ALIBABA CLOUD

Alibaba Cloud

消息队列Kafka版

最佳实践

文档版本：20220706

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.发布者最佳实践

本文介绍消息队列Kafka版发布者的最佳实践，帮助您降低发送消息的错误率。本文最佳实践基于Java客户端。对于其他语言的客户端，其基本概念与思想是相通的，但实现细节可能存在差异。

发送消息

发送消息的示例代码如下：

```
Future<RecordMetadata> metadataFuture = producer.send(new ProducerRecord<String, String>(
        topic,   //消息主题。
        null,   //分区编号。建议为null，由Producer分配。
        System.currentTimeMillis(),   //时间戳。
        String.valueOf(value.hashCode()),   //消息键。
        value   //消息值。
));
```

完整示例代码，请参见SDK概述。

Key和Value

0.10.2.2版本的消息队列Kafka版的消息有以下两个字段：

- Key：消息的标识。
- Value：消息内容。

为了便于追踪，请为消息设置一个唯一的Key。您可以通过Key追踪某消息，打印发送日志和消费日志，了解该消息的发送和消费情况。

如果消息发送量较大，建议不要设置Key，并使用黏性分区策略。黏性分区策略详情，请参见黏性分区策略。

> **注意** 在0.11.0以及之后的版本，消息队列Kafka版开始支持headers，如果您需要使用headers，需要将服务端升级至2.2.0版本。

失败重试

分布式环境下，由于网络等原因偶尔发送失败是常见的。导致这种失败的原因可能是消息已经发送成功，但是ACK失败，也有可能是确实没发送成功。

消息队列Kafka版是VIP网络架构，30秒没有活动就会主动断开空闲连接，因此，不是一直活跃的客户端会经常收到 `connection reset by peer` 错误，建议重试消息发送。

您可以根据业务需求，设置以下重试参数：

- `retries`：重试次数，建议设置为 `3`。
- `retry.backoff.ms`，重试间隔，建议设置为 `1000`。

异步发送

发送接口是异步的，如果您想接收发送的结果，可以调用metadataFuture.get(timeout, TimeUnit.MILLISECONDS)。

线程安全

Producer是线程安全的，且可以往任何Topic发送消息。通常情况下，一个应用对应一个Producer。

Acks

Acks的说明如下：

- `acks=0`：无需服务端的Response、性能较高、丢数据风险较大。
- `acks=1`：服务端主节点写成功即返回Response、性能中等、丢数据风险中等、主节点宕机可能导致数据丢失。
- `acks=all`：服务端主节点写成功且备节点同步成功才返回Response、性能较差、数据较为安全、主节点和备节点都宕机才会导致数据丢失。

一般建议选择`acks=1`，重要的服务可以设置`acks=all`。

提升发送性能（减少碎片化发送请求）

一般情况下，一个消息队列Kafka版Topic会有多个分区。消息队列Kafka版Producer客户端在向服务端发送消息时，需要先确认往哪个Topic的哪个分区发送。我们给同一个分区发送多条消息时，Producer客户端将相关消息打包成一个Batch，批量发送到服务端。Producer客户端在处理Batch时，是有额外开销的。一般情况下，小Batch会导致Producer客户端产生大量请求，造成请求队列在客户端和服务端的排队，并造成相关机器的CPU升高，从而整体推高了消息发送和消费延迟。一个合适的Batch大小，可以减少发送消息时客户端向服务端发起的请求次数，在整体上提高消息发送的吞吐和延迟。

Batch机制，消息队列Kafka版Producer端主要通过两个参数进行控制：

- `batch.size`：发往每个分区（Partition）的消息缓存量（消息内容的字节数之和，不是条数）。达到设置的数值时，就会触发一次网络请求，然后Producer客户端把消息批量发往服务器。如果`batch.size`设置过小，有可能会影响稳定性问题。
- `linger.ms`：每条消息在缓存中的最长时间。若超过这个时间，Producer客户端就会忽略`batch.size`的限制，立即把消息发往服务器。建议根据业务场景，设置`linger.ms`在100~1000之间。

因此，消息队列Kafka版Producer客户端什么时候把消息批量发送至服务器是由`batch.size`和`linger.ms`共同决定的。您可以根据具体业务需求进行调整。为了提升发送的性能，保障服务的稳定性，建议您设置`batch.size=16384`和`linger.ms=1000`。

黏性分区策略

只有发送到相同分区的消息，才会被放到同一个Batch中，因此决定一个Batch如何形成的一个因素是消息队列Kafka版Producer端设置的分区策略。消息队列Kafka版Producer允许通过设置Partitioner的实现类来选择适合自己业务的分区。在消息指定Key的情况下，消息队列Kafka版Producer的默认策略是对消息的Key进行哈希，然后根据哈希结果选择分区，保证相同Key的消息会发送到同一个分区。

在消息没有指定Key的情况下，消息队列Kafka版2.4版本之前的默认策略是循环使用主题的所有分区，将消息以轮询的方式发送到每一个分区上。但是，这种默认策略Batch的效果会比较差，在实际使用中，可能会产生大量的小Batch，从而使得实际的延迟增加。鉴于该默认策略对无Key消息的分区效率低问题，消息队列Kafka版在2.4版本引入了黏性分区策略（Sticky Partitioning Strategy）。

黏性分区策略主要解决无Key消息分散到不同分区，造成小Batch问题。其主要策略是如果一个分区的Batch完成后，就随机选择另一个分区，然后后续的消息尽可能地使用该分区。这种策略在短时间内看，会将消息发送到同一个分区，如果拉长整个运行时间，消息还是可以均匀地发布到各个分区上的。这样可以避免消息出现分区倾斜，同时还可以降低延迟，提升服务整体性能。

如果您使用的消息队列Kafka版Producer客户端是2.4及以上版本，默认的分区策略就采用黏性分区策略。如果您使用的Producer客户端版本小于2.4，可以根据黏性分区策略原理，自行实现分区策略，然后通过参数`partitioner.class`设置指定的分区策略。

