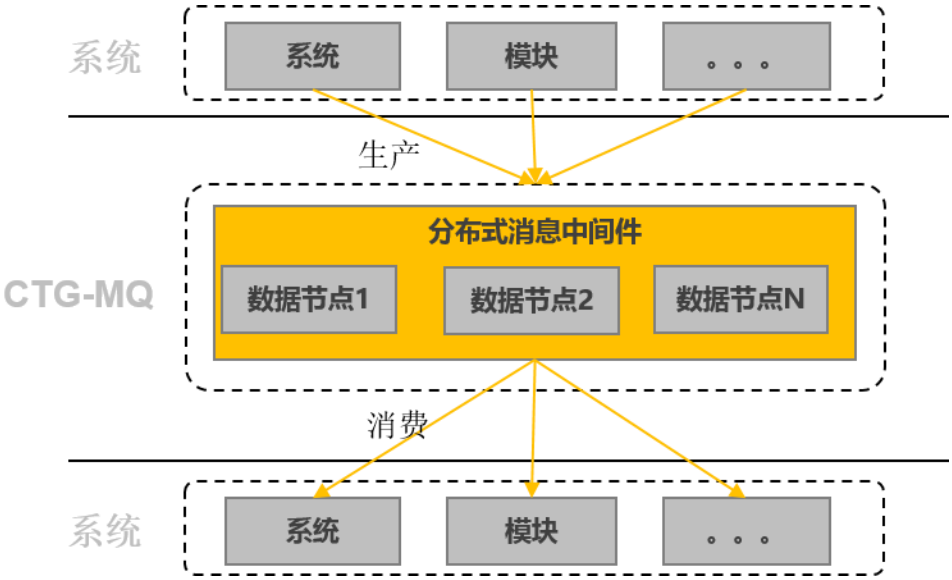
严格有序消息

有序消息的一种。无论正常异常情况都能保证顺序，但是牺牲了分布式 Failover 特性，即 Broker 集群中只要有一台机器不可用，则整个集群都不可用（或者影响 hash 值对应队列的使用），服务可用性大大降低。

如果服务器部署为同步双写模式，此缺陷可通过备机自动切换为主避免，不过仍然会存在几分钟的服务不可用。

若业务能容忍短暂乱序，推荐普通有序消费。

1.3 应用场景



典型场景特征：

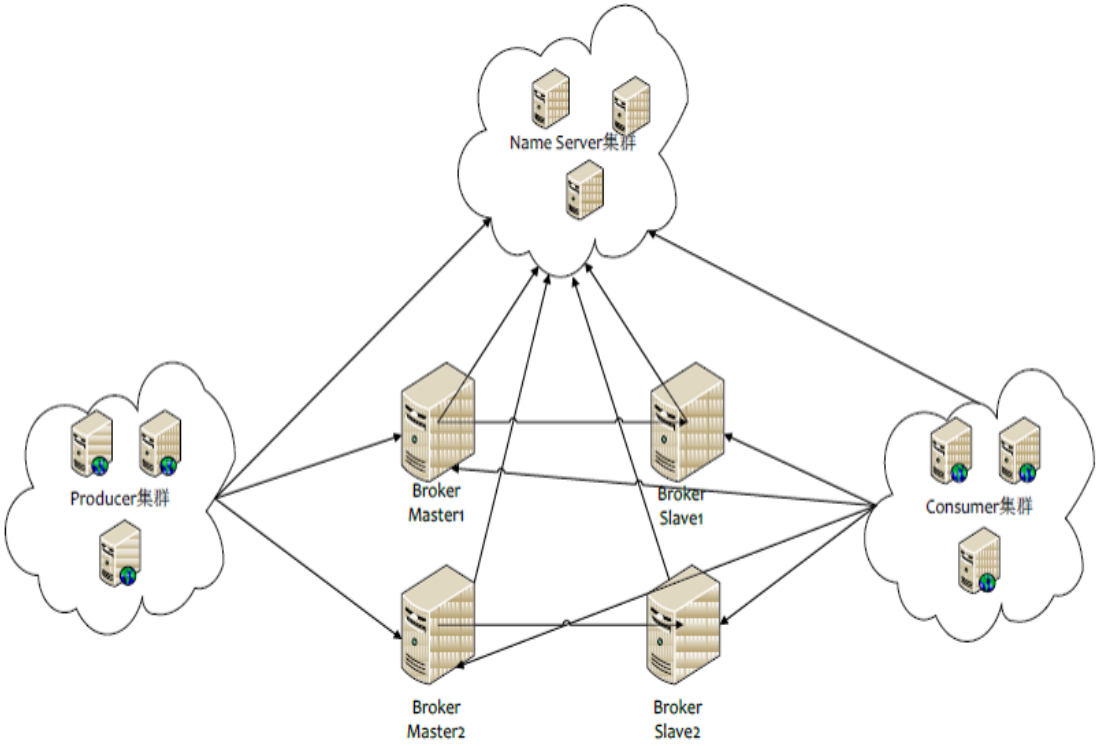
异步解耦：系统间请求异步解耦，通过消息堆积与高性能特性，实现平谷削峰；

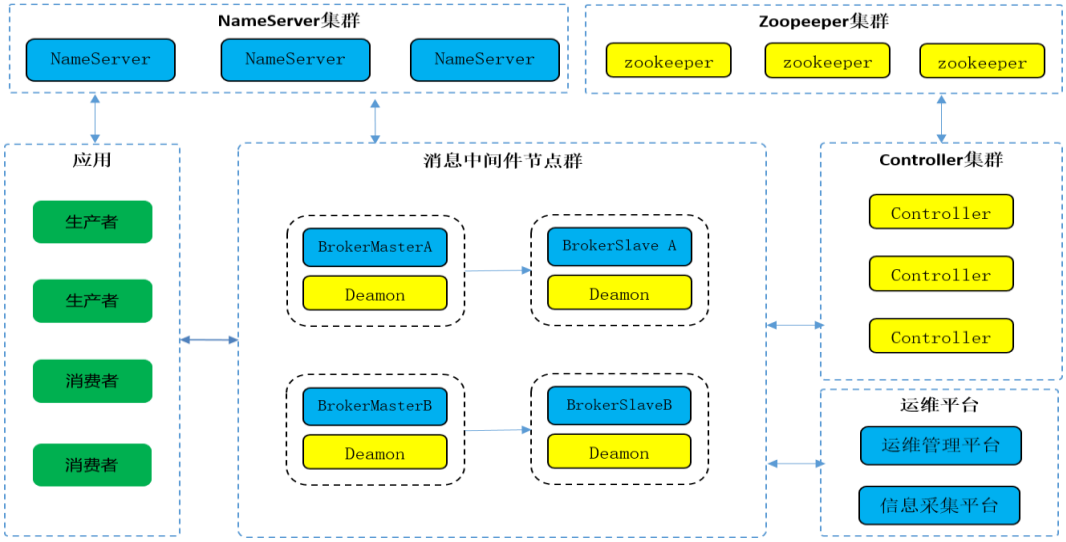
数据复制：通过消息将数据分发到多个系统进行处理；

事件通知：通过消息广播，高效地把分布式应用联系起来；

日志处理：高效地异步同步日志，进而做实时或离线分析。

1.4 整体架构





Name Server

Name Server 是一个几乎无状态节点，一般集群部署（2 个节点或以上），节点之间无同步信息。它主要提供 broker 注册、Topic 路由管理等功能。

Broker

分布式消息中间件核心组件，提供消息生产、消费，主从同步、数据刷盘等核心功能。可以横向扩展、在线扩容以提高集群性能。每个 Broker 与 Name Server 集群的所有节点建立长连接，并定时注册 Topic 等信息。

Producer

生产者，一般为应用调用 API 进行消息生产。Producer 与 Name Server 集群中的其中一个节点（随机选择）建立长连接，定期从 Name Server 取 Topic 路由信息，并向提供 Topic 服务的 Master 建立长连接，且定时向 Master 发送心跳。Producer 完全无状态，可集群部署。

Consumer

消费者，一般为应用调用 API 进行消息消费。Consumer 与 Producer 一样，与一个 Name Server 建立长连接并取 Topic 路由信息。Consumer 与提供 Topic 服务的 Master 建立长连接，进行消息消费。

1.5 应用使用建议

分布式消息服务 RocketMQ 对某些参数配置进行了约束和规范：

参数项	约束
主题，订阅组，应用用户名	不能包含特殊字符，只能是字母数字下划线横线
单主题的消息生产消费 TPS	5000
消息包体大小	在 RocketMQ 中，建议消息包体在 50K 以内；超过 50K 建议减小包体，过大的包体，会加大网络超时、网络拥堵、FLUSH_SLAVE_TIMEOUT 等异常情况出现的概率；
支持消息体的自动压缩，可通过设置 CompressMsgBodyOverHowmuch 参数，超过该值则会对消息体启用压缩。	默认 4k
生产者实例、消费者实例	同一个 jvm 进程初始化时启动一次，最后做成单例的。同一个生产者实例或消费者实例不要多次启动

2 产品功能

2.1 消息类型

2.1.1 无序消息

区别于其他类型的消息，无序消息在应用中最多，就是发送、消费无序消息，允许并行消费。

2.1.2 全局有序消息和局部有序消息

有序消息是指消费消息的顺序要同发送消息的顺序一致，在 RocketMQ 中，主要有两种有序消息，全局有序消息和局部有序消息（又叫普通有序消息、分区有序消息）。

普通有序消息

有序消息的一种，在正常情况下可以保证完全的顺序消息，但是一旦发生通信异常，Broker 重启，由于队列总数发生变化，哈希取模后定位的队列会变化，产生短暂的消息顺序不一致。

如果业务能容忍在集群异常情况（如某个 Broker 宕机或者重启）下，消息短暂的乱序，使用普通顺序方式比较合适。

严格有序消息

有序消息的一种。无论正常异常情况都能保证顺序，但是牺牲了分布式 Failover 特性，即 Broker 集群中只要有一台机器不可用，则整个集群都不可用（或者影响 hash 值对应队列的使用），服务可用性大大降低。

如果服务器部署为同步双写模式，此缺陷可通过备机自动切换为主避免，不过仍然会存在几分钟的服务不可用。

若业务能容忍短暂乱序，推荐普通有序消费。

2.1.3 事务消息

消息发送后，根据预设的事务进行判断，满足事务的消息将会被确认，不满足的事务的消息不会被服务端接收。

事务消息：

消息队列 RocketMQ 提供类似 X/Open XA 的分布事务功能，通过消息队列 事务消息能达到分布式事务的最终一致。

半消息：

暂不能投递的消息，发送方已经将消息成功发送到了消息队列 服务端，但是服务端未收到生产者对该消息的二次确认，此时该消息被标记成“暂不能投递”状态，处于该种状态下的消息即半消息。

消息回查：

由于网络闪断、生产者应用重启等原因，导致某条事务消息的二次确认丢失，消息队列 RocketMQ 服务端通过扫描发现某条消息长期处于“半消息”时，需要主动向消息生产者询问该消息的最终状态（Commit 或是 Rollback），该过程即消息回查。

2.1.4 延时消息

延时消息：Producer 将消息发送到消息队列 MQ 服务端，设计消费时延，消息发送到服务端后，过了预设的时间后才可以被消费者消费。发送延时消息时需要设定一个延时时间长度，消息将从当前发送时间点开始延迟固定时间之后才开始投递。

目前 RocketMQ 只支持特定级别的延时，16 个级别的延时。需要在发送消息时设置延时级别，如：

```
mq 默认的 16 个等级：messageDelayLevel：1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h
message.setDelayTimeLevel(5);
```

2.2 消费模型

2.2.1 集群消费和广播消息

广播消费：

注意：使用消费模式，在很多使用场景都会带来影响或限制，在 RocketMQ 中，应尽量避免使用此消费模式。

在 RocketMQ 中，消费者有两种不同的方式消费 topic 中的消息，其中一种是广播消费。在广播消费模式下，一条消息被多个 Consumer 消费，即使这些 Consumer 属于同一个 Consumer Group，消息也会被 Consumer Group 中的每个 Consumer 都消费一次，广播消费中的 Consumer Group 概念可以认为在消息划分方面无意义。

集群消费：

一个 Topic 可以被一个或多个 Consumer Group 消费，每个 Consumer Group 有自己独立的消费进度，消费进度是保存在服务端的。

一个 Consumer Group 中的消费者实例可以平均分摊消费消息，做到负载均衡。例如某个 Topic 有 9 条消息，其中一个 Consumer Group 有 3 个不同的消费者实例（可能是 3 个进程，或者 3 台机器），那么每个实例只消费其中的 3 条消息。

在此消费模式下，可以做到 Point-To-Point 的消费，也可以做到 JMS 里面广播消费，能满足绝大部分场景，推荐使用此消费模式。

2.2.2 Push 消费和 Pull 消费

从应用的角度，在 RocketMQ 中，支持 push 与 pull 消费方式。

Push 消费：

1）客户端通过注册监听 Listener 的方式，当有个消息可消费时，API 会调用 Listener 方法，主动推送消息；
2）为了能做到实时收消息，PUSH 方式使用长轮询机制，保证消息实时性。

Pull 消费：

1）客户端调用 pull 接口，主动拉取数据；

2）PULL 是应用控制线程的，应该可以多线程调用 pull 接口，也可以单线程拉取；

3）对于无序消费，应用可以多次调用 pull 并拉到数据，且与是否签收无关；

4）对于有序消费，只要同一 Queue 的消息被拉出去消费，但未签收，则此 Queue 无法再拉取消费。

2.2.3 有序消费和无序消费

无序消息、顺序消息，更多是针对消费者来说的。不同的队列数，配上不同的生产模式、消费者模式，可以适用于无序消费，有序消费的场景。

有序消费的缺点：

发送顺序消息无法利用集群 FailOver 特性；

消费顺序消息的并行度依赖于队列数量；

队列热点问题，个别队列由于哈希不均导致消息过多，消费速度跟不上，产生消息堆积问题；

遇到消息失败的消息，无法跳过，当前队列消费暂停。

2.3 消息重试

2.3.1 对于有序消息

有序消息不能跳跃签收，当消费者消费消息失败后，消息队列 RocketMQ 会自动不断进行消息重试（每次间隔时间为 1 秒），这时，应用会出现消息消费被阻塞的情况。因此，建议您使用有序消息时，务必保证应用能够及时监控并处理消费失败的情况，避免阻塞现象的发生。

2.3.2 对于无序消息

消费重试次数

消息队列 RocketMQ 默认允许每条消息最多重试 16 次，每次重试的间隔时间如下：

第几次重试	与上次重试的间隔时间
1	10 秒
2	30 秒
3	1 分钟
4	2 分钟
5	3 分钟

6	4 分钟
7	5 分钟
8	6 分钟
9	7 分钟
10	8 分钟
11	9 分钟
12	10 分钟
13	20 分钟
14	30 分钟
15	1 小时
16	2 小时

如果消息重试 16 次后仍然失败，消息将不再投递。如果严格按照上述重试时间间隔计算，某条消息在一直消费失败的前提下，将会在接下来的 4 小时 46 分钟之内进行 16 次重试，超过这个时间范围消息将不再重试投递。

注意： 消息重试，原消息会被签收，然后进入重试队列，重试消息的 Message ID 会改变。

2.4 消息过滤

1）消息 Tag，用于对某个 Topic 下的消息进行分类。 消息队列 RocketMQ 允许消费者按照 Tag 对消息进行过滤，确保消费者最终只消费到关心的消息类型。

2）针对消息归类，您可以选择创建多个 Topic， 或者在同一个 Topic 下创建多个 Tag。 但通常情况下，不同的 Topic 之间的消息没有必然的联系，而 Tag 则用来区分同一个 Topic 下相互关联的消息。

3）消费者如需订阅某 Topic 下所有类型的消息，Tag 用符号 * 表示;消费者如需订阅某 Topic 下多种类型的消息，请在多个 Tag 之间用 || 分隔。

2.5 自动主备切换

1）主 broker 故障，提供备 broker 节点自动拉起，主宕机自动主备切换功能，保证消息集群的高可用。由于主 broker 机器故障或者其他原因主 broker 挂掉不可用时，从 broker 会自动变为异步主。