关于黏性分区策略实现，您可以参考如下Java版代码实现。该代码的实现逻辑主要是根据一定的时间间隔，切换一次分区。

```
public class MyStickyPartitioner implements Partitioner {
    // 记录上一次切换分区时间。
    private long lastPartitionChangeTimeMillis = 0L;
    // 记录当前分区。
    private int currentPartition = -1;
    // 分区切换时间间隔，可以根据实际业务选择切换分区的时间间隔。
    private long partitionChangeTimeGap = 100L;
    public void configure(Map<String, ?> configs) {}
    /**
     * Compute the partition for the given record.
     *
     * @param topic The topic name
     * @param key The key to partition on (or null if no key)
     * @param keyBytes serialized key to partition on (or null if no key)
     * @param value The value to partition on or null
     * @param valueBytes serialized value to partition on or null
     * @param cluster The current cluster metadata
     */
    public int partition(String topic, Object key, byte[] keyBytes, Object value, byte[] va
lueBytes, Cluster cluster) {
        // 获取所有分区信息。
        List<PartitionInfo> partitions = cluster.partitionsForTopic(topic);
        int numPartitions = partitions.size();
        if (keyBytes == null) {
            List<PartitionInfo> availablePartitions = cluster.availablePartitionsForTopic(t
opic);
            int availablePartitionSize = availablePartitions.size();
            // 判断当前可用分区。
            if (availablePartitionSize > 0) {
                handlePartitionChange(availablePartitionSize);
                return availablePartitions.get(currentPartition).partition();
            } else {
                handlePartitionChange(numPartitions);
                return currentPartition;
            }
        } else {
            // 对于有key的消息，根据key的哈希值选择分区。
            return Utils.toPositive(Utils.murmur2(keyBytes)) % numPartitions;
        }
    }
    private void handlePartitionChange(int partitionNum) {
        long currentTimeMillis = System.currentTimeMillis();
        // 如果超过分区切换时间间隔，则切换下一个分区，否则还是选择之前的分区。
        if (currentTimeMillis - lastPartitionChangeTimeMillis >= partitionChangeTimeGap
            || currentPartition < 0 || currentPartition >= partitionNum) {
            lastPartitionChangeTimeMillis = currentTimeMillis;
            currentPartition = Utils.toPositive(ThreadLocalRandom.current().nextInt()) % pa
rtitionNum;
        }
    }
    public void close() {}
}
```

OOM

结合消息队列Kafka版的Batch设计思路，消息队列Kafka版会缓存消息并打包发送，如果缓存太多，则有可能造成OOM（Out of Memory）。

- `buffer.memory`：所有缓存消息的总体大小超过这个数值后，就会触发把消息发往服务器。此时会忽略`batch.size`和`linger.ms`的限制。
- `buffer.memory`的默认数值是32 MB，对于单个Producer而言，可以保证足够的性能。

  > **注意** 如果您在同一个JVM中启动多个Producer，那么每个Producer都有可能占用32 MB缓存空间，此时便有可能触发OOM。

- 在生产时，一般没有必要启动多个Producer；如有特殊情况需要，则需要考虑`buffer.memory`的大小，避免触发OOM。

分区顺序

单个分区（Partition）内，消息是按照发送顺序储存的，是基本有序的。

默认情况下，消息队列Kafka版为了提升可用性，并不保证单个分区内绝对有序，在升级或者宕机时，会发生少量消息乱序（某个分区挂掉后把消息Failover到其它分区）。

对于包年包月计费模式下的专业版实例，如果业务要求分区保证严格有序，请在创建Topic时选择使用Local存储。

2.订阅者最佳实践

本文主要介绍消息队列Kafka版订阅者的最佳实践，帮助您减少消费消息出错的可能性。

消费消息基本流程

消息队列Kafka版订阅者在订阅消息时的基本流程是：

1. Poll数据。
2. 执行消费逻辑。
3. 再次poll数据。

负载均衡

每个Group可以包含多个消费实例，即可以启动多个消息队列Kafka版Consumer，并把参数`group.id`设置成相同的值。属于同一个Group的消费实例会负载消费订阅的Topic。

例如Group A订阅了Topic A，并开启三个消费实例C1、C2、C3，则发送到Topic A的每条消息最终只会传给C1、C2、C3的某一个。消息队列Kafka版默认会均匀地把消息传给各个消息实例，以做到消费负载均衡。

消息队列Kafka版负载均衡消费的内部原理是，把订阅的Topic的分区，平均分配给各个消费实例。因此，消费实例的个数不要大于分区的数量，否则会有消费实例分配不到任何分区而处于空跑状态。这个负载均衡发生的时间，除了第一次启动上线之外，后续消费实例发生重启、增加、减少等变更时，都会触发一次负载均衡。

消费客户端（Consumer）频繁出现Rebalance

心跳超时会引发Rebalance，可以通过参数调整、提高消费速度等方法解决。更多信息，请参见使用消息队列Kafka版时消费客户端频繁出现Rebalance。

分区个数

分区个数主要影响的是消费者的并发数量。

对于同一个Group内的消费者来说，一个分区最多只能被一个消费者消费。因此，消费实例的个数不要大于分区的数量，否则会有消费实例分配不到任何分区而处于空跑状态。

控制台的默认分区个数是12，可以满足绝大部分场景的需求。您可以根据业务使用量进行增加。不建议分区数小于12，否则可能影响消费发送性能；也不建议超过100个，否则易引发消费端Rebalance。

> **注意** 分区增加后，将不能减少，请小幅度调整。

多个订阅

消息队列Kafka版支持以下多个订阅方式：

- Group订阅多个Topic。

  一个Group可以订阅多个Topic，多个Topic的消息被Group中的Consumer均匀消费。例如Group A订阅了Topic A、Topic B、Topic C，则这三个Topic中的消息，被Group中的Consumer均匀消费。

  Group订阅多个Topic的示例代码如下：

```
String topicStr = kafkaProperties.getProperty("topic");
String[] topics = topicStr.split(",");
for (String topic: topics) {
subscribedTopics.add(topic.trim());
}
consumer.subscribe(subscribedTopics);
```

- Topic被多个Group订阅。

  一个Topic可以被多个Group订阅，且各个Group独立消费Topic下的所有消息。例如Group A订阅了Topic A，Group B也订阅了Topic A，则发送到Topic A的每条消息，不仅会传一份给Group A的消费实例，也会传一份给Group B的消费实例，且这两个过程相互独立，相互没有任何影响。

一个Group对应一个应用

建议一个Group对应一个应用，即不同的应用对应不同的代码。如果您需要将不同的代码写在同一个应用中，请准备多份不同的kafka.properties。例如kafka1.properties、kafka2.properties。

消费位点

每个Topic会有多个分区，每个分区会统计当前消息的总条数，这个称为最大位点MaxOffset。

消息队列Kafka版Consumer会按顺序依次消费分区内的每条消息，记录已经消费了的消息条数，称为消费位点ConsumerOffset。

剩余的未消费的条数（也称为消息堆积量）=MaxOffset-ConsumerOffset。

消费位点提交

消息队列Kafka版消费者有两个相关参数：

- enable.auto.commit：是否采用自动提交位点机制。默认值为true，表示默认采用自动提交机制。
- auto.commit.interval.ms： 自动提交位点时间间隔。默认值为1000，即1s。

这两个参数组合的结果就是，每次poll数据前会先检查上次提交位点的时间，如果距离当前时间已经超过参数auto.commit.interval.ms规定的时长，则客户端会启动位点提交动作。