生产者消费者在新主 broker 正常后，无报错，继续生产消费消息。

2）从 broker 故障，如果原主是同步 SYNC_MASTER，则原主变为 ASYNC_MASTER，不影响生产消费。

3 快速入门

3.1 创建队列

步骤 1：登录天翼云官网，开通 RocketMQ 服务；

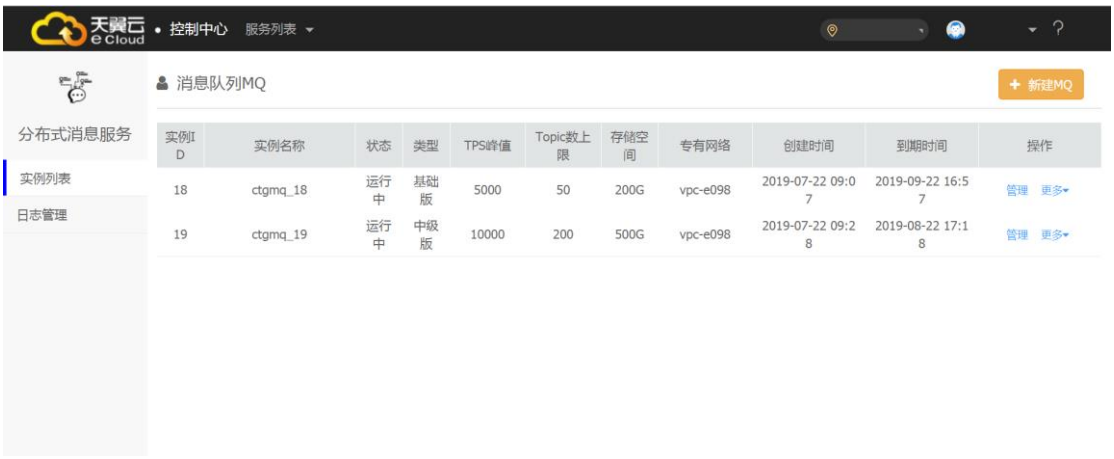
注意：

1） 如果用户需要创建 ipv4 实例，则 vpc 子网只需配置 ipv4 子网：

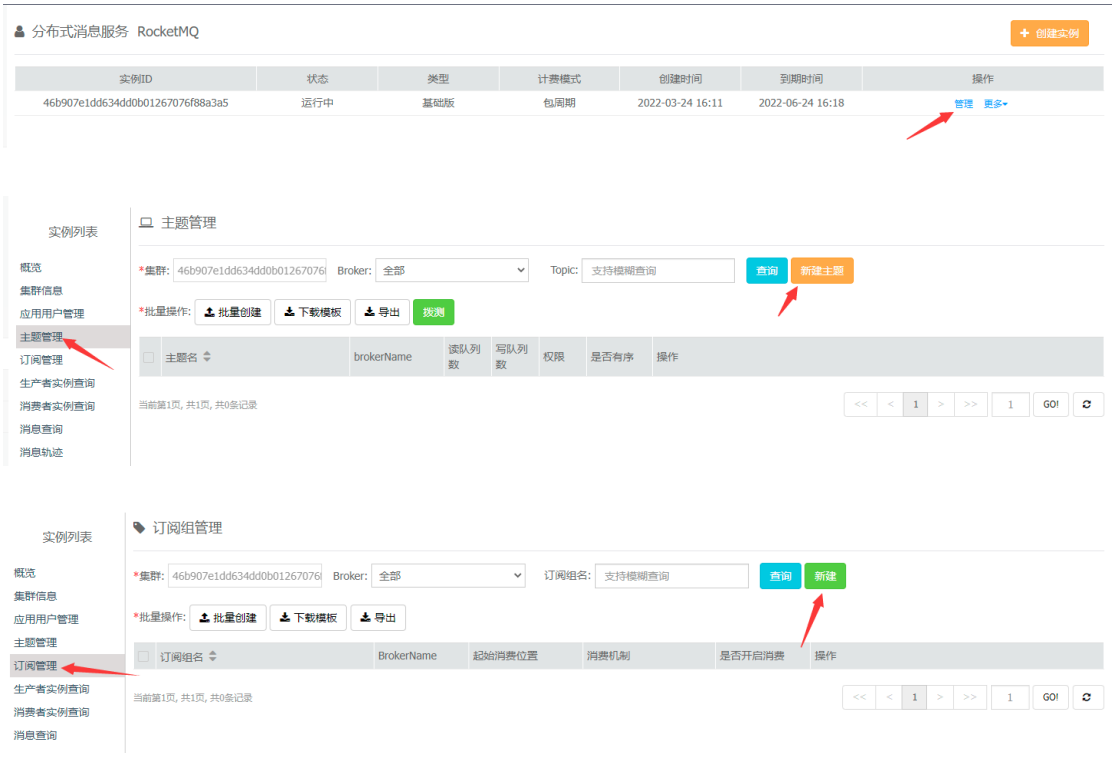
2） 如果用户需要创建 ipv6 实例，则 vpc 子网只需配置 ipv4/ipv6 双栈子网：

步骤 2：进入消息管理控制台；

步骤 3：创建实例，注意选择规格；

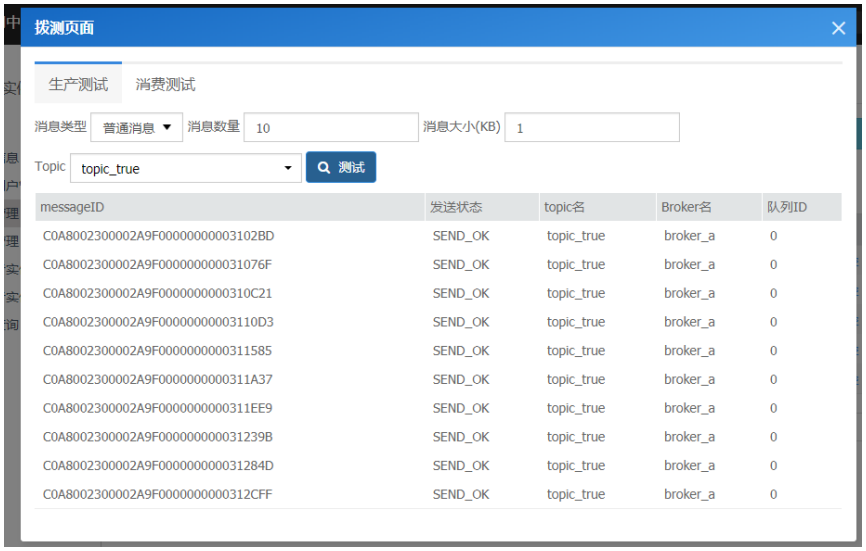


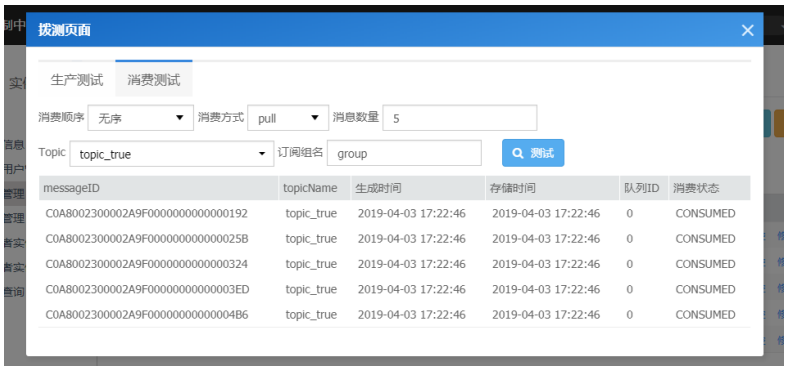
步骤 4：在消息管理台上创建主题和订阅组；



步骤 5：在消息管理台上创建应用用户和密码；

步骤 6：拨测验证是否可以发送、消费消息；





步骤 7：用户应用安装 sdk 示例接入消息队列，进行发送、消费消息。

参考：《天翼云-分布式消息服务-RocketMQ-API 说明文档（Java）.pdf》

《天翼云-分布式消息服务-RocketMQ-API 说明文档（C++）.pdf》

3.2 生产消费验证

1）在消息管理台上进行生产消费的拨测验证，验证开通的消息实例和主题。

2）用户应用按照规范接入 RocketMQ，发送、消费消息。

4 SDK 使用

4.1 java sdk 示例

参见：《中国电信 IT 云平台-RocketMQ-API 说明文档(Java)》

4.2 c++ sdk 示例

RocketMQ 中，提供 SDK 供应用使用。SDK 包括 JAVA 与 C++版本。

需要注意的是，C++相比于 JAVA，不提供以下功能：

namesrv 不支持 http 方式

不提供自动创建 Topic 功能

在生产者功能上，不提供事务消息、SendByGroupID(按 Hash 值发送)的功能

在消息者功能上，不提供 PUSH 消费功能

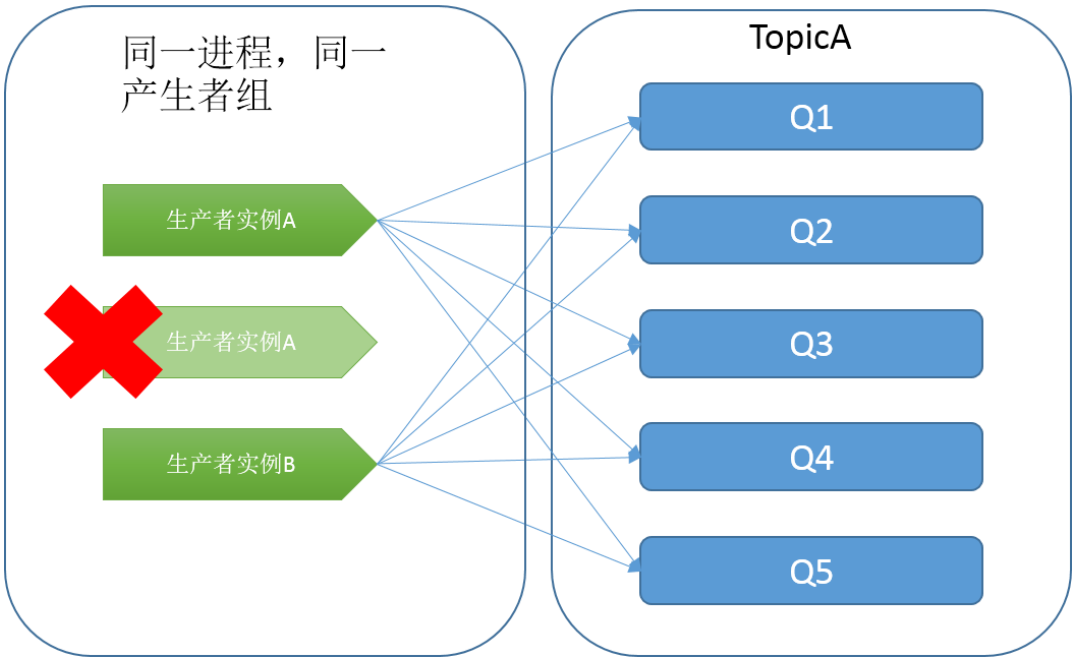
参见：《中国电信 IT 云平台-RocketMQ-API 说明文档(C++).doc》。

5 最佳实践

5.1 生产者

5.1.1 应用进程、生产组、生产实例的关系

同一进程内，同一生产组不能有相同的实例



5.1.2 实例的创建和销毁

- 使用 CTGFactory 进行创建 Producer 实例【强制规范】;
- Prouder 是重量级的实例，每次创建、销毁都会消耗系统资源，建议系统启动的时候创建，系统退出的时候关闭，禁止每次发送消息创建新的实例【强制规范】。

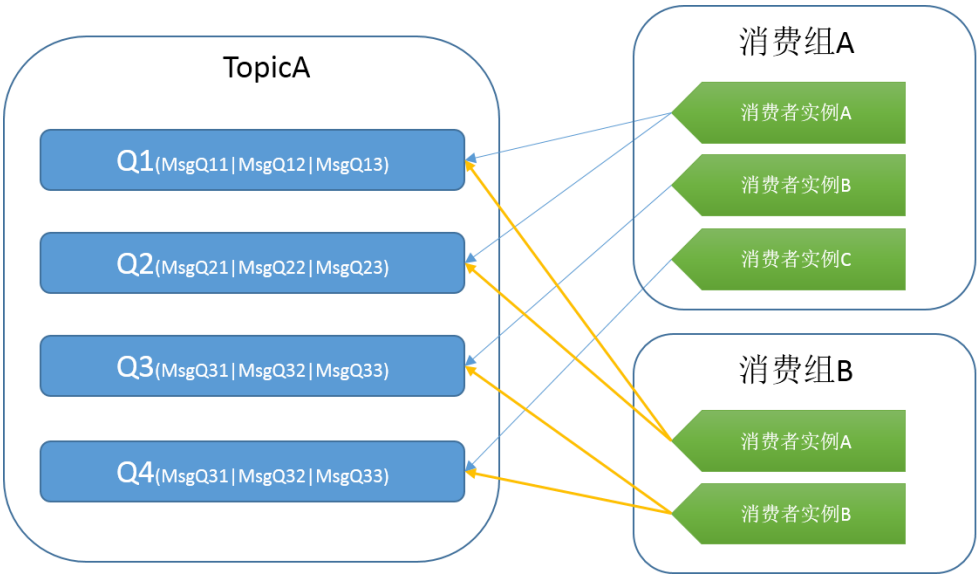
5.1.3 客户端参数建议

常量字段	说明
CompressMsgBodyOverHowmuch	消息体多大时开始压缩，默认 4k
SendMaxRetryTimes	发送失败默认重试次数，默认 2
SendMsgTimeout	发送超时时间，默认 3s
MaxMessageSize	消息最大字节数，默认 1024 * 128，消息体超过该值，消息发送失败，该值需根据具体的业务估算消息体大小设置

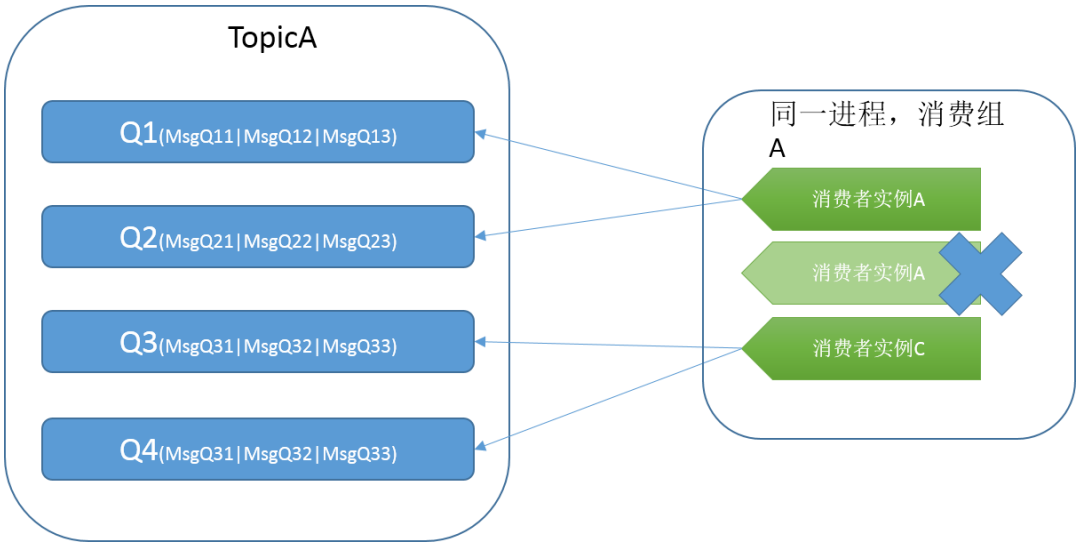
5.2 消费者

5.2.1 应用进程、消费组、消费者实例的关系

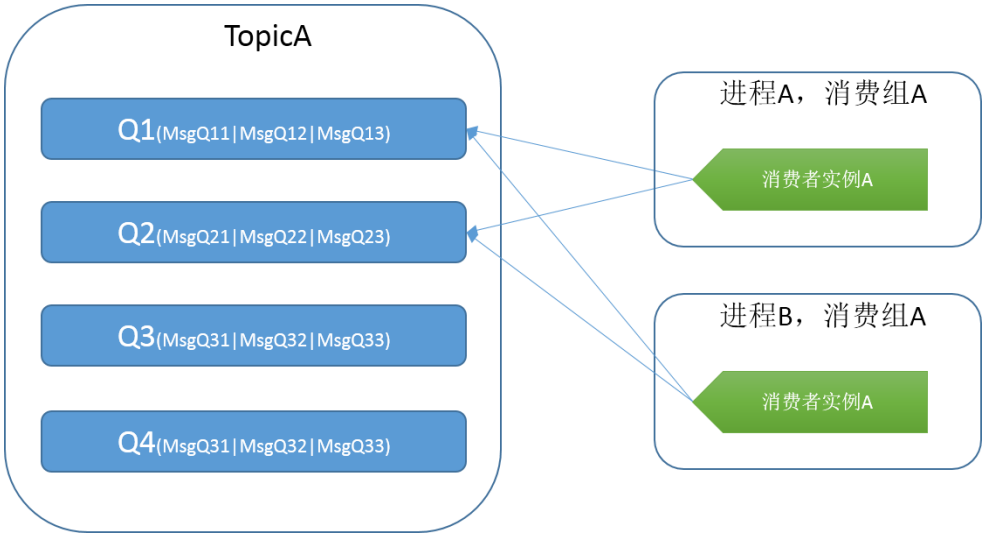
消费组可以有多个消费者实例



同一进程，同一个消费组不允许有相同的消费者实例。



不同进程，不能创建相同消费者实例，可能产生两个实例分配到相同的队列，部分队列却没有消费的情况。



不同进程，不能创建相同消费者实例，如上图中的实例 A，两个实例均连到 Q1 跟 Q2，但 Q3 与 Q4 并无消费。

应用在创建消费者实例时，指定消费者实例名，应用需要保证不同的进程间，同一消费组不能有相同的实例名。或者应用在创建消费者实例时，不指定实例名，RocketMQ 会创建唯一的实例名(JAVA SDK)，规则是：groupName|ip|pid|线程 id|uuid。

5.2.2 同组 Consumer 订阅关系一致

RocketMQ 里的一个 Consumer Group 代表一个 Consumer 群组。对于大多数分布式应用来说，一个 Consumer Group 下通常会有多个 Consumer 实例。订阅关系一致指的是同一个 Consumer Group 下所有 Consumer 实例的处理逻辑必须完全一致，一旦订阅关系不一致，消息消费的逻辑就会混乱，甚至导致消息丢失。

由于 RocketMQ 的订阅关系主要由 Topic+Tag 共同组成，因此，保持订阅关系一致意味着同一个 Consumer group 下所有的实例需在以下两方面均保持一致：

- 订阅的 Topic 必须一致；

订阅的 Topic 中的 Tag 必须一致。

5.2.3 实例的创建和销毁

使用 CTGFactory 进行创建实例【强制规范】；

Consumer 是重量级的实例，每次创建、销毁都会消耗系统资源，建议系统启动的时候创建，系统退出的时候关闭，禁止每次消费消息都创建新的实例【强制规范】。

5.2.4 多消费组消费

一个 Topic，可以使用多个消费组消费消息，每个消费组将在服务端独立保存进度。

集群消费意味着多个消费者均衡消费 Topic 的消息，由于经常不同的程序由多个开发者进行研发和调试，如果使用同一个消费组，在调试过程中存在被其他程序消费者消费的可能，因此应该尽量避免多种类型的应用程序使用同一个消费组。

5.2.5 消费位置设置

消费位置重置可通过控制台进行按时间重置，客户端必须离线。

5.2.6 堆积量

不建议高堆积量的消费，为了预防出现高堆积的情况，建议：

边生产边消费，如果消费速度跟不上，增加消费者。

消费者一直在线，不要等生产了一段时间再开启消费者，这样会造成消费的堆积。

5.3 topic、queue 的规划

在 RocketMQ 中，队列数直接影响到消费者实例数的上限，同一消费组消费者实例数的上限=队列数，需要集群消费的情况，需考虑队列数的设置。

在 RocketMQ 中，队列能分布到不同的 Broker 上，是 RocketMQ 分布式的基础。Queue 分布在 Broker 中，则能使用 Broker 的资源，包括存储、IO 等，一般情况下，分布在某个 Broker 上的 Queue 比例越大，则占用此 Broker 的资源越多，Topic 中的 Queue 分布到的 Broker 数量越多，则性能越好、存储越大。若 Broker 的所在机器性能不同，可以通过调整 Queue 数量，达到资源调优的目的，在应用设计时，需要充分利用上述特性。

在 Push 消费模式中，API 会默认为每个队列预拉取消息 1000 条，若队列数过大、或者单条消息包体过大，则需要考虑设置减少预拉数量，防止预拉消息过大导致内存溢出。

5.4 ava 客户端 Pull 和 Push 的选择

- **Java 客户端必须使用 Push Consumer**

（1） 使用 Pull 可以实现的所有场景，均可使用 Push 实现，并且更简单。

（2） Push 其实是长轮询的 Pull（依然是由客户端发起），在客户端通过配置参数是可以实现流控的，并不会出现服务端的流量压垮客户端的情况。

（3） Push 封装了拉取消息，分发给消费线程的线程模型，非流控的情况下，由后台线程主动拉取消息，并缓存在本地，消费线程池有空闲线程时，分发给消费线程，在有堆积量的情况下，可以保证消费线程一直工作，性能更高（PS:Pull 只提供了拉取消息的功能，并且何时去拉取，拉取时机，这些都需要应用去控制；分发给消费线程的逻辑需要应用封装，除了增加应用工作量外，还可能有不稳定、性能问题等）。

（4） Push 经过多个大型项目的长时间的使用(比如物联网，使用能台，多个试点省份)，更成熟稳定。

（5） Push 会自动订阅重试队列，不需要再次调用拉取重试队列的 API 来取得重试队列的消息（PS:Pull 需要另外调用 API 拉取重试队列的消息）。

Pull 是一种遗留的消费模式（兼容早期的 API），新开发的应用，或者未上线的应用，都要求使用 Push 消费模式。

5.5 有序消费和无序消费的选择

有序消息。

在业务场景允许的情况下，优先选择无序消息，或者在业务能变通的情况下，将有序消息转化为无序消息。

无序消息的优点：

1) 生产者可以使用多进程、多线程往同一个 Topic 发送消息，性能更好。

2) 消费者可以使用多进程、多线程同时消费，性能较好

3) 可以充分使用集群的 Failover 特点，无须依赖自动主备切换（切换过程服务会中断），包括：

✧ 当集群中某一 Broker 节点故障时，不影响业务消息生产，消息将 failover 发送到其它节点

✧ 当集群中某一 Broker 节点故障时，不影响其它节点数据消费，故障恢复后即可消费

4) 能动态地扩容

有序消息的缺点：

1) 对于有序消息，当节点故障时，Queue 数不会变化，生产与消费都会出现异常，直到故障节点恢复。

对于有序消息，需要将所有消息消费完，并且停止客户端，才能扩容。

5.6 消费幂等

（1） RocketMQ 无法避免消息重复，原因主要有以下几点：

- 签收的偏移量是定时（每 5 秒/次）同步到服务端的。
- 为保证消息不丢失，SDK 每次提交的总是队列未签收的最小偏移量（比如无序消费，offset 为 1，2，3，4，5 的消息，1，3，4，5 消费并已签收，2 未签收，签收时最后提交的偏移量将会是 2，如果此时，客户端重启，会从 2 这个位置开始消费）。
- 有网络交互就不能确保每一次的交互数据都是送达的，为保证数据不丢失就要进行重试，有重试就存在重复的可能。

（2） 如果业务对消费重复非常敏感，务必要注意，建议可以采用以下两种方式处理：

- 业务层面做去重处理，可以根据 msgId；如果 key 字段为业务唯一字段，也可采用 key 去重。

业务逻辑实现消费幂等，即多次处理同一消息，对业务的影响是幂等的。

5.7 业务消息设计：Topic 与 Tag

Topic：消息主题，通过 Topic 对不同的业务消息进行分类。

Tag：消息标签，用来进一步区分某个 Topic 下的消息分类，消息队列 RocketMQ 允许消费者按照 Tag 对消息进行过滤，确保消费者最终只消费到他关注的消息类型。

Topic 与 Tag 都是业务上用来归类的标识，区分在于 Topic 是一级分类，而 Tag 可以说是二级分类。

5.8 同组 Consumer 订阅关系一致

RocketMQ 里的一个 Consumer Group 代表一个 Consumer 群组。对于大多数分布式应用来说，一个 Consumer Group 下通常会有多个 Consumer 实例。订阅关系一致指的是同一个 Consumer Group 下所有 Consumer 实例的处理逻辑必须完全一致，一旦订阅关系不一致，消息消费的逻辑就会混乱，甚至导致消息丢失。

由于 RocketMQ 的订阅关系主要由 Topic+Tag 共同组成，因此，保持订阅关系一致意味着同一个 Consumer group 下所有的实例需在以下两方面均保持一致：

- 订阅的 Topic 必须一致；

订阅的 Topic 中的 Tag 必须一致。

5.9 消息 key

每个消息在业务层面的唯一标识码，要设置到 keys 字段，方便将来定位消息丢失问题。服务器会为每个消息创建索引（哈希索引），应用可以通过 topic key 来查询该消息内容。由于是哈希索引，请务必保证 key 唯一，这样可以避免潜在的哈希冲突。

6 消息队列管理

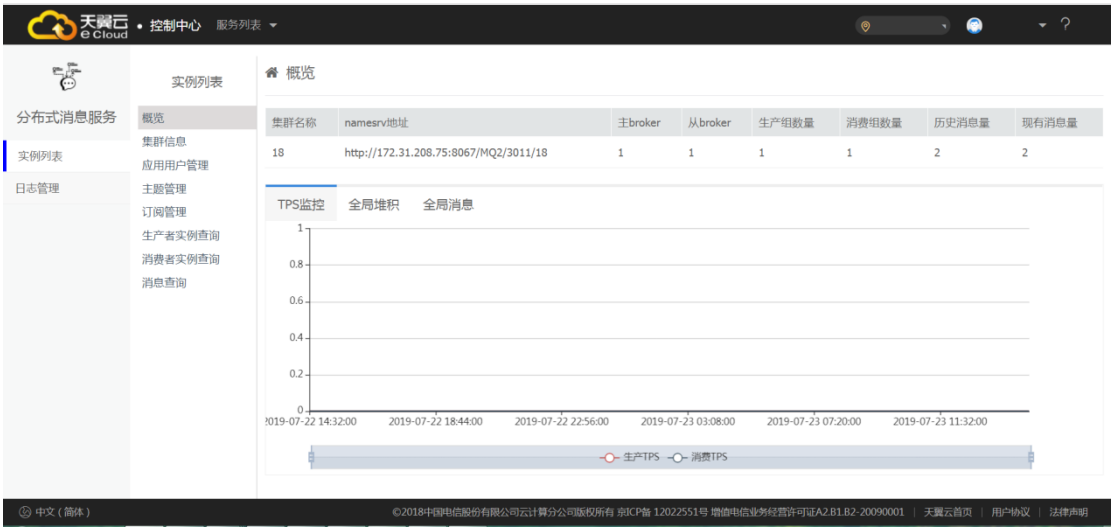
6.1 实例列表

登录管理台后，首页显示的是：该用户可以管理的集群列表，说明该用户，可以对该集群列表进行管理。

点击“管理”，进入该集群的管理页。

6.2 概览

概览界面如下图：

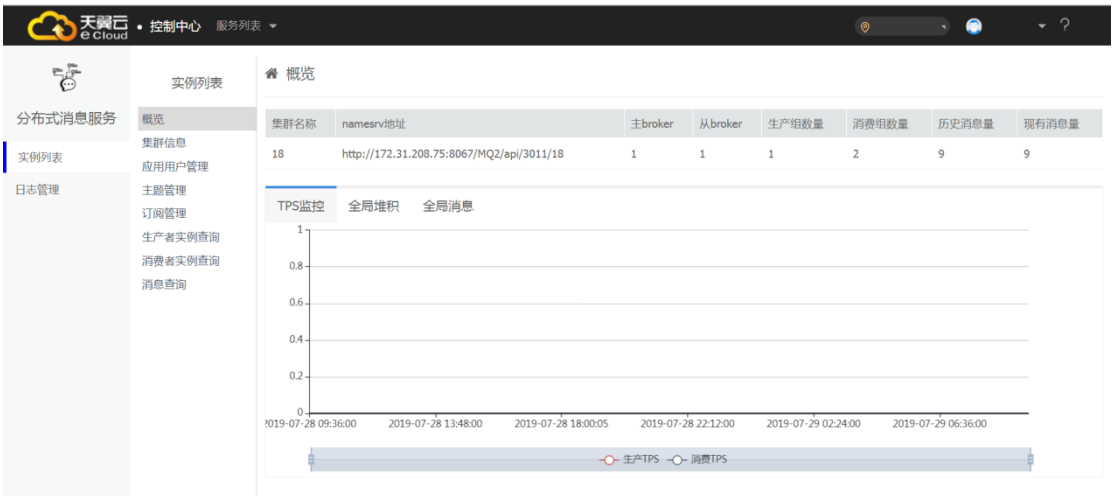


1）表格：显示的是当前集群的统计详情；

2）二维图表：

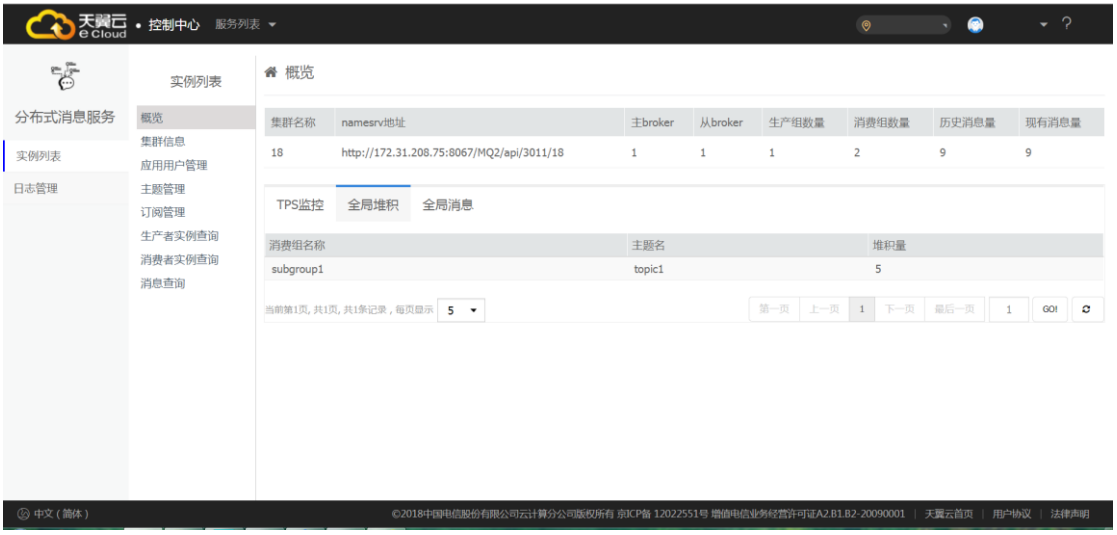
- Tag1:TPS 监控 ：

标识该集群所有 broker 下，所有 topic 的生产、消费 tps(2min)的时间变化曲线；



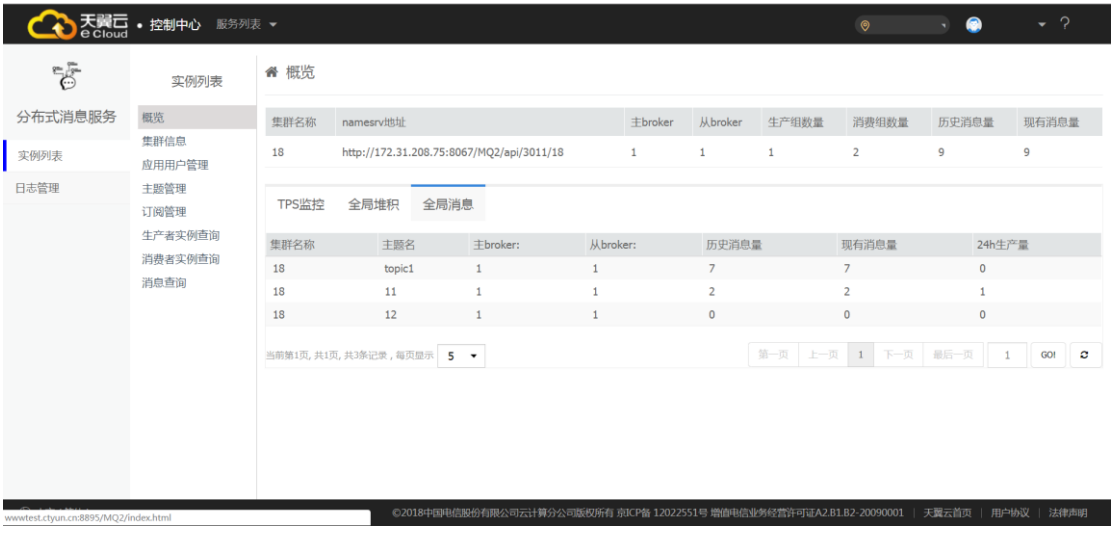
- Tag2:全局堆积

标识该集群下，所有 topic 的消费堆积情况，列表按倒叙排列了堆积量最大的前 20 个 topic 的堆积情况；



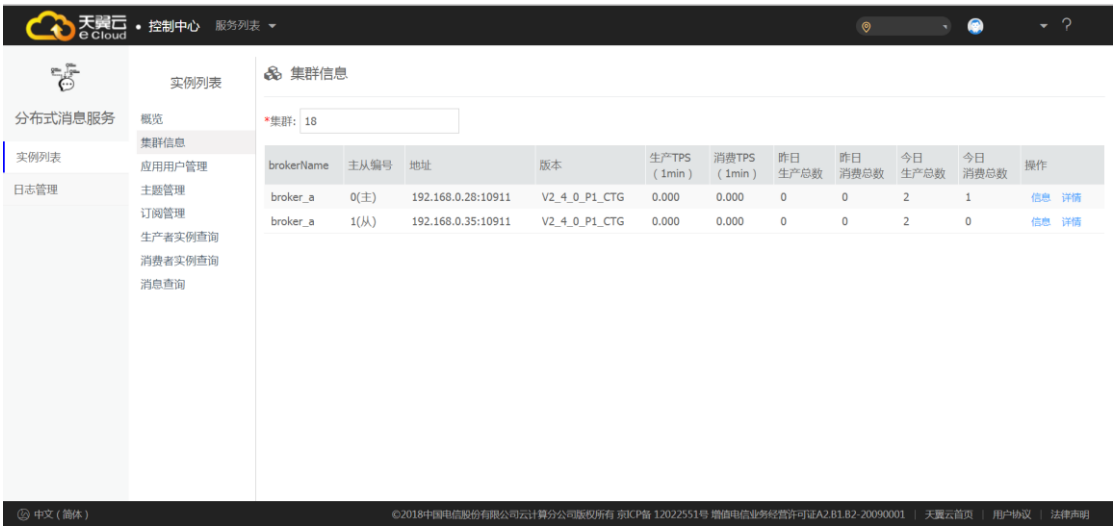
- Tag3：全局消息

  标识该集群，所有 topic 的生产情况，按现有消息量倒叙排列



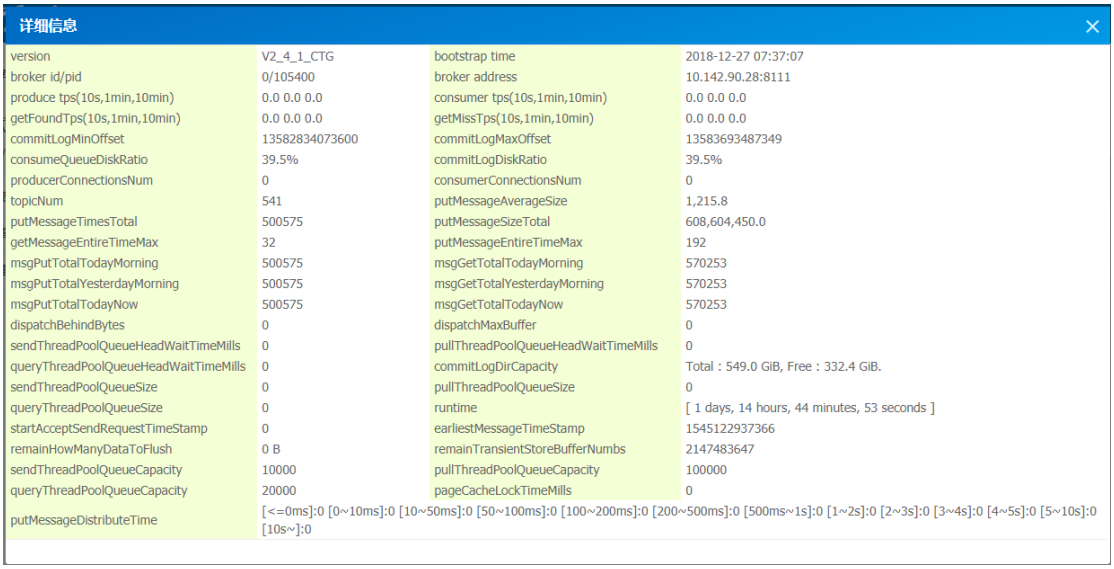
6.3 集群信息

界面展示如下图：



表格：该集群每个 broker 的统计指标概况；

信息：包括该 broker 的详细指标，如下：



详情：

1）上方表格：选定 broker 的统计指标；

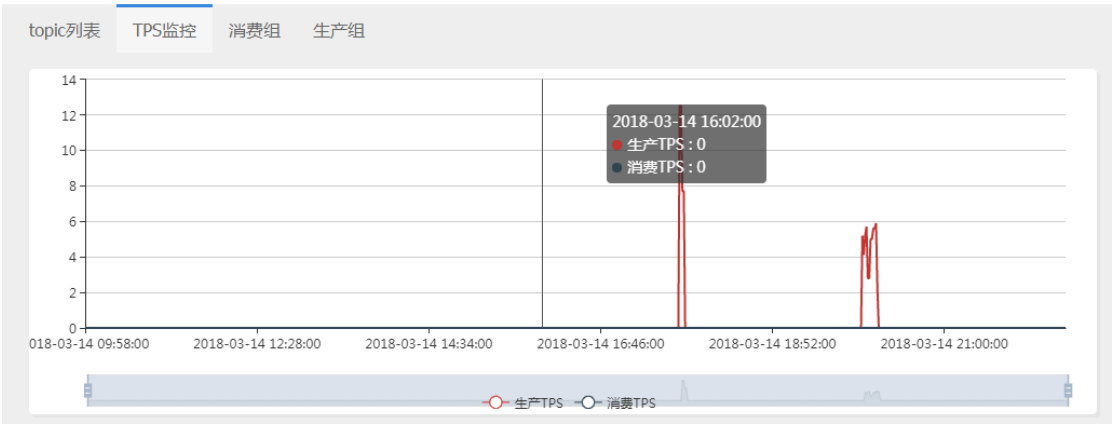
2）下方图表：

- Tag1:Topic 列表

  选定 broker 的所有 topic 的基本配置信息

- Tag2:Tps 监控

选定 broker 的所有 topic 的生产、消费 tps(2min)的时间变化曲线



- Tag3：消费组

  正在连接该 broker 进行消费的消费组

- Tag4:生产组

  正在连接该 broker 进行生产的生产组

6.4 应用用户管理

新建消息实例后，必须在此菜单新建应用用户，然后应用才能在此消息实例上发送、消费消息。此处的用户名即为管控 openapi 中的 accessKey，加密后的密码即为管控 openapi 中的 secretKey。详情见 7.2 章节。