因此，如果将enable.auto.commit设置为true，则需要在每次poll数据时，确保前一次poll出来的数据已经消费完毕，否则可能导致位点跳跃。

如果想自己控制位点提交，请把enable.auto.commit设为false，并调用commit(offsets)函数自行控制位点提交。

消费位点重置

以下两种情况，会发生消费位点重置：

- 当服务端不存在曾经提交过的位点时（例如客户端第一次上线）。
- 当从非法位点拉取消息时（例如某个分区最大位点是10，但客户端却从11开始拉取消息）。

Java客户端可以通过auto.offset.reset来配置重置策略，主要有三种策略：

- latest：从最大位点开始消费。
- earliest：从最小位点开始消费。
- none：不做任何操作，即不重置。

> 说明
> - 建议设置成latest，而不要设置成earliest，避免因位点非法时从头开始消费，从而造成大量重复。
> - 如果是您自己管理位点，可以设置成none。

拉取大消息

消费过程是由客户端主动去服务端拉取消息的，在拉取大消息时，需要注意控制拉取速度，注意修改配置：

- max.poll.records：如果单条消息超过1 MB，建议设置为1。
- fetch.max.bytes：设置比单条消息的大小略大一点。
- max.partition.fetch.bytes：设置比单条消息的大小略大一点。

拉取大消息的核心是逐条拉取的。

消息重复和消费幂等

消息队列Kafka版消费的语义是at least once， 也就是至少投递一次，保证消息不丢失，但是无法保证消息不重复。在出现网络问题、客户端重启时均有可能造成少量重复消息，此时应用消费端如果对消息重复比较敏感（例如订单交易类），则应该做消息幂等。

以数据库类应用为例，常用做法是：

- 发送消息时，传入key作为唯一流水号ID。
- 消费消息时，判断key是否已经消费过，如果已经被消费，则忽略，如果没消费过，则消费一次。

如果应用本身对少量消息重复不敏感，则不需要做此类幂等检查。

消费失败

消息队列Kafka版是按分区逐条消息顺序向前推进消费的，如果消费端拿到某条消息后执行消费逻辑失败，例如应用服务器出现了脏数据，导致某条消息处理失败，等待人工干预，那么有以下两种处理方式：

- 失败后一直尝试再次执行消费逻辑。这种方式有可能造成消费线程阻塞在当前消息，无法向前推进，造成消息堆积。
- 消息队列Kafka版没有处理失败消息的设计，实践中通常会打印失败的消息或者存储到某个服务（例如创建一个Topic专门用来放失败的消息），然后定时检查失败消息的情况，分析失败原因，根据情况处理。

消费延迟

消息队列Kafka版的消费机制是由客户端主动去服务端拉取消息进行消费的。因此，如果客户端能够及时消费，则不会产生较大延迟。如果产生了较大延迟，请先关注是否有堆积，并注意提高消费速度。

消费阻塞以及堆积

消费端最常见的问题就是消费堆积，最常造成堆积的原因是：

- 消费速度跟不上生产速度，此时应该提高消费速度，详情请参见提高消费速度。
- 消费端产生了阻塞。

消费端拿到消息后，执行消费逻辑，通常会执行一些远程调用，如果这个时候同步等待结果，则有可能造成一直等待，消费进程无法向前推进。

消费端应该竭力避免堵塞消费线程，如果存在等待调用结果的情况，建议设置等待的超时时间，超时后作为消费失败进行处理。

提高消费速度

提高消费速度有以下两个办法：

- 增加Consumer实例个数。

  可以在进程内直接增加（需要保证每个实例对应一个线程，否则没有太大意义），也可以部署多个消费实例进程；需要注意的是，实例个数超过分区数量后就不再能提高速度，将会有消费实例不工作。

- 增加消费线程。

  增加Consumer实例本质上也是增加线程的方式来提升速度，因此更加重要的性能提升方式是增加消费线程，最基本的步骤如下：

  i. 定义一个线程池。
  ii. Poll数据。
  iii. 把数据提交到线程池进行并发处理。
  iv. 等并发结果返回成功后，再次poll数据执行。

消息过滤

消息队列Kafka版自身没有消息过滤的语义。实践中可以采取以下两个办法：

- 如果过滤的种类不多，可以采取多个Topic的方式达到过滤的目的。
- 如果过滤的种类多，则最好在客户端业务层面自行过滤。

实践中请根据业务具体情况进行选择，也可以综合运用上面两种办法。

消息广播

消息队列Kafka版没有消息广播的语义，可以通过创建不同的Group来模拟实现。

订阅关系

同一个Group内，各个消费实例订阅的Topic最好保持一致，避免给排查问题带来干扰。

3.自建Kafka集群迁移上云

3.1. 迁移概述

本文介绍将自建Kafka集群迁移到消息队列Kafka版实例的优势、原理、方案架构、迁移工具和操作流程。

迁移优势

将自建Kafka集群迁移到消息队列Kafka版实例的优势，请参见产品优势。

迁移原理

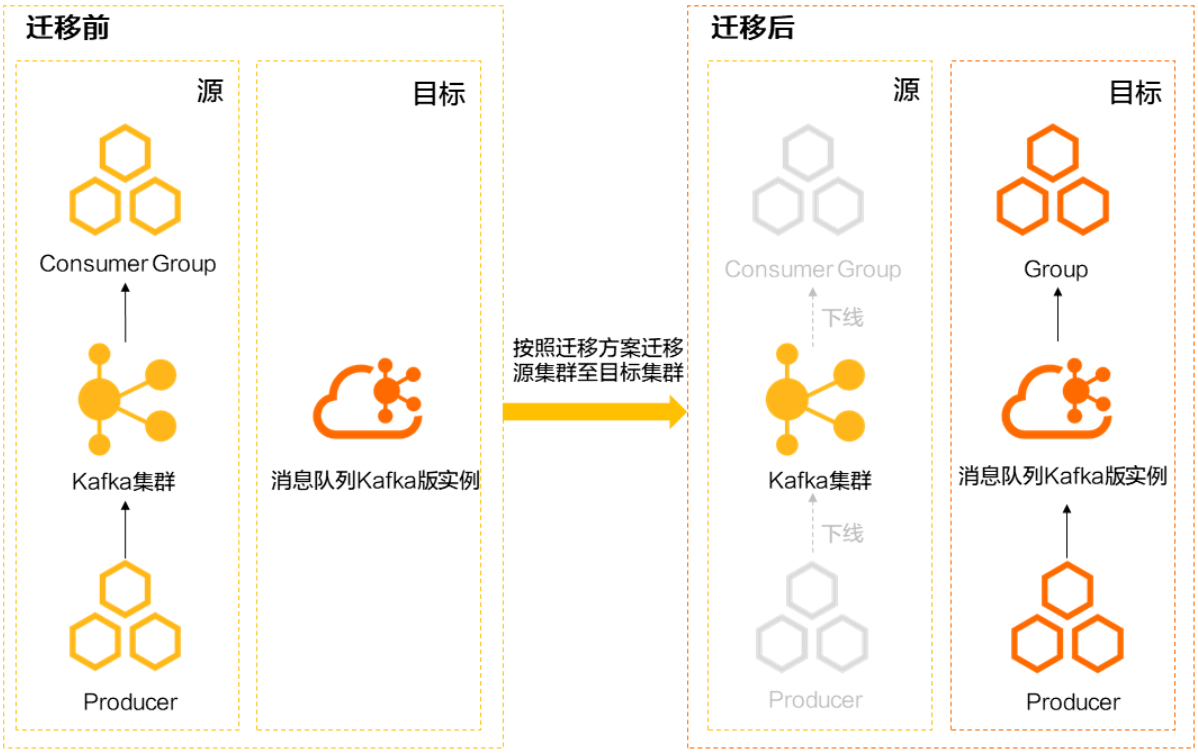
对于消息队列来说，如果要实现集群迁移，只需消费完旧集群的消息即可。由于Producer和Consumer都是集群化的，您可以通过一台一台机器操作的方式实现上层业务无感知。