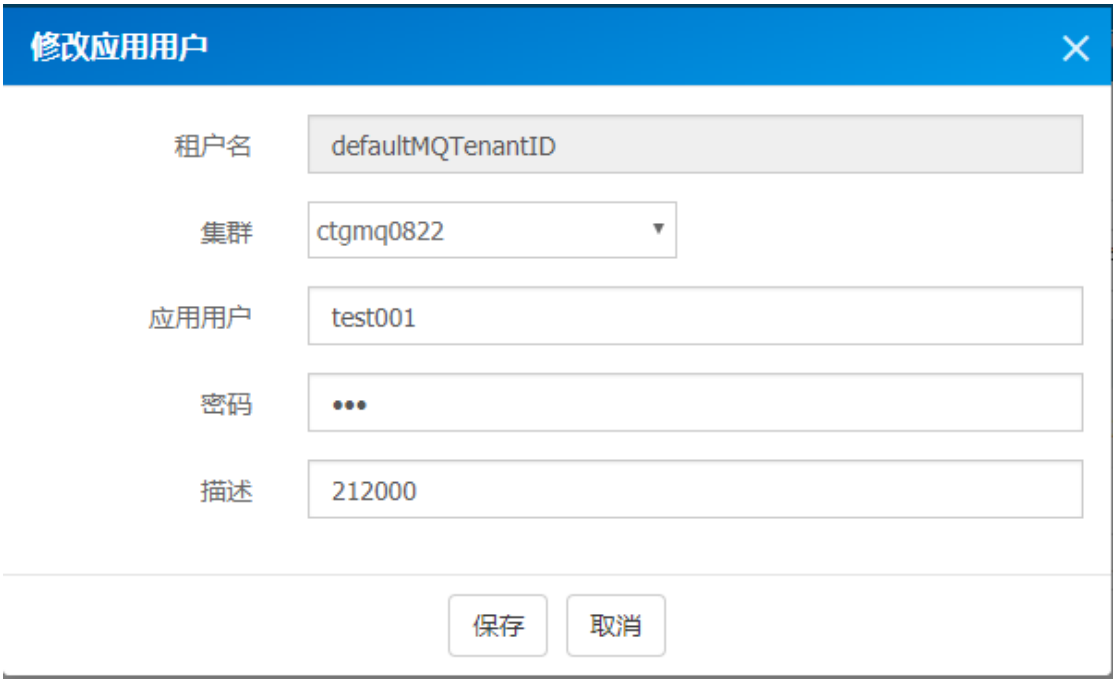
应用用户：指 MQ 客户端，连接服务器生产消费时，需要进行权限校验，所以 MQ 客户端的用户，称为应用用户；

除了用户密码的校验，还可以为用户指定 topic，代表该用户只能生产消费，指定的 topic，其他 topic 不能生产消费。

6.4.1 新建用户

点击“新建用户”，进入应用用户新建页面：

6.4.2 修改用户



6.4.3 删除用户

点击列表“删除”，提示删除成功。

6.5 主题管理

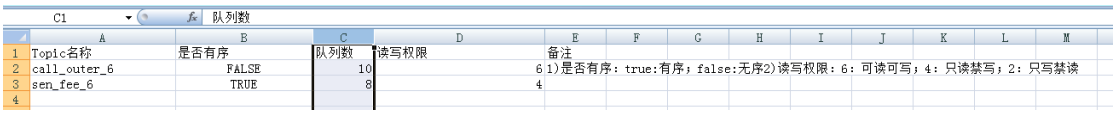
6.5.1 批量创建

注意：注意输入的主题名不要带空格等特殊字符。

通过上传 csv 文件,批量创建主题。格式：点击【主题模板】按钮下载。

6.5.2 主题模板

批量上传主题的模板，必须使用模板，才能够上传成功，模板格式如下：



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Topic名称	是否有序	队列数	读写权限	备注								
2	call_outer_6	FALSE	10	6	1)是否有序：true:有序；false:无序2)读写权限：6：可读可写；4：只读禁写；2：只写禁读								
3	sen_fee_6	TRUE	8	4									
4													

6.5.3 导出

将勾选的 topic，导出其配置及路由情况，为 excel 文件，导出格式如下：

	A	B	C	D	E	F	G
1	topic名称	queue数量	读写权限	broker名称	broker IP	集群名称	是否有序
2	huxl_topic234223	8	4	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
3	testTopic3	16	6	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
4	huxl_topic1213	10	2	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
5	testTopic1	4	6	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
6	testTopic2	16	6	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
7	zipkin	4	6	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
8	huxl_topic23413	10	2	brokeryhh	10.142.90.28:8245	testyhhcluster2	false
9	huxl_topic1123	8	4	brokeryhh	10.142.90.28:8245	testyhhcluster2	true
10							
11							

备注：

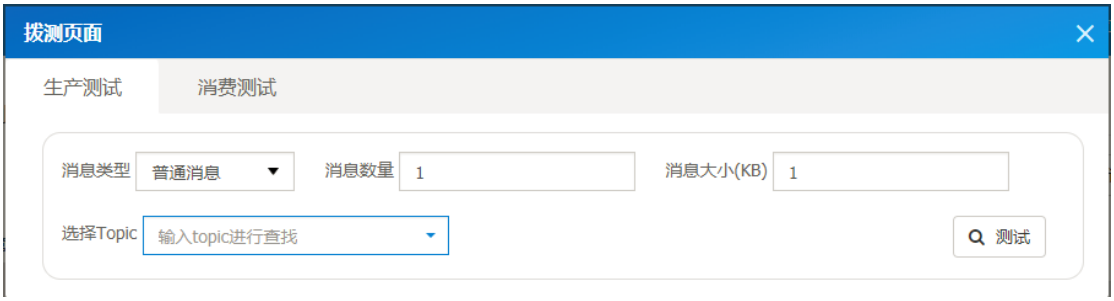
如果勾选了 topic，则导出勾选的 topic;如果没有勾选，则导出该集群，选定 broker 下的所有 topic 信息。

6.5.4 拨测

对集群、topic 的运行情况进行测试，可以进行生产、消费的测试。

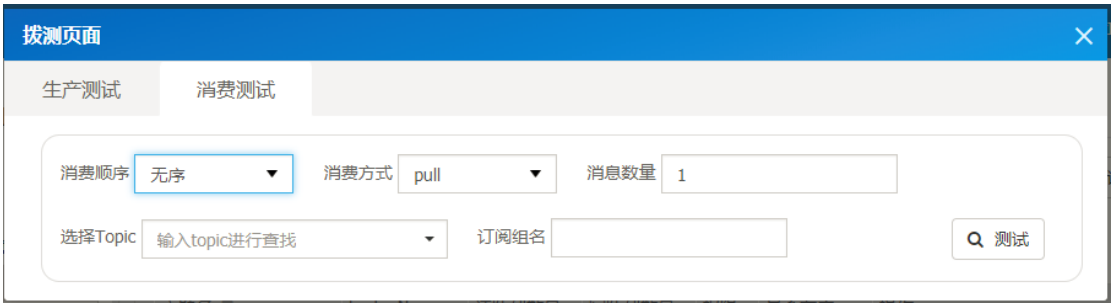
Tag1:生产测试

可设定生产的消息数目，每条消息的大小。



Tag2:消费测试

可设定消费顺序，消费方式，消费数量

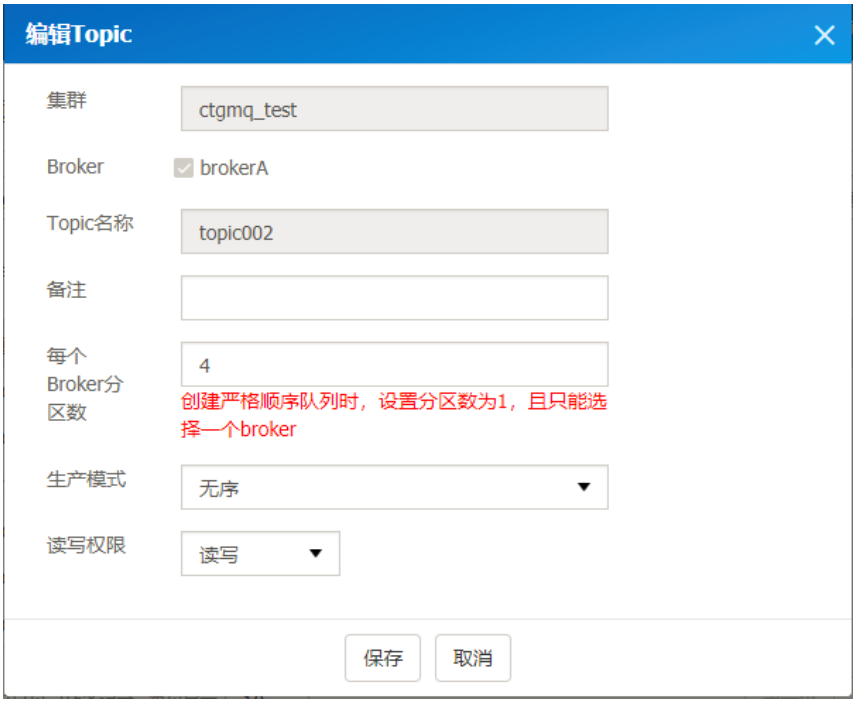


6.5.5 新建主题

每条记录的操作：

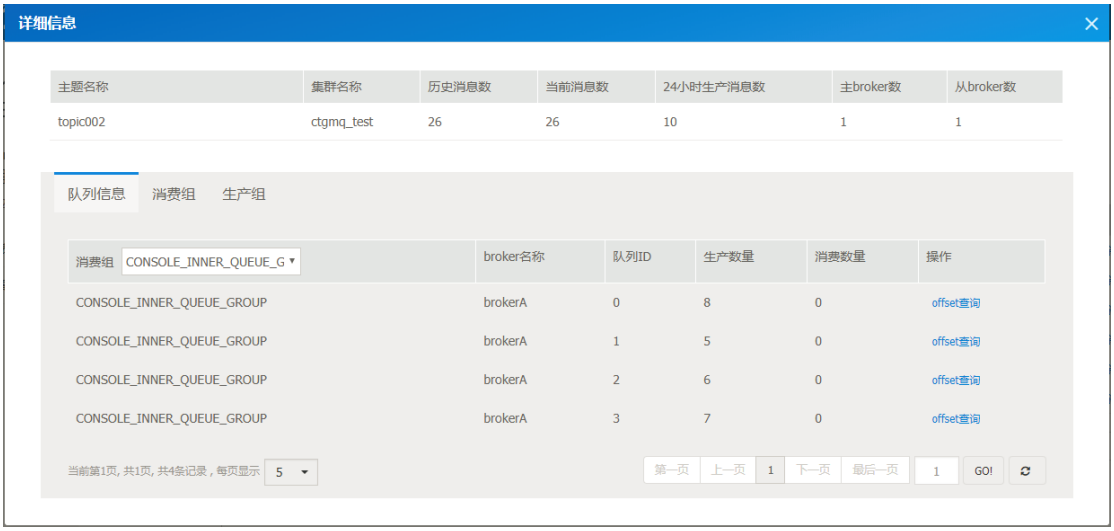
6.5.6 修改主题

主题修改时，不能修改集群、broker、topic 名称；可以修改分区数、生产模式、读写权限、备注；



6.5.7 详情

当前主题的统计指标、队列分布信息、消费组、生产组的情况。



- Tag1：队列信息

该 topic 的队列分布情况，包括每个队列的生产数量、指定消费组的消费数量（CONSOLE_INNER_QUEUE_GROUP 代表还没有被消费过）；

通过“offset 查询”，可以查找队列 相应 偏移量的消息；

通过“查看”，可以查看该消息的，消息包体内容；

- Tag2：消费组

标识：正在连接该 topic 的消费组列表，通过“连接实例”，可以查看该消费组的在线实例列表；

- Tag3:生产组

标识：正在连接该 topic 的生产组列表，通过“连接实例”，可以查看该生产组的在线实例列表；

6.5.8 路由

topic 的队列分布情况，队列分布在哪些 broker 上

Topic路由　无序

代理信息

代理名	代理地址
test0620	0:10.142.233.65:18132
test0612	0:10.142.233.65:16133

队列信息

代理名	读队列	写队列	权限
test0620	8	8	读写
test0612	8	8	读写

6.5.9 堆积量

该 topic 被哪些消费组消费，及该消费组对应的未消费的消息数量统计；

堆积量

Topic名	Broker	消费组名	已消费	未消费	消费组24h消费	未签收消息数	操作
testTopic	test0620	testGroupV1	31027	13	0	0	offset查询
testTopic	test0620	testGroupV2	31035	5	0	0	offset查询
testTopic	test0612	testGroupV1	30766	15	0	0	offset查询
testTopic	test0612	testGroupV2	30772	9	0	0	offset查询

通过“offset 查询”，可以查找队列 相应 偏移量的消息；

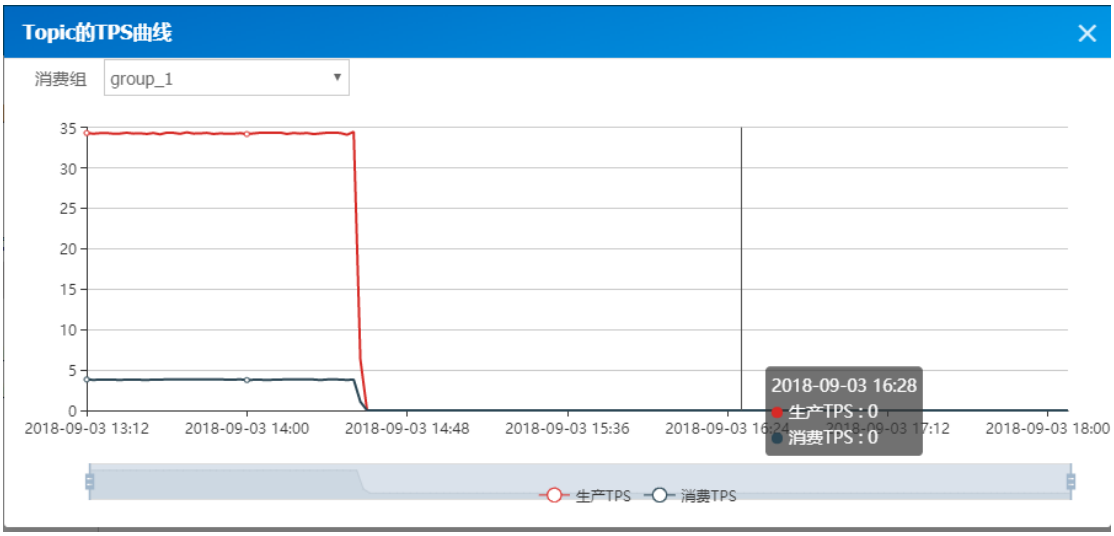
OFFSET查询

*topic: testTopic　*队列ID: 0

订阅组名: testGroupV1　*偏移量: 0　查 询

MsgID	主题	标签	keys	分区ID	逻辑偏移量	物理偏移量	创建时间	存储时间	客户端地址	服务器地址	属性	文件路径	消费状态	查看
0A8EE941000046D400000000000A1B89	testTopic	0	key-7	0	0	662409	2018-06-20 17:22:49	2018-06-20 17:22:49	10.142.233.65:51454	10.142.233.65:18132	{"KEYS":"key-7","UNIQ_KEY":"0A8EE941D0033390975265938EB00006"}	/tmp/rocketmq/msgbodys/0A8EE941000046D400000000000A1B89	CONSUMED	查看

“TPS 监控”：显示指定 topic 的生产、消费的 tps 趋势图。



6.5.10 删除主题

删除主题操作

6.5.11 重置消费位置

使用场景：

当将消费时间重置到历史时间 T1,则 T1 之前生产的消息，都变成已消费，T1 之后生产的消息，都变成未消费；将消费时间重置到未来时间 T2，则存量的所有消息都变成已消费。

备注：

1. 重置时间：是指消息生产的时间

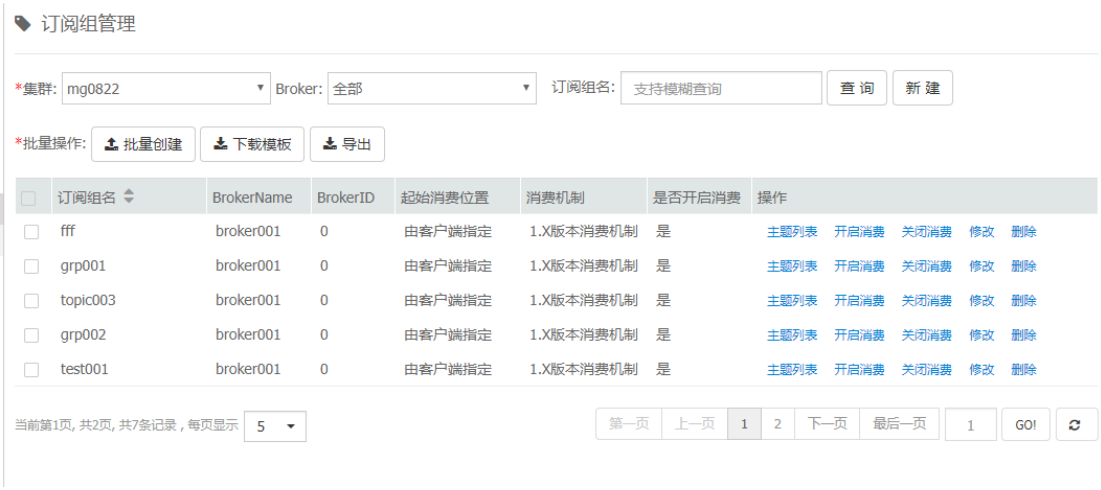
2. 不选时间：所有消息都变成未消费

3 未来时间：所有消息都变成已消费

4. 某时间点：该时间之前变成已消费，改时间之后未消费；

5. v1 消费模式必须关闭 消费客户端

6.6 订阅管理



6.6.1 批量创建

注意：注意输入的订阅组名不要带空格等特殊字符

通过 csv 格式模板上传，批量创建订阅组。格式：点击【订阅组模板】按钮下载。

6.6.2 订阅组模板

指批量上传订阅组的模板，必须使用模板，才能够上传成功，模板格式如下：

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	订阅组名	消费机制	消费起始位	topic权限	备注								
2	test_subgroup_e_1	1	1	*	1)消费机制：1： 由客户端保存消费进度,原生v1版本；2：由服务端通过BDB保存消费进度，V2版								
3	test_subgroup_f_1	2	2	topicA									
4													

6.6.3 下载

将勾选的订阅组，导出其配置及路由情况，为 excel 文件，导出格式如下：

A	B	C	D	E	F	G
订阅组名	起始消费位置	消费机制	集群名称	Broker组名	是否开启	Broker IP
CID_ONS-HTTP-PROXY	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
CID_ONSAPI_PULL	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
CID_ONSAPI_PERMISSION	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
testsub4	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
testsub2	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
CID_ONSAPI_OWNER	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
group_huxl_11	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
testsub3	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245
testSub1	由客户端指定	1.X版本消费机制	testyhhcluster2	brokeryhh	true	10.142.90.28:8245

备注：

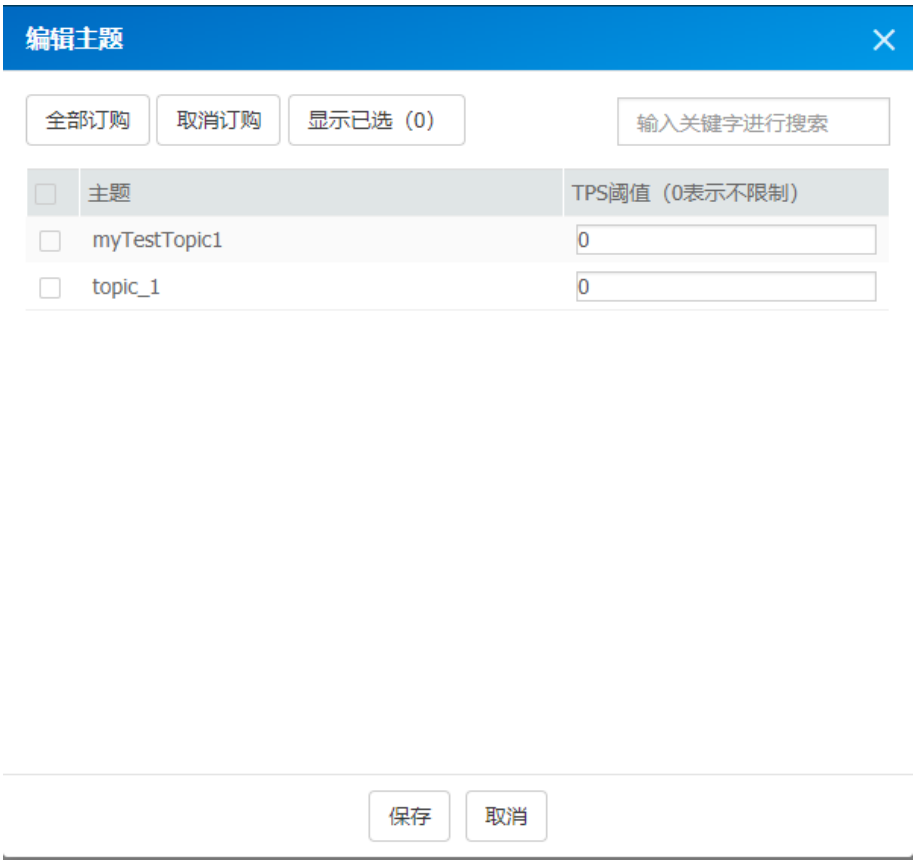
如果勾选了订阅组，则导出勾选的订阅组;如果没有勾选，则导出该集群，选定 broker 下的所有订阅组信息；

6.6.4 新建订阅组

创建订阅组，可以填写基本信息及选择该订阅组感兴趣的 topic。

每条记录的操作按钮：

1. “主题列表”：查看该订阅组可消费的 topic 列表，及消费 TPS 限制情况。



2. “开启消费”:开启订阅组消费功能；

3. “关闭消费”：关闭订阅组消费功能；

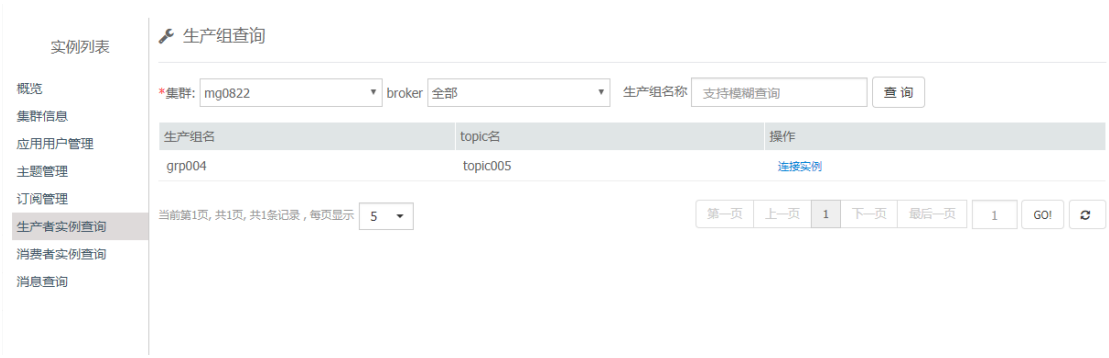
6.6.5 修改订阅组

只能修改订阅组的主题；不能修改基本配置信息。

6.6.6 删除订阅组

删除订阅组及相关主题权限信息。

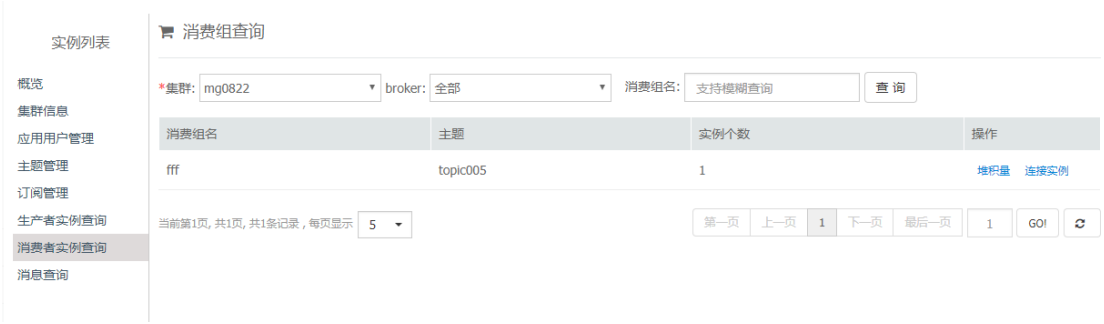
6.7 生产者实例查询



“表格”:列出了指定集群，指定 broker 下的生产组和关联的 topic 的情况；

“连接实例”：显示该生产组，当前在线的生产者实例列表；

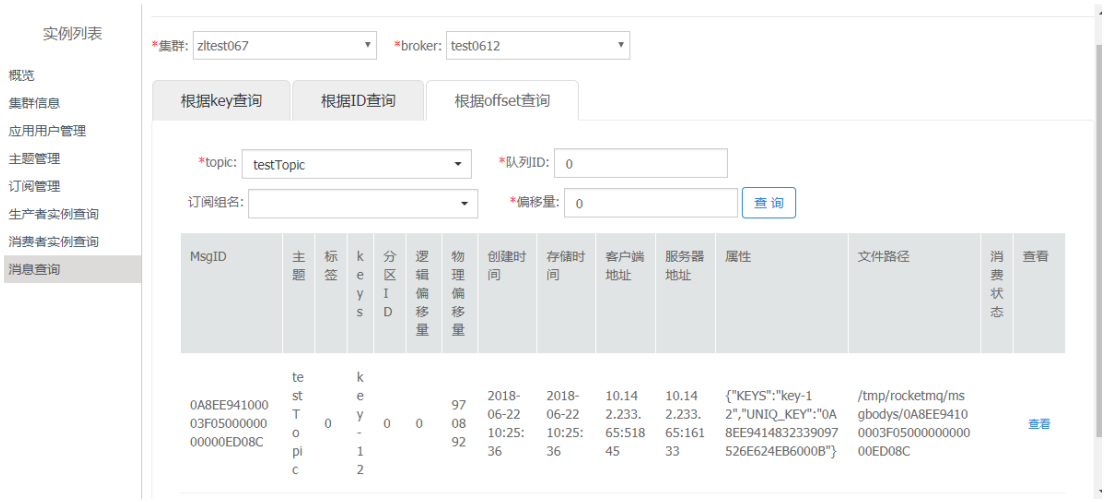
6.8 消费者实例查询



- “表格 1”:列出了，指定集群，指定 broker 下的 消费组消费的情况；
- “连接实例”：显示该消费组，当前在线的消费者实例列表；

- “堆积量”：该消费组消费指定 topic 时，还有多少未消费；

6.9 消息查询



提供三种查询消息的方式：按 key，按 ID，按偏移量。

6.9.1 根据 Key 查询

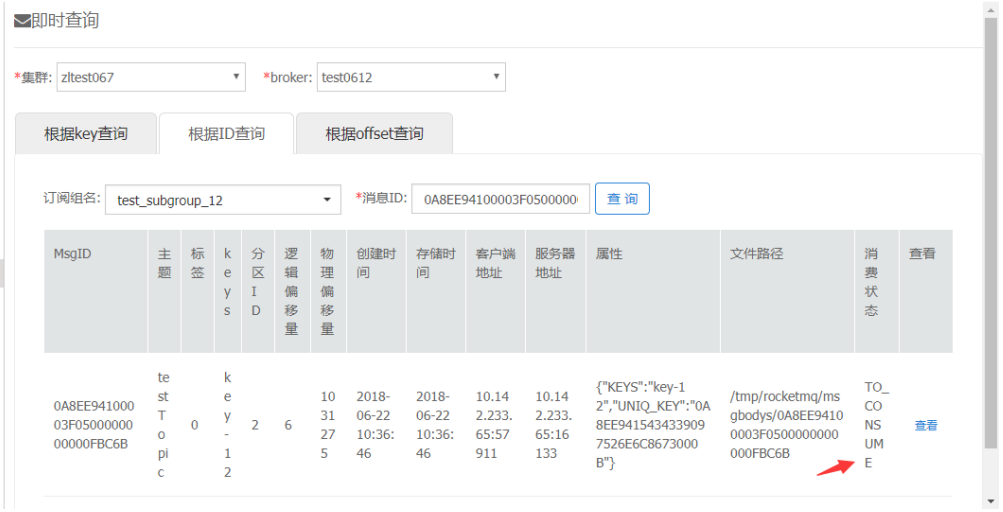
根据消息的 key 查询消息列表，key 要求尽可能全局唯一，每次查询最多返回 64 条 key 相同的记录。

点击“查看”，可以查询该消息的包体内容。

6.9.2 根据 ID 查询

根据消息 ID 查询唯一消息，选择订阅组后，可以查询到，该消息是否被该订阅组消费过，查看“消费状态”；

点击“查看”，可以查询该消息的包体内容



消费状态标志含义：

1）To-consume:未消费

2）Consumed:已签收

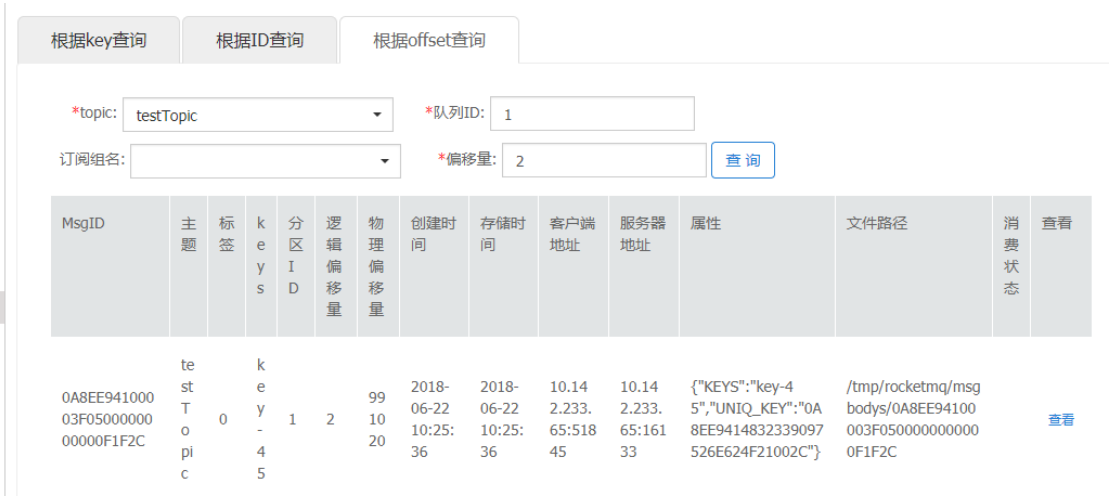
3）Consuming:已拉取，未签收

Tag3:根据 offset 查询

6.9.3 根据偏移量查询

根据指定队列指定偏移量查询唯一消息，选择订阅组后，可以查询到，该消息是否被该订阅组消费过，查看“消费状态”；

点击“查看”，可以查询该消息的包体内容：



7 管控 API

7.1 接口列表

本章介绍消息队列 RocketMQ 的管控 API 信息。用于为用户提供接口的详细功能说明，其中包括对 API 的功能描述、字段元素描述，调用参数的定义及说明，响应参数的元素说明及响应样例，API 调用地址和 API 交互方式等功能。

RocketMQ 提供的接口列表：

接口名	功能	备注
获取实例的详细信息	用于查询实例的详细信息，包括 broker 列表及 broker 的生产消费详细信息	
查询实例的 24h 生产消费 tps 统计数据	查询实例的 24h 生产消费 tps 统计数据	
创建主题	创建主题	
删除主题	删除主题	
修改主题	修改主题	
查询主题详细信息	查询主题详细信息	
查询主题列表	查询主题列表	
查询主题的生产统计信息	查询主题的生产统计信息	
查询主题的消息堆积信息	查询主题的消息堆积信息	
查询主题的生产组列表	查询主题的生产组列表	
查询主题的消费组列表	查询主题的消费组列表	
创建消费组	创建消费组	
修改消费组	修改消费组	

删除消费组	删除消费组	
查询消费组详细信息包括订阅关系	查询消费组详细信息包括订阅关系	
查询消费组列表	查询消费组列表	
开启消费组消费权限	开启消费组消费权限	
关闭消费组消费权限	关闭消费组消费权限	
查询消费组的消费、tps 及堆积信息	查询消费组的消费及堆积信息	
重置消费组进度	消费重置是指用户可以自定义消费位置	重置时间必须在消息持久化存储的时间范围内（默认 72 小时）
查询生产组的实例列表	查询生产组实例列表	
查询消费组的实例列表	查询消费组的实例列表	
获取指定主题-消费组维度的 24h 消费消息统计信息	注意：如果应用方消息量比较小，间隔不均匀，此时采用 TPS 查询可能会导致数据不明显，建议查询消费消息总量	注意：如果应用方消息量比较小，间隔不均匀，此时采用 TPS 查询可能会导致数据不明显，建议查询消费消息总量
根据消息 id 查询消息及消息的消费状态、消息内容	根据消息 id 查询消息及消息的消费状态	注意：查询消费状态必须选择正确的消费组
根据主题查询消息及消息的消费状态、消息内容	根据主题查询消息及消息的消费状态	注意：查询消费状态必须选择正确的消费组
根据消息 key 查询消息及消息的消费状态、消息内容	根据消息 key 查询消息	

7.2 接口访问控制的鉴权信息

1）调用接口时，请求参数必须带上鉴权使用的 accessKey 和 secretKey，accessKey 和 secretKey

RocketMQ 消息管理控制台中的应用用户模块获取，secretKey 是加密的字符串。开通消息队列实例后，必须先创建应用用户，然后才能进行生产、消费消息、使用管控 api。

2）http 的方式获取鉴权信息 accessKey 和 secretKey

新建实例，新建应用用户后，除了在控制台上可以查看 accessKey 和 secretKey；也可以通过 http 接口获取鉴权信息。

1）功能介绍

查询当前用户所属租户的所有消息实例列表

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/access/account

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	开通的消息组件实例名

B、请求示例

```
URL/api/access/account?tenantId=2&accessKey=user1&clusterName=0524
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

其中 data 中的 accountCode 即为 accessKey,accountPasswd 即为 secretKey

C、响应示例：


```
{
    "code":200,
    "message":"success",
    "result":{
          "data":
        {
            "accountId":"166",
            "appId":"RocketMQ",
            "tenantId":"2",
            "tenantName":"ctg",
            "clusterName":"0524",
            "accountCode":"user",

"accountPasswd":"hf96/2MU+Q12fdb9oZN9ghub1OHmUBa8YuW7NJf8Pll/sawcaRVscHTpr4t5SB39+KbJn31Lqy76uEDvj+sgMw==",
            "description":null,
            "createdBy":"admin",
            "createdTime":1561714444000,
            "shardingId":"2"
        }
}
}
```

7.3 接口返回值说明

code=200 代表请求响应成功；非 200 代表异常。

7.4 实例管理接口

7.4.1 查询实例基本信息

1）功能介绍

查询指定实例的集群基本信息，包括主从节点数，实例名，现有消息量等信息。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/cluster/base

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	指定的实例名

B、请求示例