迁移说明

- 迁移不会删除自建的源Kafka集群的Topic和Group，只是在目标消息队列Kafka版实例中创建相同配置的Topic和Group。
- 迁移内容仅为Topic和Group 配置，不包含Topic中存储的消息以及Group的订阅关系和消费位点信息。

方案架构

迁移自建Kafka集群至消息队列Kafka版实例方案如下图所示。



方案说明如下：

1. 使用迁移工具迁移自建Kafka集群至消息队列Kafka版实例。
2. 为消息队列Kafka版实例开启新的Group，准备消费消息队列Kafka版实例的消息。
3. 为消息队列Kafka版实例开启新的Producer，下线旧的Producer，并使旧的Group继续消费自建Kafka集

群的消息。

4. 待自建Kafka集群的消息全部被旧的Group消费后，下线旧的Group和自建Kafka集群。

迁移工具

迁移工具	说明	参考文档
kafka-migration-assessment.jar.zip	• 您可以使用该工具将自建Kafka集群元数据通过控制台任务导出为JSON格式文件，然后上传该文件至消息队列Kafka版迁移任务并运行任务，实现元数据迁移。 • 您可以使用该迁移工具将自建Kafka集群的Topic和Group迁移到消息队列Kafka版实例。	• 迁移自建Kafka集群元数据至已有实例 • 迁移自建Kafka集群元数据至新实例 • 迁移Topic上云 • 迁移Group上云
MirrorMaker	您可以使用该工具将源自建Kafka集群中的数据镜像拷贝到目标消息队列Kafka版集群。	迁移数据上云

迁移操作流程

流程说明如下：

1. （可选）评估规格：如果您已了解待迁移自建Kafka集群的信息，如集群流量情况、磁盘容量和类型、分区数量等信息，您可以直接评估需要购买的消息队列Kafka版实例规格。

   如果您不清楚自建Kafka集群的相关信息，您无需执行该操作，在迁移自建Kafka集群元数据至新实例时，使用迁移工具进行自动评估。

2. 实施迁移：您可以通过执行命令实现单资源迁移，也可以通过迁移任务实现迁移。
   - 迁移自建Kafka集群至新实例：在消息队列Kafka版控制台，创建迁移任务，根据推荐意见购买消息队列Kafka版实例并部署启动任务。

     > **说明** 如果元数据文件中包含了自建Kafka集群的流量、磁盘、配置等信息，购买实例时，消息队列Kafka版会根据元数据文件信息，自动评估并推荐实例规格，您可以根据推荐意见购买实例。

   - 迁移自建Kafka集群元数据至已有实例：在消息队列Kafka版控制台，通过创建迁移任务，将自建Kafka集群元数据迁移至消息队列Kafka版已有实例。
   - 单资源迁移：如果您只需要迁移Topic、Group或消息数据，您可以选择对应资源迁移方法迁移。
     a. 迁移Topic上云：迁移自建Kafka集群的Topic到消息队列Kafka版实例。
     b. 迁移Group上云：迁移自建Kafka集群的Group到消息队列Kafka版实例。
     c. （可选）迁移数据上云：迁移自建Kafka集群的数据到消息队列Kafka版实例。

     > **注意** 消息队列的特点是，数据一旦被消费，则已经完成使命。因此，除了需要将自建Kafka集群的数据备份消息队列Kafka版实例的情况外，一般情况下不推荐您迁移数据。

3. 查看迁移进度：查看自建Kafka集群迁移到消息队列Kafka版的进度。
4. 验证迁移结果：查看迁移成功后的资源列表。

3.2. 评估规格

本文说明迁移前如何使用消息队列Kafka版的规格评估功能为迁移上云的自建Kafka集群选择合适的实例规格。

前提条件

注册阿里云账号并完成实名认证。详细信息，请参见注册阿里云账号。

使用说明

消息队列Kafka版提供的规格评估功能将根据您输入的自建Kafka集群的信息，如集群流量情况、磁盘容量和类型、分区数量等，来为您评估并推荐需要的消息队列Kafka版实例规格，帮助您提前根据推荐部署消息队列Kafka版实例。在迁移时，您可直接选择已部署的目标实例进行迁移。

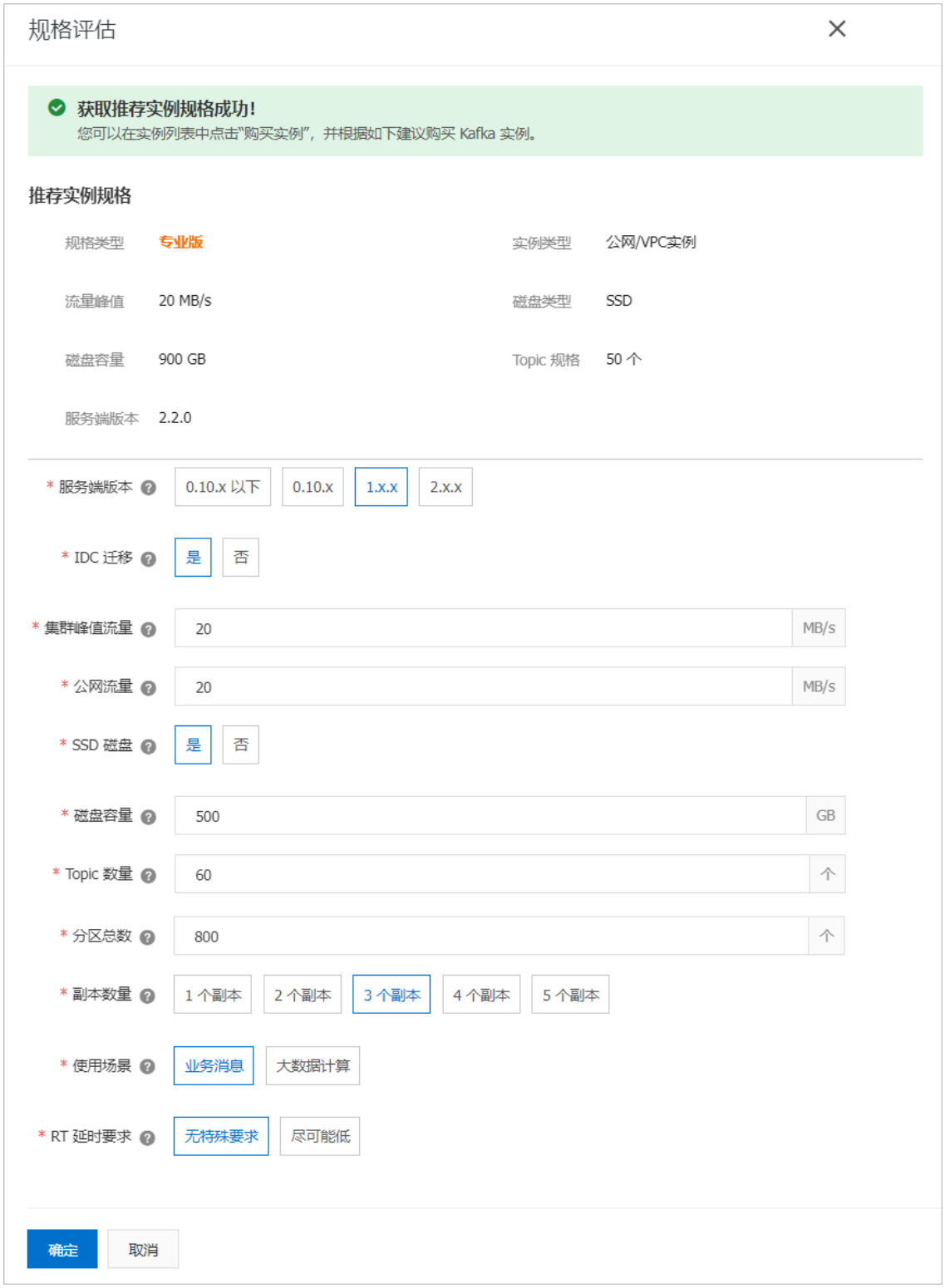
如果您不清楚待迁移自建Kafka集群相关的信息，您可以使用迁移工具评估，在迁移任务中评估，并根据推荐购买新的消息队列Kafka版实例以完成迁移。具体操作，请参见迁移自建Kafka集群至新实例。