```
URL/api/manager/cluster/base?tenantId=2&&clusterName=0524&&accessKey=user1&&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "code":200,
    "message":"success",
    "result":{
        "data":[
            {
                "mainBrokerNums":2,
                "slaveBrokerNums":2,
                "produceGroupNums":1,
                "consumeGroupNums":0,
                "msgHistoryNums":44643,
                "msgNowNums":44643,
                "clusterName":"0524",
                "namesrvAddr":"",
                "remark":""
            }
        ]
    }
}
```

7.4.2 查询实例的概览信息

1）功能介绍

查询指定实例的集群具体信息，包括主从节点数，实例名，现有消息量等信息。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/cluster/detail

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	指定的实例名

B、请求示例

```
URL/api/manager/cluster/detail?tenantId=2&clusterName=0524&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常

参数	类型	描述
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：


```
{
    "code":200,
    "message":"success",
    "result":{
        "data":{
            "brokerInfoMap":{
                "0524":{
                    "brokerMap":{
                        "a":{
                            "0":{
                                "msgPutTotalTodayNow":"0",
                                "commitLogDiskRatio":"0.36264168995119367",
                                "getFoundTps":"0.0 0.0 0.0",
                                "sendThreadPoolQueueHeadWaitTimeMills":"0",
                                "putMessageDistributeTime":"[<=0ms]:0 [0~10ms]:0 [10~50ms]:0
[50~100ms]:0 [100~200ms]:0 [200~500ms]:0 [500ms~1s]:0 [1~2s]:0 [2~3s]:0 [3~4s]:0 [4~5s]:0 [5~10s]:0 [10s~]:0 ",
                                "queryThreadPoolQueueHeadWaitTimeMills":"0",
                                "remainHowManyDataToFlush":"0 B",
                                "msgGetTotalTodayNow":"0",
                                "brokerPid":"99689",
                                "commitLogMaxOffset":"26433040",
                                "queryThreadPoolQueueSize":"0",
                                "getMessageEntireTimeMax":"0",
                                "msgPutTotalTodayMorning":"0",
                                "putMessageTimesTotal":"1",
                                "bootTimestamp":"1562032127676",
                                "msgGetTotalTodayMorning":"0",
                                "producerConnectionsNum":"0",
                                "msgPutTotalYesterdayMorning":"0",
                                "brokerId":"0",
                                "msgGetTotalYesterdayMorning":"0",
```

```
                                    "pullThreadPoolQueueSize":"0",
                                    "brokerVersionDesc":"V2_4_2_CTG",
                                    "sendThreadPoolQueueSize":"0",
                                    "startAcceptSendRequestTimeStamp":"0",
                                    "commitLogMinOffset":"0",
                                    "putMessageEntireTimeMax":"0",
                                    "topicNum":"13",
                                    "pullThreadPoolQueueHeadWaitTimeMills":"0",
                                    "runtime":"[ 1 days, 5 hours, 3 minutes, 12 seconds ]",
                                    "earliestMessageTimeStamp":"1561616634956",
                                    "commitLogDirCapacity":"Total : 549.0 GiB, Free : 349.9 GiB.",
                                    "dispatchMaxBuffer":"0",
                                    "brokerVersion":"283",
                                    "putTps":"0.0 0.0 0.0",
                                    "remainTransientStoreBufferNumbs":"2147483647",
                                    "getMissTps":"0.0 0.0 0.0",
                                    "consumerConnectionsNum":"0",
                                    "brokerAddress":"10.142.90.29:8400",
                                    "queryThreadPoolQueueCapacity":"20000",
                                    "putMessageAverageSize":"0.0",
                                    "getTransferedTps":"0.0 0.0 0.0",
                                    "dispatchBehindBytes":"0",
                                    "putMessageSizeTotal":"0",
                                    "sendThreadPoolQueueCapacity":"10000",
                                    "getTotalTps":"0.0 0.0 0.0",
                                    "pullThreadPoolQueueCapacity":"100000",
                                    "consumeQueueDiskRatio":"0.36264168995119367",
                                    "pageCacheLockTimeMills":"0"
                                }
                            }
                        },
                        "brokerList":[
                            "a"
                        ]
                    }
                }
            }
        }
    }
}
```


7.4.3 查询实例的 24h 生产消费 tps 统计数据

1）功能介绍

查询指定实例的 24h 生产消费 tps 统计曲线数据。

2）URL

接口类型	HTTP

接口提交方式	GET
接口地址	URL/api/manager/{clusterName}/tps

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	指定的实例名

C、请求示例

```
URL/api/manager/cluster/0524/tps?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

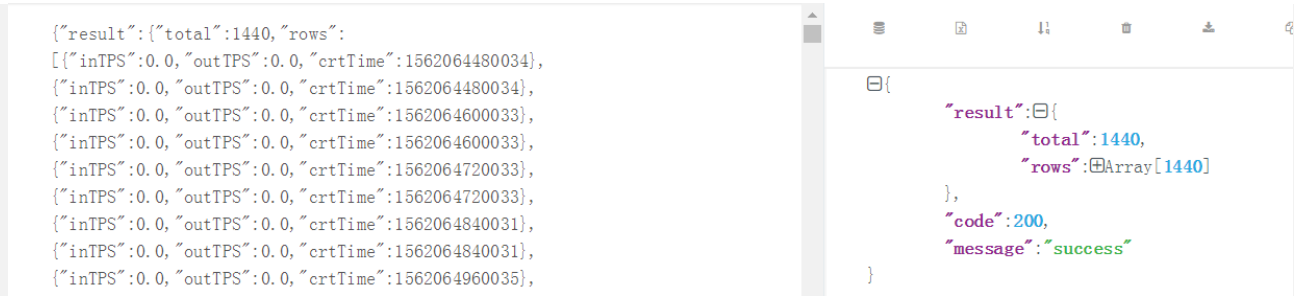
A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：



7.5 主题管理接口

7.5.1 创建主题

1）功能介绍

在实例上创建主题，应用接入实例进行发送消息时，必须先创建主题。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/topics

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取

clusterName	*必填	String	实例名

RequestBody

```
{
    "clusterNameList":[
        "0524"
    ],
    "brokerNameList":[
        "a",
        "b"
    ],
    "writeQueueNums":4,
    "readQueueNums":4,
    "remark":"123",
    "order":false,
    "perm":6,
    "allowdConsumerGroups":[

    ],
    "topicName":"123"
}
```

B、请求示例

```
URL/api/manager/topics?tenantId=2&clusterName=0524&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.5.2 删除主题

1）功能介绍

在实例上创建主题

2）URL

接口类型	HTTP
接口提交方式	DELETE
接口地址	URL/api/manager/topics/{clusterName}/{topicName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名

topicName	*必填	String	主题名

B、请求示例

```
URL/api/manager/topics/0524/testTopic_1?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.5.3 修改主题

1）功能介绍

在实例上修改主题，主要包括修改主题的队列数、读写权限及备注信息等。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/topics/{clusterName}/{topicName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名

RequestBody

```
{
    "clusterNameList":[
        "0524"
    ],
    "brokerNameList":[
        "a",
        "b"
    ],
    "writeQueueNums":4,
    "readQueueNums":4,
```

```
    "remark":"123",
    "order":false,
    "perm":6,
    "allowdConsumerGroups":[

    ],
    "topicName":"123"
}
```

B、请求示例：

```
URL/api/manager/topics/0524/testTopic_1?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "result":{
    },
    "code":200,
```

```
    "message":"success"
}
```

7.5.4 查询主题详细信息

1）功能介绍

在实例上修改主题，主要包括修改主题的队列数、读写权限及备注信息等。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/basedata/{topicName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名

B、请求示例：

```
URL/api/manager/topics/0524/basedata/testTopic_1?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "result":{
        "data":{
            "topicName":"testTopic",
            "readQueueNums":4,
            "writeQueueNums":4,
            "perm":6,
            "topicFilterType":"SINGLE_TAG",
            "topicSysFlag":0,
            "order":false,
            "remark":null,
            "namesrvName":null,
            "clusterName":null,
            "brokerName":"a,b",
            "brokerId":0
        }
    },
    "code":200,
    "message":"success"
}
```

7.5.5 查询主题列表

7.5.6 查询主题的生产统计信息

1）功能介绍

查询主题的在线消费组列表。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/basedatas

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
pageSize	非必填	int	默认 5
pageNumber	非必填	int	默认 1

B、请求示例：

URL/api/manager/topics/0524/basedatas?tenantId=2pageNumber=1&pageSize=5&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：


```
{
    "result":{
        "total":3,
        "rows":[

            {
                "topicName":"testTopic",
                "readQueueNums":4,
                "writeQueueNums":4,
                "perm":6,
                "topicFilterType":"SINGLE_TAG",
                "topicSysFlag":0,
                "order":false,
                "remark":null,
                "namesrvName":null,
                "clusterName":"0524",
                "brokerName":"a",
                "brokerId":0
            },
```

```
            {
                "topicName":"testTopic",
                "readQueueNums":4,
                "writeQueueNums":4,
                "perm":6,
                "topicFilterType":"SINGLE_TAG",
                "topicSysFlag":0,
                "order":false,
                "remark":null,
                "namesrvName":null,
                "clusterName":"0524",
                "brokerName":"b",
                "brokerId":0
            },
            {
                "topicName":"topic_1",
                "readQueueNums":4,
                "writeQueueNums":4,
                "perm":6,
                "topicFilterType":"SINGLE_TAG",
                "topicSysFlag":0,
                "order":false,
                "remark":null,
                "namesrvName":null,
                "clusterName":"0524",
                "brokerName":"a",
                "brokerId":0
            }
        ]
    },
    "code":200,
    "message":"success"
}
```


7.5.7 查询主题的队列的消息统计信息

1）功能介绍

查询主题的在线消费组列表。

2）URL

接口类型	HTTP
接口提交方式	GET

接口地址	URL/api/manager/topics//{clusterName}/{topicName}/queues

3）请求参数列表

A、参数说明：

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名

B、请求示例：

URL/api/manager/topics/0524/testTopic/queues?tenantId=2&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：


```
{
    "result":{
        "total":1,
        "rows":[
            {
                "topic":"testTopic",
                "consumerGroup":"testGroup",
                "diffTotal":44682,
                "lastTimestamp":1562146038387,
                "queueStatInfoList":[
                    {
                        "brokerName":"a",
                        "queueId":0,
                        "brokerOffset":5591,
                        "consumerOffset":13,
                        "clientInfo":null,
                        "lastTimestamp":1562145061438
                    },
                    {
                        "brokerName":"a",
                        "queueId":1,
                        "brokerOffset":5592,
                        "consumerOffset":12,
                        "clientInfo":null,
                        "lastTimestamp":1562146038387
                    }
                ]
            }
        ]
    },
    "code":200,
    "message":"success"
}
```


7.5.8 查询主题的消息堆积信息

1）功能介绍

查询主题的在线消费组列表。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/{topicName}/depth

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
pageSize	非必填	int	默认 5
pageNumber	非必填	int	默认 1

B、请求示例：

```
URL/api/manager/topics/0524/testTopic/depth?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：


```
{
    "result":{
        "total":2,
        "rows":[
            {
                "diff":22364,
                "total":22365,
                "consumedTotal":1,
                "retryNums":0,
                "groupName":"testGroup",
                "brokerName":"a",
                "topicName":"testTopic",
                "outTps":0,
                "outTotalMsgToday":0,
                "namesvrId":0,
                "offSetDiff":0,
                "notAckMsgNum":0,
                "notAckMsgTime":0,
                "notConsumeMsgTime":515654097
            },
            {
                "diff":22377,
                "total":22377,
```

```
                    "consumedTotal":0,
                    "retryNums":0,
                    "groupName":"testGroup",
                    "brokerName":"b",
                    "topicName":"testTopic",
                    "outTps":0,
                    "outTotalMsgToday":0,
                    "namesvrId":0,
                    "offSetDiff":0,
                    "notAckMsgNum":0,
                    "notAckMsgTime":0,
                    "notConsumeMsgTime":528696676
                }
            ]
        },
        "code":200,
        "message":"success"
}
```


7.5.9 查询主题的生产组列表

1）功能介绍

获取指定主题的在线生产组列表。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/{topicName}/producers

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取

secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
pageNumber	非必填	int	默认 1
pageSize	pageSize	int	默认 5

B、请求示例：

```
URL/api/manager/topics/0524/testTopic/producers?pageNumber=1&pageSize=5&tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
  "result":{
```

```
        "total":1,
        "rows":[
            "CLIENT_INNER_PRODUCER"
        ]
    },
    "code":200,
    "message":"success"
}
```


7.5.10 查询主题的消费组列表

1）功能介绍

获取指定主题的在线消费组列表。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/{topicName}/consumers

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
pageNumber	非必填	int	默认 1

pageSize	pageSize	int	默认 5

B、请求示例：

```
URL/api/manager/topics/0524/testTopic/consumers?pageNumber=1&pageSize=5&tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
  {
    "result":{
          "total":1,
          "rows":[
                "testGroup"
          ]
    },
    "code":200,
    "message":"success"
}
```

7.6 消费组管理接口

7.6.1 创建消费组

1）功能介绍

在实例上创建消费组，应用接入实例进行消费时，必须先创建消费组。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/subscriptions/{clusterName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名

RequestBody:

```
{
  "subscriptionGroupConfig":{
        "consumeEnable":true,
        "firstConsumeMechanism":"1",
        "groupName":"123",
        "pullMechanism":"1",
        "subscribeMap":{

        }
  },
  "brokerNameList":[
```

```
            "a",
            "b"
        ],
        "clusterNameList":[
            "0110"
        ],
        "remark":"123"
}
```

B、请求示例：

```
    URL/api/manager/subscriptions/0524?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.6.2 修改消费组

1）功能介绍

在实例上修改消费组。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/subscriptions/{clusterName}/{groupName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
groupName	*必填	String	消费组名

RequestBody：

```
{
  "subscriptionGroupConfig":{
        "consumeEnable":true,
        "firstConsumeMechanism":"1",
        "groupName":"123",
        "pullMechanism":"1",
        "subscribeMap":{

        }
  },
```

```
    "brokerNameList":[
        "a",
        "b"
    ],
    "clusterNameList":[
        "0110"
    ],
    "remark":"123"
}
```

B、请求示例：

```
URL/api/manager/subscriptions/0524/test_group?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例：

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.6.3 删除消费组

1）功能介绍

在实例上修改消费组。

2）URL

接口类型	HTTP
接口提交方式	DELETE
接口地址	URL/api/manager/subscriptions/{clusterName}/{groupName}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
groupName	*必填	String	消费组名

B、请求示例：

```
URL/api/manager/subscriptions/0524/test_group?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.6.4 查询消费组详细信息包括订阅关系

1）功能介绍

在实例上创建消费组，应用接入实例进行消费时，必须先创建消费组。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/subscriptions/{clusterName}/{groupName}/detail

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
groupName	*必填	String	消费组名

B、请求示例

URL/api/manager/subscriptions/0524/testGroup/detail?tenantId=2&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
    "result":{
        "data":{
            "clusterNameList":[
                "0524"
            ],
            "brokerNameList":[
                "a",
                "b"
            ],
            "subscriptionGroupConfig":{
                "groupName":"testGroup",
                "consumeEnable":true,
                "consumeFromMinEnable":true,
                "consumeBroadcastEnable":true,
                "retryQueueNums":1,
                "retryMaxTimes":16,
                "brokerId":0,
                "whichBrokerWhenConsumeSlowly":1,
                "notifyConsumerIdsChangedEnable":true,
                "pullMechanism":1,
                "firstConsumeMechanism":1,
                "subscribeMap":{

                },
                "allowdAllTopics":false
            },
            "remark":null
        }
    },
    "code":200,
    "message":"success"
}
```


7.6.5 查询消费组列表

1）功能介绍

在实例上创建消费组，应用接入实例进行消费时，必须先创建消费组。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/subscriptions/{clusterName}/basedatas

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
pageSize	非必填	int	默认 5
pageNumber	非必填	int	默认 1

B、请求示例

```
URL/api/manager/subscriptions/0524/basedatas?tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态

参数	类型	描述
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
   "result":{
        "total":2,
        "rows":[
            {
                "groupName":"testGroup",
                "consumeEnable":true,
                "consumeFromMinEnable":true,
                "consumeBroadcastEnable":true,
                "retryQueueNums":1,
                "retryMaxTimes":16,
                "brokerId":0,
                "whichBrokerWhenConsumeSlowly":1,
                "notifyConsumerIdsChangedEnable":true,
                "pullMechanism":1,
                "firstConsumeMechanism":1,
                "subscribeMap":{

                },
                "remark":null,
                "namesrvAddr":null,
                "clusterName":null,
                "brokerName":"a",
                "type":0,
                "allowdAllTopics":false
            },
            {
                "groupName":"testGroup",
                "consumeEnable":true,
                "consumeFromMinEnable":true,
                "consumeBroadcastEnable":true,
                "retryQueueNums":1,
```

```
                    "retryMaxTimes":16,
                    "brokerId":0,
                    "whichBrokerWhenConsumeSlowly":1,
                    "notifyConsumerIdsChangedEnable":true,
                    "pullMechanism":1,
                    "firstConsumeMechanism":1,
                    "subscribeMap":{

                    },
                    "remark":null,
                    "namesrvAddr":null,
                    "clusterName":null,
                    "brokerName":"b",
                    "type":0,
                    "allowdAllTopics":false
                }
            ]
        },
        "code":200,
        "message":"success"
}
```


7.7 消费管理接口

7.7.1 开启、关闭消费组消费权限

1）功能介绍

在实例上开启、关闭消费组。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/subscriptions/{clusterName}/{groupName}/openStatus

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
brokerNames	*必填	String	Broker 列表，逗号分隔的 broker 列表的 String
groupName	*必填	String	消费组名
isOpen	*必填	int	0 开启；1 关闭

B、请求示例

URL/api/manager/subscriptions/0524/testGroup/openStatus?brokerNames=%5B%0A++%22a%22,%0A++%22b%22%0A%5D&isOpen=1&tenantId=2?tenantId=2&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例

```
{
    "result":{
    },
    "code":200,
    "message":"success"
}
```

7.7.2 查询消费组的消费、tps 及堆积信息

1）功能介绍

查询消费组的消费、tps 及堆积信息，包括重试队列。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/subscriptions/{clusterName}/{groupName}/depth

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
groupName	*必填	String	消费组名
topicName	*必填	String	主题名

B、请求示例

URL/api/manager/subscriptions/0524/testGroup/depth?tenantId=2&accessKey=user1&secretKey=user1&topicName=testTopic

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
 "result":{
      "total":2,
      "rows":[
           {
                "diff":9872,
                "total":54743,
                "consumedTotal":44871,
                "retryNums":0,
                "groupName":"testGroup",
                "brokerName":"["a","b"]",
                "topicName":"testTopic",
                "outTps":2.1333333333333333,
                "outTotalMsgToday":44682,
                "namesvrId":0,
```

```
                "offSetDiff":0,
                "notAckMsgNum":0,
                "notAckMsgTime":0,
                "notConsumeMsgTime":102503
            },
            {
                "diff":0,
                "total":0,
                "consumedTotal":0,
                "retryNums":0,
                "groupName":"testGroup",
                "brokerName":"["a","b"]",
                "topicName":"%RETRY%testGroup",
                "outTps":0,
                "outTotalMsgToday":0,
                "namesvrId":0,
                "offSetDiff":0,
                "notAckMsgNum":0,
                "notAckMsgTime":0,
                "notConsumeMsgTime":0
            }
        ]
    },
    "code":200,
    "message":"success"
}
```


7.7.3 重置消费组进度

1）功能介绍

在实例上创建消费组，应用接入实例进行消费时，必须先创建消费组。

2）URL

接口类型	HTTP
接口提交方式	POST
接口地址	URL/api/manager/topics/consumeOffset

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名

RequestBody

```
{
  "consumerGroupList":[
        "testGroup"
  ],
  "resetForward":true,
  "topicName":"testTopic",
  "force":true,
  "resetTime":"1562083200000",
  "pageNumber":1,
  "pageSize":1000000000
}
```

B、请求示例

URL/api/manage/topics/consumeOffset?clusterName=0524&tenantId=2&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
   "result":{
        "total":8,
        "diffOffset":101,
        "rows":[
            {
                "brokerName":"a",
                "queueId":0,
                "brokerOffset":5591,
                "consumerOffset":5591,
                "timestampOffset":5579,
                "rollbackOffset":5579
            },
            {
                "brokerName":"b",
                "queueId":0,
                "brokerOffset":5595,
                "consumerOffset":5595,
                "timestampOffset":5582,
                "rollbackOffset":5582
            },
            {
                "brokerName":"a",
                "queueId":1,
                "brokerOffset":5592,
                "consumerOffset":5592,
                "timestampOffset":5579,
                "rollbackOffset":5579
            },
            {
                "brokerName":"b",
                "queueId":1,
                "brokerOffset":5594,
                "consumerOffset":5594,
                "timestampOffset":5581,
                "rollbackOffset":5581
            },
            {
                "brokerName":"a",
                "queueId":2,
                "brokerOffset":5591,
                "consumerOffset":5591,
                "timestampOffset":5579,
```

```
                "rollbackOffset":5579
            },
            {
                "brokerName":"b",
                "queueId":2,
                "brokerOffset":5595,
                "consumerOffset":5595,
                "timestampOffset":5582,
                "rollbackOffset":5582
            },
            {
                "brokerName":"a",
                "queueId":3,
                "brokerOffset":5592,
                "consumerOffset":5592,
                "timestampOffset":5579,
                "rollbackOffset":5579
            },
            {
                "brokerName":"b",
                "queueId":3,
                "brokerOffset":5593,
                "consumerOffset":5593,
                "timestampOffset":5581,
                "rollbackOffset":5581
            }
        ]
    },
    "code":200,
    "message":"success"
}
```


7.8 生产消费统计接口

7.8.1 获取指定主题-消费组维度的 24h 生产、消费消息 tps 统计信息

1）功能介绍

获取指定主题-消费组维度的 24h 生产、消费消息统计信息。如果输入消费组为空则只查询生产统计。

注意：生产和消费是分开统计的，时间点不一样。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/topics/{clusterName}/{topicName}/tps

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
brokerName	*必填	String	实例中的某个 broker 名
topicName	*必填	String	主题名
groupName	*必填	String	消费组名，可以设置为空

B、请求示例

```
URL/api/manager/topics/0524/testTopic/tps?brokerName=a&groupName=testGroup&tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态

参数	类型	描述
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

```
 {
                    "inTPS":0,//生产 tps
                    "outTPS":0,//消费 tps
                    "crtTime":1562148960015//时间戳
}
```

C、响应示例

```
{
    "result":{
        "taotal":4,
        "inTpslist":[
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562148960015
            },
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562149080019
            }
        ],
        "outTpslist":[
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562081160019
            },
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562120280015
            }
        ],
        "totalTpslist":[
            {
                "inTPS":0,
```

```
                "outTPS":0,
                "crtTime":1562062920018
            },
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562136840015
            },
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562136840016
            },
            {
                "inTPS":0,
                "outTPS":0,
                "crtTime":1562136960014
            }
        ]
    },
    "code":200,
    "message":"success"
}
```


7.9 消息查询接口

7.9.1 根据消息 id 查询消息

1）功能介绍

通过传入 Message ID 查询指定消息的信息以及判断该指定的消息是否曾被消费过。查询到的信息包括发送时间、存储服务器和消息的 Key 和 Tag 等属性。

根据 Message ID 查询消息的方式属于精确查询，查询条件需要的 Message ID 从每次消息发送成功的 SendResult 中获取。输入消费组会查询该条消息的消费状态。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/messages/{msgId}

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
groupName	非必填	String	查询消息是否被该消费者效果过时需要该参数
msgId	*必填	String	

B、请求示例

```
URL/api/manager/messages/0A8E5A1D000020D00000000001943B5E?clusterName=0524&groupName=testGroup&tenantId=2&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
    "result":{
        "data":{
            "queueId":1,
            "storeSize":1198,
            "queueOffset":5591,
            "sysFlag":0,
            "bornTimestamp":1562146038373,
            "bornHost":"10.142.90.31:52112",
            "storeTimestamp":1562146038387,
            "storeHost":"10.142.90.29:8400",
            "msgId":"0A8E5A1D000020D00000000001943B5E",
            "commitLogOffset":26491742,
            "bodyCRC":1949382714,
            "reconsumeTimes":0,
            "preparedTransactionOffset":0,
            "topic":"testTopic",
            "properties":{
                "KEYS":"1562146038371",
                "UNIQ_KEY":"AC181F0113AC1BE6F5C30E0B8E650064",
                "TAGS":"webtest"
            },
            "messageBody":"WebTestTools_1562146038371",
            "status":"TO_CONSUME",
            "messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1D000020D00000000001943B5E"
        }
    },
    "code":200,
    "message":"success"
}
```


7.9.2 根据偏移量 offset 查询消息

1）功能介绍

通过传入 Topic 名称和逻辑偏移量 offset 进行查询，属于精确查询。输入消费组会查询该条消息的消费状态。

2）URL

接口类型	HTTP

接口提交方式	GET
接口地址	URL/api/manager/messages/offset

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
brokerName	*必填	int	默认 10
topicName	*必填	int	默认 1
queueId	*必填	int	
queueOffset	*必填	int	
groupName	非必填	String	查询消息是否被该消费者效果过时需要该参数

B、请求示例

```
URL/api/manager/messages/messages/offset?brokerName=b&clusterName=0524&groupName=testGroup&
queueId=0&queueOffset=0&tenantId=2&topicName=testTopic&accessKey=user1&secretKey=user1
```

4）响应参数列表

A、响应参数说明：

参数	类型	描述

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
    "result":{
        "data":{
            "queueId":0,
            "storeSize":1198,
            "queueOffset":0,
            "sysFlag":0,
            "bornTimestamp":1561616634774,
            "bornHost":"10.142.90.28:52992",
            "storeTimestamp":1561616634785,
            "storeHost":"10.142.90.30:8430",
            "msgId":"0A8E5A1E000020EE00000000000004AE",
            "commitLogOffset":1198,
            "bodyCRC":1091717825,
            "reconsumeTimes":0,
            "preparedTransactionOffset":0,
            "topic":"testTopic",
            "properties":{
                "KEYS":"1561616634770",
                "UNIQ_KEY":"AC140001A0E81BE6F5C388FC47960001",
                "TAGS":"webtest"
            },
            "messageBody":"WebTestTools_1561616634770",
            "status":"TO_CONSUME",
            "messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE00000000000004AE"
        }
    },
    "code":200,
    "message":"success"
}
```

7.9.3 根据消息 key 查询消息

1）功能介绍

通过传入 Topic 名称和 Message Key 进行模糊查询，得到符合条件的消息的信息列表。

使用本接口根据 Message Key 查询消息的方式属于模糊查询。由于业务方的 Key 可能不唯一，所以查询结果可能为多条。

2）URL

接口类型	HTTP
接口提交方式	GET
接口地址	URL/api/manager/messages/key

3）请求参数列表

A、参数说明:

参数名	选项	参数类型	说明
tenantId	*必填	String	租户 id
accessKey	*必填	String	鉴权使用的 accessKey，由消息管理控制台应用用户菜单中获取
secretKey	*必填	String	鉴权使用的 secretKey，由消息管理控制台应用用户菜单中获取
clusterName	*必填	String	实例名
topicName	*必填	String	主题名
pageSize	非必填	int	默认 5
pageNumber	非必填	int	默认 1
key	*必填	String	消息 key

B、请求示例

URL/api/manager/messages/messages/key?brokerName=b&clusterName=0524&key=1561616634770&pageNumber=1&pageSize=5&tenantId=2&topicName=testTopic&accessKey=user1&secretKey=user1

4）响应参数列表

A、响应参数说明：

参数	类型	描述
code	string	状态码，200 代表响应成功，其它代表响应异常
message	string	描述状态
result	string	返回对象。此参数所包含的参数请见“B、result 参数说明”

B、result 参数说明：

参数	类型	描述
data	string	创建结果描述

C、响应示例


```
{
    "result":{
        "total":10,
        "rows":[
            {
                "queueId":0,
                "storeSize":1198,
                "queueOffset":0,
                "sysFlag":0,
                "bornTimestamp":1561616634774,
                "bornHost":"10.142.90.28:52992",
                "storeTimestamp":1561616634785,
                "storeHost":"10.142.90.30:8430",
                "msgId":"0A8E5A1E000020EE00000000000004AE",
                "commitLogOffset":1198,
                "bodyCRC":1091717825,
                "reconsumeTimes":0,
                "preparedTransactionOffset":0,
                "topic":"testTopic",
                "properties":{
                    "KEYS":"1561616634770",
                    "UNIQ_KEY":"AC140001A0E81BE6F5C388FC47960001",
                    "TAGS":"webtest"
                },
```

```
                "messageBody":"WebTestTools_1561616634770",
                "status":null,

"messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE00000000000004AE"
            },
            {
                "queueId":1,
                "storeSize":1198,
                "queueOffset":0,
                "sysFlag":0,
                "bornTimestamp":1561616634891,
                "bornHost":"10.142.90.28:52992",
                "storeTimestamp":1561616634896,
                "storeHost":"10.142.90.30:8430",
                "msgId":"0A8E5A1E000020EE000000000000095C",
                "commitLogOffset":2396,
                "bodyCRC":1091717825,
                "reconsumeTimes":0,
                "preparedTransactionOffset":0,
                "topic":"testTopic",
                "properties":{
                    "KEYS":"1561616634770",
                    "UNIQ_KEY":"AC140001A0E81BE6F5C388FC480B0002",
                    "TAGS":"webtest"
                },
                "messageBody":"WebTestTools_1561616634770",
                "status":null,

"messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE000000000000095C"
            },
            {
                "queueId":2,
                "storeSize":1198,
                "queueOffset":0,
                "sysFlag":0,
                "bornTimestamp":1561616634898,
                "bornHost":"10.142.90.28:52992",
                "storeTimestamp":1561616634902,
                "storeHost":"10.142.90.30:8430",
                "msgId":"0A8E5A1E000020EE0000000000000E0A",
                "commitLogOffset":3594,
                "bodyCRC":1091717825,
                "reconsumeTimes":0,
                "preparedTransactionOffset":0,
                "topic":"testTopic",
                "properties":{
                    "KEYS":"1561616634770",
                    "UNIQ_KEY":"AC140001A0E81BE6F5C388FC48120003",
                    "TAGS":"webtest"
                },
                "messageBody":"WebTestTools_1561616634770",
```

```
                "status":null,

"messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE0000000000000E0A"
            },
            {
                "queueId":3,
                "storeSize":1198,
                "queueOffset":1,
                "sysFlag":0,
                "bornTimestamp":1561616634905,
                "bornHost":"10.142.90.28:52992",
                "storeTimestamp":1561616634909,
                "storeHost":"10.142.90.30:8430",
                "msgId":"0A8E5A1E000020EE00000000000012B8",
                "commitLogOffset":4792,
                "bodyCRC":1091717825,
                "reconsumeTimes":0,
                "preparedTransactionOffset":0,
                "topic":"testTopic",
                "properties":{
                    "KEYS":"1561616634770",
                    "UNIQ_KEY":"AC140001A0E81BE6F5C388FC48190004",
                    "TAGS":"webtest"
                },
                "messageBody":"WebTestTools_1561616634770",
                "status":null,

"messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE00000000000012B8"
            },
            {
                "queueId":0,
                "storeSize":1198,
                "queueOffset":1,
                "sysFlag":0,
                "bornTimestamp":1561616634982,
                "bornHost":"10.142.90.28:52992",
                "storeTimestamp":1561616634986,
                "storeHost":"10.142.90.30:8430",
                "msgId":"0A8E5A1E000020EE0000000000001766",
                "commitLogOffset":5990,
                "bodyCRC":1091717825,
                "reconsumeTimes":0,
                "preparedTransactionOffset":0,
                "topic":"testTopic",
                "properties":{
                    "KEYS":"1561616634770",
                    "UNIQ_KEY":"AC140001A0E81BE6F5C388FC48660009",
                    "TAGS":"webtest"
                },
                "messageBody":"WebTestTools_1561616634770",
                "status":null,
```

```
"messageBodyPath":"/tmp/rocketmq/msgbodys/0A8E5A1E000020EE0000000000001766"
                }
            ]
        },
        "code":200,
        "message":"success"
}
```

8 常见问题

8.1 服务端异常

1）地址冲突

在启动 broker 时，出现“地址已在使用”，如：

```
load config properties file OK, ../conf/2m-noslave/broker-b.properties
java.net.BindException: 地址已在使用
        at sun.nio.ch.Net.bind0(Native Method)
        at sun.nio.ch.Net.bind(Net.java:444)
        at sun.nio.ch.Net.bind(Net.java:436)
        at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:214)
        at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74)
        at io.netty.channel.socket.nio.NioServerSocketChannel.doBind(NioServerSocketChannel.java:125)
        at io.netty.channel.AbstractChannel$AbstractUnsafe.bind(AbstractChannel.java:484)
        at io.netty.channel.DefaultChannelPipeline$HeadContext.bind(DefaultChannelPipeline.java:1080)
        at io.netty.channel.AbstractChannelHandlerContext.invokeBind(AbstractChannelHandlerContext.jav
        at io.netty.channel.AbstractChannelHandlerContext.bind(AbstractChannelHandlerContext.java:415)
        at io.netty.channel.DefaultChannelPipeline.bind(DefaultChannelPipeline.java:903)
        at io.netty.channel.AbstractChannel.bind(AbstractChannel.java:197)
        at io.netty.bootstrap.AbstractBootstrap$2.run(AbstractBootstrap.java:350)
        at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.ja
        at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:357)
        at io.netty.util.concurrent.SingleThreadEventExecutor$2.run(SingleThreadEventExecutor.java:116
        at java.lang.Thread.run(Thread.java:745)
```

解决方案：修改配置文件里面的 listenPort 的值，然后重新启动。

```
[root@bdserver3 2m-2s-async]# vim broker-a.properties
brokerClusterName=DefaultCluster
brokerName=broker-a
brokerId=0
deleteWhen=04
fileReservedTime=48
brokerRole=ASYNC_MASTER
flushDiskType=ASYNC_FLUSH
namesrvAddr= 172.31.25.24:9876
brokerIP1=172.31.25.24
listenPort=10911
storePathRootDir=/data2/broker-a-s/store
storePathCommitLog=/data2/broker-a-s/store/commitlog
```

2）brokerName 不匹配

启动出现异常：broker-b does not match the expected group name: broker-a

原因：启动其他 Master broker 服务时，直接将之前使用过的 store 目录以及 bdb 目录复制过来，仅仅只是修改了 brokerName，导致此问题出现。

解决方案：2.0 以后版本 brokerName 一旦创建启动后就不能改变，否则只能删除 store 目录才能解决。

3）service not available

现象：发送消息一定量的时候，出现 create maped file failed, please make sure OS and JDK both 64bit。或者当 topic 的队列数位 1024 个的时候，会出现 service not available now, maybe disk full，maybe your broker machine memory too small。

解决：使用 ulimit-a 命令查询系统参数，检查 open files 是否超过 655350，max memory size 是为否 unlimited，若不是，需要重新根据安装手册的步骤，重新调整系统参数。

```
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 4133973
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 655360
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) unlimited
cpu time               (seconds, -t) unlimited
max user processes              (-u) 16384
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

4）磁盘空间不足

当磁盘空间大于 85%时，会出现“ CODE: 14 DESC: service not available now, maybe disk full, CL: 0.87 CQ: 0.87 INDEX: 0.87, maybe your broker machine memory too small.”的异常。

处理办法：

消息中间件有两种策略， 包括数据高安全性与服务高可靠性，分别如下

策略	配置	说明
数据高安全性	cleanFileForciblyEnable=false	若磁盘使用率大于85%，有消息生产时或默认凌晨四点，则触发删除过期的消息，若没过期消息则不会被删除
服务高可靠性	cleanFileForciblyEnable=true	若磁盘使用率大于85%，有消息生产时或默认凌晨四点，则触发删除消息（在有效期内的数据将被删除）

若磁盘使用率大于 85%，策略为数据高安全性，且无过期文件，可以按实际需求，减少数据保存时间来触发消息删除，腾出磁盘空间

1. 使用 updateBrokerConfig 命令，修改 fileReservedTime 属性，此属性为消息保存时间，单位为小时。按需减少保存时间，则可以腾出磁盘空间。

2. 主备都需要同时修改

消息过期时间的配置可以参考文档 RocketMQ 配置文件详细说明.docx，fileReservedTime 配置

5）通过 deamon 拉起 broker 时报错

deamon.log 日志中报 Fail to queryBrokerMaxOffset