评估规格

> **说明** 消息队列Kafka版的规格评估功能遵循最小原则，即在满足业务正常需求的基础上，尽量推荐价格较低的消息队列Kafka版实例。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在左侧导航栏，单击**迁移上云**。
4. 在**迁移上云**页面，单击**规格评估**。
5. 在**规格评估**面板，输入自建Kafka集群的规格信息，然后单击**确定**获取规格建议。

   在**获取推荐实例规格成功！**之后，在**规格评估**面板显示系统为您推荐的实例规格。



如需了解消息队列Kafka版实例规格详情，请参见实例规格。

3.3. 迁移自建Kafka集群至已有实例

3.3.1. 单资源迁移

3.3.1.1. 迁移Topic上云

本文介绍如何使用消息队列Kafka版提供的迁移工具将自建Kafka集群的Topic迁移到消息队列Kafka版实例。

前提条件

您已完成以下操作：

- 下载JDK 8
- 下载迁移工具kafka-migration-assessment.jar.zip
- 购买并部署消息队列Kafka版实例：
  - 购买并部署VPC实例
  - 购买并部署公网/VPC实例

操作步骤

1. 打开命令行工具。
2. 使用cd命令将路径切换到迁移工具所在目录。
3. 执行以下命令确认要迁移的Topic。

```
java -jar kafka-migration-assessment.jar TopicMigrationFromZk --sourceZkConnect 192.168.XX.XX --destAk <yourdestAccessKeyId> --destSk <yourdestAccessKeySecret> --destRegionId <yourdestRegionId> --destInstanceId <yourdestInstanceId>
```

参数	描述
sourceZkConnect	自建的源ZooKeeper集群的IP地址
destAk	目标消息队列Kafka版实例所属阿里云账号的AccessKey ID
destSk	目标消息队列Kafka版实例所属阿里云账号的AccessKey Secret
destRegionId	目标消息队列Kafka版实例的地域ID
destInstanceId	目标消息队列Kafka版实例的ID

待确认的返回结果示例如下：

```
13:40:08 INFO - Begin to migrate topics:[test]
13:40:08 INFO - Total topic number:1
13:40:08 INFO - Will create topic:test, isCompactTopic:false, partition number:1
```

4. 执行以下命令提交要迁移的Topic。

```
java -jar kafka-migration-assessment.jar TopicMigrationFromZk --sourceZkConnect 192.168.XX.XX --destAk <yourAccessKeyId> --destSk <yourAccessKeySecret> --destRegionId <yourRegionID> --destInstanceId <yourInstanceId> --commit
```

参数	描述
commit	提交迁移

提交迁移的返回结果示例如下：

```
13:51:12 INFO - Begin to migrate topics:[test]
13:51:12 INFO - Total topic number:1
13:51:13 INFO - cmd=TopicMigrationFromZk, request=null, response={"code":200,"requestId
":"7F76C7D7-AAB5-4E29-B49B-CD6F1E0F508B","success":true,"message":"operation success"}
13:51:13 INFO - TopicCreate success, topic=test, partition number=1, isCompactTopic=fal
se
```

5. 确认Topic迁移是否成功。
   i. 登录消息队列Kafka版控制台。
   ii. 在**概览**页面的**资源分布**区域，选择地域。
   iii. 在**实例列表**页面，单击目标实例名称。
   iv. 在左侧导航栏，单击**Topic 管理**。
   v. 在**Topic 管理**页面的Topic列表中，显示成功迁移的Topic。

3.3.1.2. 迁移Group上云

本文介绍如何使用消息队列Kafka版提供的迁移工具将自建Kafka集群的Group迁移到消息队列Kafka版实例。

前提条件

您已完成以下操作：

- 下载JDK 8
- 下载迁移工具kafka-migration-assessment.jar.zip
- 购买并部署消息队列Kafka版实例：
  - 购买并部署VPC实例
  - 购买并部署公网/VPC实例

操作步骤

1. 打开命令行工具。
2. 使用**cd**命令将路径切换到迁移工具所在目录。
3. 创建配置文件*kafka.properties*。

   *kafka.properties*用于构造Kafka Consumer，从自建Kafka集群获取消费者位点信息。配置文件内容如下：

```
## 接入点。
bootstrap.servers=localhost:9092
## Group ID。注意该Group不能有消费者位点信息，以保证能从第一个消息开始消费。
group.id=XXX
## 如果无安全配置，可以不配置以下内容。
## SASL鉴权方式。
#sasl.mechanism=PLAIN
## 接入协议。
#security.protocol=SASL_SSL
## SSL根证书的路径。
#ssl.truststore.location=/Users/***/Documents/code/aliware-kafka-demos/main/resources/kafka.client.truststore.jks
## SSL密码。
#ssl.truststore.password=***
## SASL路径。
#java.security.auth.login.config=/Users/***/kafka-java-demo/vpc-ssl/src/main/resources/kafka_client_jaas.conf
```

4. 执行以下命令确认要迁移的Group。

```
java -jar kafka-migration-assessment.jar ConsumerGroupMigrationFromTopic --propertiesPath /usr/local/kafka_2.12-2.4.0/config/kafka.properties --destAk <yourAccessKeyId> --destSk <yourAccessKeySecret> --destRegionId <yourRegionId> --destInstanceId <yourInstanceId>
```

参数	描述
propertiesPath	配置文件*kafka.properties*的文件路径
destAk	目标消息队列Kafka版实例所属阿里云账号的AccessKey ID
destSk	目标消息队列Kafka版实例所属阿里云账号的AccessKey Secret
destRegionId	目标消息队列Kafka版实例的地域ID
destInstanceId	目标消息队列Kafka版实例的ID

待确认的返回结果示例如下：

```
15:29:45 INFO - Will create consumer groups:[XXX, test-consumer-group]
```

5. 执行以下命令提交要迁移的Group。

```
java -jar kafka-migration-assessment.jar ConsumerGroupMigrationFromTopic --propertiesPath /usr/local/kafka_2.12-2.4.0/config/kafka.properties --destAk LTAI4FwQ5aK1mFYCspJ1**** --destSk wvDxjjRQ1tHPiL0oj7Y2Z7WDNk**** --destRegionId cn-hangzhou --destInstanceId alikafka_pre-cn-v0h1cng0**** --commit
```

参数	描述
commit	提交迁移

提交迁移的返回结果示例如下：

```
15:35:51 INFO - cmd=ConsumerGroupMigrationFromTopic, request=null, response={"code":200
,"requestId":"C9797848-FD4C-411F-966D-0D4AB5D12F55","success":true,"message":"operation
success"}
15:35:51 INFO - ConsumerCreate success, consumer group=XXX
15:35:57 INFO - cmd=ConsumerGroupMigrationFromTopic, request=null, response={"code":200
,"requestId":"3BCFDBF2-3CD9-4D48-92C3-385C8DBB9709","success":true,"message":"operation
success"}
15:35:57 INFO - ConsumerCreate success, consumer group=test-consumer-group
```

提交迁移后，在消息队列Kafka版控制台的**迁移上云**页面任务列表中，您可以看到实例ID为目标消息队列Kafka版实例ID的任务。

相关文档

- 查看迁移进度
- 验证迁移结果

3.3.1.3. 迁移数据上云

本文介绍如何使用MirrorMaker将自建Kafka集群的数据迁移到消息队列Kafka版集群。

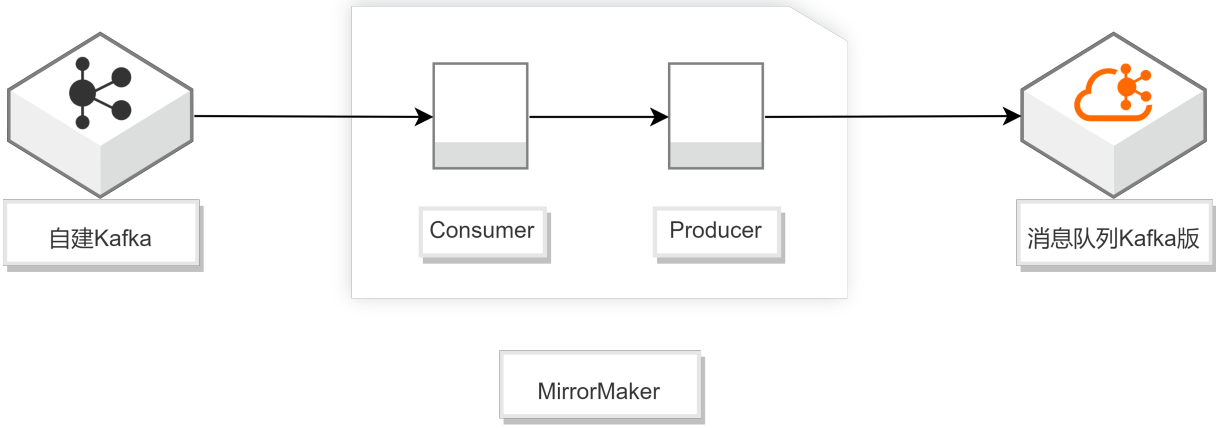
前提条件

您已完成以下操作：

- 下载迁移工具MirrorMaker
- 迁移Topic上云

背景信息

Kafka的镜像特性可实现Kafka集群的数据备份。实现这一特性的工具就是MirrorMaker。您可以使用MirrorMaker将源集群中的数据镜像拷贝到目标集群。如下图所示，Mirror Maker使用一个内置的Consumer从源自建Kafka集群消费消息，然后再使用一个内置的Producer将这些消息重新发送到目标消息队列Kafka版集群。



更多信息，请参见Apache Kafka MirrorMaker。

注意事项

- Topic名称必须一致。
- 分区数量可以不一致。

- 在同一个分区中的数据迁移后并不保证依旧在同一个分区中。
- 默认情况下，Key相同的消息会分布在同一分区中。
- 普通消息在宕机时可能会乱序，分区顺序消息在宕机时依然保持顺序。

VPC接入

1. 配置*consumer.properties*。

```
## 自建Kafka集群的接入点
bootstrap.servers=XXX.XXX.XXX.XXX:9092
## 消费者分区分配策略
partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor
## Group的名称
group.id=test-consumer-group
```

2. 配置*producer.properties*。

```
## 消息队列Kafka版集群的默认接入点（可在消息队列Kafka版控制台获取）
bootstrap.servers=XXX.XXX.XXX.XXX:9092
## 数据压缩方式
compression.type=none
```

3. 执行以下命令开启迁移进程。

```
sh bin/kafka-mirror-maker.sh --consumer.config config/consumer.properties --producer.co
nfig config/producer.properties --whitelist topicName
```

公网接入

1. 下载kafka.client.truststore.jks。
2. 配置*kafka_client_jaas.conf*。

```
KafkaClient {
   org.apache.kafka.common.security.plain.PlainLoginModule required
   username="your username"
   password="your password";
};
```

3. 配置*consumer.properties*。

```
## 自建Kafka集群的接入点
bootstrap.servers=XXX.XXX.XXX.XXX:9092
## 消费者分区分配策略
partition.assignment.strategy=org.apache.kafka.clients.consumer.RoundRobinAssignor
## Group名称
group.id=test-consumer-group
```

4. 配置*producer.properties*。

```
## 消息队列Kafka版集群的SSL接入点（可在消息队列Kafka版控制台获取）
bootstrap.servers=XXX.XXX.XXX.XXX:9093
## 数据压缩方式
compression.type=none
## truststore（使用步骤1下载的文件）
ssl.truststore.location=kafka.client.truststore.jks
ssl.truststore.password=KafkaOnsClient
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
## 消息队列Kafka版2.X版本在配置SASL接入时需要做以下配置，2.X以下版本不需要配置。
ssl.endpoint.identification.algorithm=
```

5. 设置*java.security.auth.login.config*。

```
export KAFKA_OPTS="-Djava.security.auth.login.config=kafka_client_jaas.conf"
```