```
        at java.lang.Thread.run(Thread.java:745) [na:1.7.0_79]
2017-09-20 14:58:00 ERROR monitorSchseduledThread1 - fail to queryBrokerMaxOffset, exception:
com.alibaba.rocketmq.remoting.exception.RemotingConnectException: connect to <192.168.25.134:10911> failed
        at com.alibaba.rocketmq.remoting.netty.NettyRemotingClient.invokeSync(NettyRemotingClient.java:662) ~[ctgmq-rocketmq-remoting-2.2.6_P2.jar:na]
        at com.ctg.mq.deamon.remoting.RemotingWrapper.getBrokerRuntimeInfo(RemotingWrapper.java:157) ~[ctgmq-rocketmq-deamon-2.2.6_P2.jar:na]
        at com.ctg.mq.deamon.broker.BrokerMonitorService.queryBrokerMaxOffset(BrokerMonitorService.java:144) [ctgmq-rocketmq-deamon-2.2.6_P2.jar:na]
        at com.ctg.mq.deamon.broker.BrokerMonitorService.getReportData(BrokerMonitorService.java:116) [ctgmq-rocketmq-deamon-2.2.6_P2.jar:na]
        at com.ctg.mq.deamon.broker.BrokerMonitorService.run(BrokerMonitorService.java:70) [ctgmq-rocketmq-deamon-2.2.6_P2.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471) [na:1.7.0_79]
        at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:304) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:178) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615) [na:1.7.0_79]
        at java.lang.Thread.run(Thread.java:745) [na:1.7.0_79]
[ctgmq@study02 logs]$
```

原因：

1. 配置文件错误。

2. 做过主备切换，然后手动干预或重启集群，启动进程的地址和角色与 zookeeper 中存储的不同，造成启动失败。

3. 上次启动失败后未清理错误数据

解决方法：

1. 删除 zk 中该集群的信息

2. 核对配置文件，确保端口和路径有效

3. 删除 running 目录

4. 重新通过自动部署或者手动部署启动进程

6）消费进度停滞不前

现象：通过 consumerProgress 命令查询消费进度某些队列无变化，而客户端正在正常消费

原因：某些队列有消息没有签收，导致服务端消费进度没有后移。

解决：

通过 consumerProgress 命令显示的 consumer offset 找到对应消息

1. 如果是 BDB 消费模式，重启应用即可或者通过以下 api

void com.ctg.mq.api.IMQAckHandler.ackMessageSuccess(String msgID)签收卡住的消息即可。

2. 如果是原生有序消费模式，重启应用即可。

3. 如果是原生无序消费模式，启动一个同消费组的实例，会将该消息签收。

7）删除 topic 失败

现象：删除 topic 时出现 topic **** is consuming by consumer ****，或者 topic *** is publishing by producer ***异常

原因：删除 topic 必须没有生产者和消费者正在订阅该 topic（与该 topic 相关的生产者消费者都必须离线），否则会失败。

解决：

可以通过一下方式查看是否还有客户端连接该 topic：

管理平台->主题管理->详情->生产组|消费组->连接实例

如果使用命令行删除有序队列，需要使用集群删除，例如：

sh mqadmin deleteTopic -n 10.142.90.33:9876 -c mq_cluster -t mytesttopic

如果使用命令行删除无序队列，可以使用 broker 删除，例如：

sh mqadmin deleteTopic -n 10.142.90.33:9876 -b 10.142.90.33:10911 -t mytesttopic

8）启动 broker 时 BDB 报错

现象：启动 broker 时，报下面的错误

原因：可能是迁移了 store 目录或者更换了 broker 的组名、地址或端口。

解决：删除 store 目录下的 consumeStore 目录，重启 broker 即可解决。

9） 从 broker 已启动，但 clusterList 看不到

现象：从 broker 已启动，但无法加入到集群（clusterList 查询不到）

原因：

（1）查看/etc/hosts 文件，机器名与 IP 的映射关系是否填写有误

（2）查看防火墙设置（是否有端口未开放），listenPort 到 listenPort+2 的端口都需要开放（如果主 broker 的 listenPort=10911，那么 10911、10912、10913 都要开放）

10） 通过命令行创建有序 topic，但是 web 管理台显示的是无序的

现象：通过 updateTopic 命令，加-o true 创建有序 topic，但是 web 端查询的时候显示是无序的。

原因：集群有多个 namesrv，但是创建的时候只填了一个 namesrv。

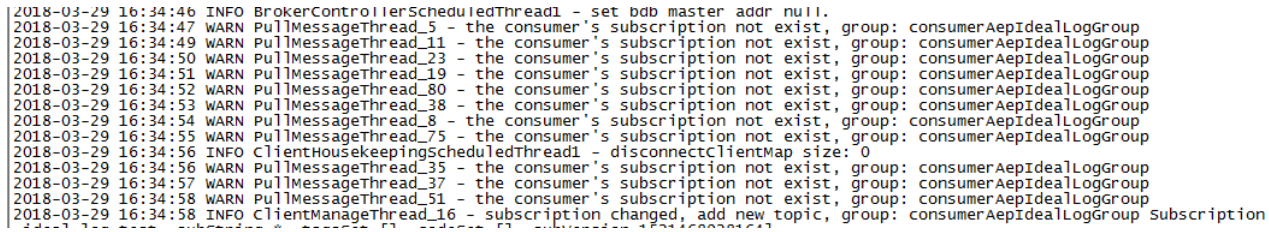
解决：

创建时加上这个 broker 集群的所有 namesrv，中间用分号分割，例如：

sh mqadmin updateTopic -n “10.142.90.30:9876;10.142.90.28:9876” -t crmTopic -o true

11）消费者订阅关系不存在

现象：broker.log 日志报错，the consumer's subscription not exist, group: consumerAepIdealLogGroup



原因：使用同个订阅组，同时消费不同的 topic，订阅关系会被覆盖。

解决：不能使用同个订阅组的消费组去订阅不同的 topic，如果需要变换订阅关系，请关闭旧消费者。

12）使用 clusterList 查询 主 TPS 不为 0，从 TPS 一直为 0

这种情况最大的可能是从同步出错，可以做进一步的确认，查看 store.log 或者 stoererror.log，一般会看到有持续的报错信息。这种情况可以删除从的 store 目录，重新进行同步。

前提：部署有高可用模块或者主的 brokerRole=ASYNC_MASTER，否则停止从的时候，生产会报错。

操作步骤：

（1） 手工停止从 broker（kill pid，注意不要加-9，如果自动拉起 broker 参数设置为 true，则需要先关掉从的 deamon）

（2） 删除或者备份从的 store 目录（为保险起见，空间允许的话，可以 mv 备份，不要直接删除）

（3） 手工启动 broker（sh sh/broker_*.sh）

（4） 从 broker 启动完成后，用 clusterList 查看，可以看到从的 TPS 比较高，因为正在同步。

13）主 broker 异常恢复

需要走异常恢复流程的一般是 consumequeue 生成有问题，导致无法拉取消息（注意有多种情况会导致无法拉取消息，不一定是 consumequeue 有问题，注意判断）。根据 offset 查询报错。

异常恢复流程：

（1） 停止需要恢复这一组 broker 的主从 deamon，主从 broker

（2） 删除主 broker store 目录下的 checkpoint consumequeue consumeStore index（也可以 mv 改下名字来备份）

（3） 检查 store 目录下的 abort 文存是否存在，如果不存在新建一个（touch abort）

（4） 启动主 broker，查看 store.log，可以看到打印恢复过程的日志，如果没有报错，说明恢复成功

（5） 如果 commitlog 文件比较多，可能恢复时间较长，可以通过查看 store.log 或者 broker 端口是否起来判断恢复是否完成。

（6） 主 broker 起来后，通过消费或者根据 offset 查询消息来验证是否恢复成功。

（7） 如果主 broker 恢复成功，启动从 broker，启动主从 deamon。

14）RPC 异常（所以服务端组件均可能出现）

异常信息

```
2018-11-09 16:36:08 ERROR NettyServerCodecThread_2 - decode exception, 192.168.10.1:53436
io.netty.handler.codec.TooLongFrameException: Adjusted frame length exceeds 16777216: 1195725860 - discarded
        at io.netty.handler.codec.LengthFieldBasedFrameDecoder.fail(LengthFieldBasedFrameDecoder.java:499)
        at io.netty.handler.codec.LengthFieldBasedFrameDecoder.failIfNecessary(LengthFieldBasedFrameDecoder.java:477)
        at io.netty.handler.codec.LengthFieldBasedFrameDecoder.decode(LengthFieldBasedFrameDecoder.java:403)
        at org.apache.rocketmq.remoting.netty.NettyDecoder.decode(NettyDecoder.java:43)
```

原因：使用非组件 RPC 协议访问导致，比如用 http 协议、或者 telnet 等，均可导致 decode 错误，

解决办法：无需解决，应服务端及客户端 RPC 请求均无影响。

8.2 应用客户端

1）连接拒绝 Connect faild

当出现：

```
2015-11-10 12:00:00 WARN BrokerControllerScheduledThread1 - registerBroker Exception, 134.130.134.46:9876
com.alibaba.rocketmq.remoting.exception.RemotingConnectException: connect to <134.130.134.46:9876> failed
        at com.alibaba.rocketmq.remoting.netty.NettyRemotingClient.invokeSync(NettyRemotingClient.java:641) ~[rocketmq-remoting-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.out.BrokerOuterAPI.registerBroker(BrokerOuterAPI.java:153) ~[rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.out.BrokerOuterAPI.registerBrokerAll(BrokerOuterAPI.java:193) ~[rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.a.a(BrokerController.java:617) [rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.a$7.run(BrokerController.java:587) [rocketmq-broker-3.2.6.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471) [na:1.7.0_79]
        at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:304) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:178) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615) [na:1.7.0_79]
        at java.lang.Thread.run(Thread.java:745) [na:1.7.0_79]
2015-11-10 13:16:33 WARN BrokerControllerScheduledThread1 - registerBroker Exception, 134.130.134.46:9876
com.alibaba.rocketmq.remoting.exception.RemotingTimeoutException: wait response on the channel <134.130.134.46:9876> timeout, 3000(ms)
        at com.alibaba.rocketmq.remoting.netty.NettyRemotingAbstract.invokeSyncImpl(NettyRemotingAbstract.java:375) ~[rocketmq-remoting-3.2.6.jar:na]
        at com.alibaba.rocketmq.remoting.netty.NettyRemotingClient.invokeSync(NettyRemotingClient.java:622) ~[rocketmq-remoting-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.out.BrokerOuterAPI.registerBroker(BrokerOuterAPI.java:153) ~[rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.out.BrokerOuterAPI.registerBrokerAll(BrokerOuterAPI.java:193) ~[rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.a.a(BrokerController.java:617) [rocketmq-broker-3.2.6.jar:na]
        at com.alibaba.rocketmq.broker.a$7.run(BrokerController.java:587) [rocketmq-broker-3.2.6.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:471) [na:1.7.0_79]
        at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:304) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$301(ScheduledThreadPoolExecutor.java:178) [na:1.7.0_79]
        at java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:293) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145) [na:1.7.0_79]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615) [na:1.7.0_79]
        at java.lang.Thread.run(Thread.java:745) [na:1.7.0_79]
"broker.log" 632L, 74762C
```

解决:当出现这类问题时，检查当前网络并无异常时，并排查下 ulimit –a openfiles 是否为 1024，修改至 65535

2）超时异常 RemotingTimeoutException

现象：当服务器端日志出现 RemotingTimeoutException：wait response on the channel <10.4.246.198:10911> timeout, 3000ms

解决：这类情况一般由于客户端与服务端通信出现问题，可以 ping Ip 以及 telnet ip port 来排查这类，同时也要检查防火墙的问题。

3）找不到路由 No route info of this topic

现象：出现 No route info of this topic,TopicTest。

这个错误可能原因

1. 没创建 topic

2. name server 填错了

3. 网络问题无法获取路由

解决方案：

1. 在管理台创建 topic

2. 检查客户端配置的 namesrv 的地址是否配错了

3. 检查网络是否正常

4）备不可用 SLAVE_NOT_AVAILABLE

现象：当生产者发送消息时，出现“status:SLAVE_NOT_AVAILABLE”，说明从节点发生状况

解决方案：

1、从节点机器出现问题，重启从节点，并查看网络连接

2、在多网卡情况下，broker 配置文件 properties 中，需增加配置项，例：

brokerIP1=10.4.246.130

brokerIP2=10.4.246.130

3、防止网卡 ip 读取错误，取不到从节点信息。

5）消息体大小越界

现象：客户端报此类异常 Fail to send message, for: message body size over max message size, max: 524288

解决方案：

1. 检查服务端的最大消息体大小，即启动 broker 配置文件的 maxMessageSize 大小，如未配置，默认是 512K

2. 检查客户端设置的最大消息体(默认 128k)是否小于当前发送的消息体大小。

注意：ROCKETMQ 建议消息体在 50K 或以下(压缩后)

6）组名已经创建

现象：当消息生产端/消费端运行时，报错 The producer/consumer group has been created

原因：在同一个 jvm 里面只允许一个 producerGroupName 被加载一次（consumerGroupName 同理），否则，就会报错。

解决方案：

1. 如果使用同一个 producerGroupName，部署多个实例（起多个进程）。

2. 在一个进程里，起多个线程，共用一个 Producer 对象实例。

7）Subscription group not exist 或者抛出%retry%的 topic 没有路由信息

原因：没有建立消费关系或者没有创建相关订阅组

解决方案：在管理台或命令行创建对应订阅组

8）Messgae already acked, ackMessage failed

现象如图：

```
2016-03-29 18:50:10,878 ERROR  (TransacUtil.java:187): Thread-19 首次签收消息失败
com.ctg.mq.api.exception.MQAckException: message already acked, ackMessage fail, record not existed: consumeQueueOffset=24
        at com.ctg.mq.api.impl.MQAbstractConsumer.ackMessage(MQAbstractConsumer.java:258)
        at com.ctg.mq.api.impl.MQAbstractConsumer.ackMessage(MQAbstractConsumer.java:420)
        at com.dcs.dispatch.EventListener.billNodeRespHandle(EventListener.java:426)
        at com.dcs.dispatch.EventListener.work(EventListener.java:222)
        at com.dcs.dispatch.EventListener.run(EventListener.java:168)
Caused by: com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 511  DESC: message already acked, ackMessage fail, record not existed: consumeQueueOffset=24
For more information, please visit the url, https://github.com/alibaba/RocketMQ/issues/48
        at com.alibaba.rocketmq.client.impl.MQClientAPIImpl.ackMessage(MQClientAPIImpl.java:741)
        at com.alibaba.rocketmq.client.impl.consumer.PullAPIWrapper.ackMessage(PullAPIWrapper.java:386)
        at com.ctg.mq.api.impl.consumer.LightMQPullConsumerImpl.ackMessage(LightMQPullConsumerImpl.java:515)
        at com.ctg.mq.api.consumer.LightMQPullConsumer.ackMessage(LightMQPullConsumer.java:345)
        at com.ctg.mq.api.impl.MQPullConsumerImpl.ackMessage(MQPullConsumerImpl.java:515)
        at com.ctg.mq.api.impl.MQAbstractConsumer.ackMessage(MQAbstractConsumer.java:248)
        ... 4 more
```

解决方案：这种异常表明该消息已被签收，直接跳过即可

9）重试签收调用次数

现象：ackMessageRetry 重试签收是只需要调一次就够了，还是需要调多次。例如，签收失败后，我调用重试签收，如果重试签收也失败，我需要再次调用重试签收吗？还是它会自动帮我重试签收

解答：现有版本接口不会自动帮你重试签收的， 重试签收失败后，需要自己再次调用重试签收接口。

10）签收时出现不确定异常，如发生超时，或者网络异常时，是需要应用判断消息是否已经签收成功

建议：

1、通过管理台“即时查询”模块，查询这消息是否已经签收成功，看结果再做处理，

2、重试签收，如果已经签收会抛已签收异常。主要还是看应用的自己处理

11）客户端注册失败

现象：

客户端日志报 No matched consumer for the PullRequest PullRequest 如下

```
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=192], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=86], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=1], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=191], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=76], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=150], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=66], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=80], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=117], nextOffset=1008], drop it
2017-09-26 11:01:06,835 WARN RocketmqClient(94) - No matched consumer for the PullRequest PullRequest [consumerGroup=C_DTS_CRM3_ETL_AUDIT, messageQueue=MessageQueue
[topic=DTS_CRM3_ETL, brokerName=crm_second_03, queueId=5], nextOffset=1008], drop it
```

原因：客户端实例注册失败

解决：检查客户端代码，重启客户端进程。

12）the consumer message buffer is full, so do flow control

现象：客户端日志出现 the consumer message buffer is full, so do flow control

原因：push 客户端消费过慢，本地缓存队列已满，暂时停止向服务端拉取消息。消费慢的原因可能是网络原因、topic 队列数过多、消费者过少，内存过小等。

解决：

1. 查看网络是否异常，缓慢

2. 增加消费者实例

3. 如果消息不重要，又不方便增加消费者实例，可以减少 topic 队列数量。

13）system busy, start flow control for a while

现象：客户端日志出现（1）[REJECTREQUEST]system busy, start flow control for a while

或者（2）

[PCBUSY_CLEAN_QUEUE]broker busy, start flow control for a while, period in queue

原因：

（1）在关闭生产者实例的同时用生产者实例发送消息，连接关闭了 netty 会拒绝请求

（2）线程少，处理发送请求过慢

解决：

（1）应用优化使用流程，禁止在 close 生产者实例后使用生产者

（2）如果 Broker 是同步主，那么改成异步主，或者将 sendMessageThreadPoolNums=32 且 waitTimeMillsInSendQueue=1000

9 修订记录

修订记录

发布日期	修改说明
2019-09-06	首次发布分布式消息服务 RocketMQ 用户使用指南