6. 执行以下命令开启迁移进程。

```
sh bin/kafka-mirror-maker.sh --consumer.config config/consumer.properties --producer.co
nfig config/producer.properties --whitelist topicName
```

结果验证

您可通过以下任一方法验证MirrorMaker是否运行成功。

- 通过 `kafka-consumer-groups.sh` 查看自建集群消费进度。

  ```
  bin/kafka-consumer-groups.sh --new-consumer --describe --bootstrap-server自建集群接入点 --gr
  oup test-consumer-group
  ```

- 往自建集群中发送消息，在消息队列Kafka版控制台中查看Topic的分区状态，确认当前服务器上消息总量是否正确。您还可以通过消息队列Kafka版控制台来查看具体消息内容。具体操作，请参见查询消息。

3.3.2. 元数据迁移

3.3.2.1. 通过元数据文件迁移上云

本文介绍如何在消息队列Kafka版控制台创建迁移任务，将元数据迁移至消息队列Kafka版已有实例。

前提条件

- 下载JDK 8
- 购买并部署消息队列Kafka版实例：
  - 购买并部署VPC实例
  - 购买并部署公网/VPC实例

背景信息

Kafka集群元数据是指Kafka集群的Topic和Group配置信息。Kafka集群元数据存储于ZooKeeper上，Kafka集群各个节点从ZooKeeper中获取最新的元数据。因此，集群的各个节点的元数据被导出时都是最新且相同的。Kafka集群元数据可以被导出成一份JSON文件，然后被导入另一个Kafka集群，实现自建Kafka集群元数据备份。

元数据迁移是指将自建Kafka集群的元数据迁移到阿里云消息队列Kafka版实例。您可以将自建Kafka集群元数据导出，然后导入消息队列Kafka版实例，消息队列Kafka版会根据成功导入的元数据在目标消息队列Kafka版实例中创建对应的Topic和Group，实现自建Kafka集群元数据迁移上云。

导出元数据

使用元数据导出工具导出自建Kafka集群的元数据文件。

1. 访问kafka-migration-assessment.jar.zip地址下载元数据导出工具。
2. 将下载好的迁移工具上传至自建Kafka集群。
3. 在工具所在目录运行以下命令，解压工具包和赋予JAR包可执行权限。

```
unzip -o kafka-migration-assessment.jar.zip
```

```
chmod 777 kafka-migration-assessment.jar
```

4. 运行以下命令，导出元数据。

```
java -jar kafka-migration-assessment.jar MigrationFromZk --sourceZkConnect <host:port>
--sourceBootstrapServers <host:port> --targetDirectory ../xxx/ --fileName metadata.json
--commit
```

参数	说明	示例
sourceZkConnect	自建Kafka集群的ZooKeeper IP地址和端口号。如果不指定，则自动获取。	192.168.XX.XX:2181
sourceBootstrapServers	自建Kafka集群的IP地址和端口号。如果不指定，则自动获取。	192.168.XX.XX:9092
targetDirectory	导出元数据文件的存放目录。如果不指定，则默认为当前目录。	../home/
fileName	导出元数据的文件名。如果不指定，则文件名称默认为： kafka-metadata-export.json。	metadata.json
commit	提交运行。	commit
installTsar	是否自动安装Tsar。默认不自动安装Tsar。 安装Tsar可以更加准确的获取当前机器规格、近期存储情况、流量、Kafka集群配置信息，但自动安装比较耗时，也有可能因为环境差异， 导致安装失败。	无
evaluate	是否获取当前机器规格、近期存储情况、流量、Kafka集群配置信息，在迁移时根据获取的信息进行评估，推荐您购买规格更合适的消息队列Kafka版实例。 默认为true，即默认获取以进行规格评估，如果不需要评估，则需设置为false。	无

文件存放目录下生成JSON文件，导出元数据成功。

在配置的文件存放目录下可以看到导出元数据文件，下载保存至本地。

创建迁移任务并迁移元数据

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在左侧导航栏，单击**迁移上云**。
4. 在**迁移上云**页面，单击**创建任务**。
5. 在**创建任务**配置向导页面，完成以下操作并单击**下一步**。
   i. 在**描述**文本框输入迁移上云的任务名称。
   ii. 在**元数据**右侧，单击**点击上传元数据文件**，选择已提前准备好的JSON格式的元数据文件上传。
6. 在**购买实例**配置向导页面，单击**下一步**。
7. 在**部署实例**配置向导页面，选择已部署的消息队列Kafka版实例，单击**创建任务**。
   在**迁移上云**页面可以看到任务状态为**迁移中**。迁移成功后，即可开始使用迁移上云的Topic和Group。

删除迁移任务

在**迁移上云**页面，找到待删除的目标任务，在其右侧**操作**列，单击**删除**。

后续步骤

查看迁移进度

3.3.2.2. 迁移消费者程序

本文介绍如何将自建Kafka集群上的消费者程序平滑地迁移到消息队列Kafka版。

背景信息

消息队列Kafka版控制台提供迁移实例功能，您可以使用该功能将自建Kafka集群或其他云Kafka集群的元数据（Topic和Group配置信息）和消息数据同步到目标集群，迁移完成后目标集群的元数据与源集群的元数据保持一致并且持续更新。

本文以开源Apache Kafka为例介绍使用实例迁移功能将自建集群上的消费者程序迁移到消息队列Kafka版。

前提条件

- 部署开源Apache Kafka集群
- 购买并部署消息队列Kafka版实例：
  - 购买并部署VPC实例
  - 购买并部署公网/VPC实例

创建迁移任务

1. 登录消息队列Kafka版控制台，在**概览**页面的**资源分布**区域，选择地域。
2. 在左侧导航栏单击**迁移路由**，然后从**实例列表**的下拉列表选择目标实例，单击**创建任务**。
3. 在**创建任务**配置向导页面，完成以下操作。
   i. 在**配置基本信息**页签，设置任务名称，选择**任务类型**为**迁移**，然后单击**下一步**。

ii. 在**配置源服务**页签，配置以下参数，然后单击**下一步**。

参数	说明	示例
接入点	配置源Apache Kafka集群的接入点信息。	192.168.XX.XX:9092
安全协议	选择创建Apache Kafka集群时设置的安全协议。	PLAINTEXT
任务数	选择同步数据的线程并发数。	12
同步sasl用户	迁移数据时是否同步SASL用户。单击**配置运行环境**显示该参数。	是
同步topic acls	迁移数据时是否同步Topic ACLS。单击**配置运行环境**显示该参数。	是
同步消费组	迁移数据时是否同步创建消费组。单击**配置运行环境**显示该参数。	是
同步消费位点	迁移数据时是否同步消费位点。单击**配置运行环境**显示该参数。	是
Topic	源实例的Topic是否需要同步到目标实例，不填写则同步所有Topic。单击**配置运行环境**显示该参数。	test-topic
创建为Local引擎 Topic	迁移到目标实例的非Compact类型的Topic如果想保持为Local引擎，则在此填写。不填写则被创建为云引擎。单击**配置运行环境**显示该参数。	test-topic

iii. 在**配置目标服务**页签，单击**创建**。

4. 创建完成后，在**迁移路由**页面，从**实例列表**的下拉列表选择目标实例，在目标实例列表中找到创建的任务，单击其**操作**列的**部署**。
   在**迁移路由**页面，您可以看到创建的任务**状态**为**运行中**，则说明任务创建成功。您可以在左侧导航栏的**实例列表**页面查看迁移到消息队列Kafka版上的Apache Kafka集群。

在创建的迁移任务右侧**操作**列，您可以查看任务详情、查看同步进度、修改任务参数配置、暂停任务和删除任务。

启动消费者程序

通过实例迁移功能，目标实例将源实例的全量数据追平并开始追加增量后：

- 若您创建消费者程序时使用Subscribe方式消费消息，可以直接在目标实例启动新的消费者程序。新消费者程序启动后，目标实例会阻止迁移组件继续同步消费位点。当确认新消费者程序没有问题后，再在源集群停止旧消费者程序。如果不停止旧消费者程序，那么当新消费者程序全部停止后，迁移组件会继续同步消费位点，产生不可预知的问题，例如覆盖新消费者程序提交的位点。
- 若您创建消费者程序时使用Assign方式消费消息，请先在源集群停止旧消费者程序，旧消费者程序停止

后，迁移组件也会停止同步旧消费者程序所用的消费组的消费位点，然后再在目标实例启动新消费者程序。

- 若您无法确定消费者程序使用哪种消费方式，请先在源集群停止旧消费者程序，旧消费者程序停止后，迁移组件也会停止同步旧消费者程序所用的消费组的消费位点，然后再在目标实例启动新消费者程序。

3.4. 迁移自建Kafka集群至新实例

本文介绍如何在消息队列Kafka版控制台创建并启动迁移任务。

前提条件

- 下载JDK 8
- 安装Tsar

  为了更准确的获取执行机器的配置，推荐您提前安装Tsar软件。您也可以在运行迁移工具时通过指定 `--installTsar` 参数自动安装Tsar软件，但自动安装比较耗时，也有可能因为环境差异， 导致安装失败。

背景信息

Kafka集群元数据是指Kafka集群的Topic和Group配置信息。Kafka集群元数据存储于ZooKeeper上，Kafka集群各个节点从ZooKeeper中获取最新的元数据。因此，集群的各个节点的元数据被导出时都是最新且相同的。Kafka集群元数据可以被导出成一份JSON文件，然后被导入另一个Kafka集群，实现自建Kafka集群元数据备份。

元数据迁移是指将自建Kafka集群的元数据迁移到阿里云消息队列Kafka版实例。您可以将自建Kafka集群元数据导出，然后导入消息队列Kafka版实例，消息队列Kafka版会根据成功导入的元数据在目标消息队列Kafka版实例中创建对应的Topic和Group，实现自建Kafka集群元数据迁移上云。

导出元数据

使用元数据导出工具导出自建Kafka集群的元数据文件。

1. 访问kafka-migration-assessment.jar.zip地址下载元数据导出工具。
2. 将下载好的迁移工具上传至自建Kafka集群。
3. 在工具所在目录运行以下命令，解压工具包和赋予JAR包可执行权限。

   ```
   unzip -o kafka-migration-assessment.jar.zip
   ```

   ```
   chmod 777 kafka-migration-assessment.jar
   ```

4. 运行以下命令，导出元数据。

   ```
   java -jar kafka-migration-assessment.jar MigrationFromZk --sourceZkConnect <host:port>
   --sourceBootstrapServers <host:port> --targetDirectory ../xxx/ --fileName metadata.json
   --commit
   ```

参数	说明	示例
sourceZkConnect	自建Kafka集群的ZooKeeper IP地址和端口号。如果不指定，则自动获取。	192.168.XX.XX:2181
sourceBootstrapServers	自建Kafka集群的IP地址和端口号。如果不指定，则自动获取。	192.168.XX.XX:9092

参数	说明	示例
targetDirectory	导出元数据文件的存放目录。如果不指定，则默认为当前目录。	../home/
fileName	导出元数据的文件名。如果不指定，则文件名称默认为： kafka-metadata-export.json。	metadata.json
commit	提交运行。	commit
installTsar	是否自动安装Tsar。默认不自动安装Tsar。 安装Tsar可以更加准确的获取当前机器规格、近期存储情况、流量、Kafka集群配置信息，但自动安装比较耗时，也有可能因为环境差异， 导致安装失败。	无
evaluate	是否获取当前机器规格、近期存储情况、流量、Kafka集群配置信息，在迁移时根据获取的信息进行评估，推荐您购买规格更合适的消息队列Kafka版实例。 默认为true，即默认获取以进行规格评估，如果不需要评估，则需设置为false。	无

文件存放目录下生成JSON文件，导出元数据成功。

在配置的文件存放目录下可以看到导出元数据文件，下载保存至本地。

步骤一：创建迁移任务

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在左侧导航栏，单击**迁移上云**。
4. 在**迁移上云**页面，单击**创建任务**。
5. 在**创建任务**配置向导页面，完成以下操作并单击**下一步**。
   i. 在**描述**文本框输入迁移上云的任务名称。
   ii. 在**元数据**右侧，单击**点击上传元数据文件**，选择已提前准备好的JSON格式的元数据文件上传。
6. 在**购买实例**配置向导页面，单击**购买实例**。
7. 在**请选择您要创建的实例的付费方式**面板，选择**付费方式**，请根据需要选择**包年包月**或者**按量付费**，然后单击**确定**。
8. 在购买面板，选择**实例类型**，根据自身业务需求选择其他相应的配置，然后单击**立即购买**，根据页面提示完成支付。

> **说明** 如果使用迁移工具导出元数据时，设置 `evaluate` 取值为 `true`，则购买时会根据获取的当前机器规格、近期存储情况、流量、Kafka集群配置信息默认选中推荐的规格。您也可以根据需要重新选择规格购买。

在**实例列表**页面可以看到您刚才购买的实例。

步骤二：部署实例并启动迁移任务

1. 在**迁移上云**页面，找到待部署的目标任务，在其**操作**列，单击**继续配置**。
2. 在**购买实例**配置向导页面，单击**下一步**。
3. 在**部署实例**配置向导页面，选择刚创建未部署的目标实例名称，并完成以下参数配置并单击**部署并创建任务**。

参数	说明	示例
目标实例	迁移到消息队列Kafka版实例的名称。	alikafka_pre-cn-tl32je2j****
VPC ID	实例部署的VPC ID。您可以登录专有网络控制台，在搭建VPC网络所在地域的**交换机**页面，查看目标VPC ID。	vpc-rj91f8ql28reon76a****
vSwitch ID	实例部署的VPC ID。您可以登录专有网络控制台，在搭建VPC网络所在地域的**交换机**页面，查看目标交换机ID。 选择交换机ID后，系统会为您自动选择该交换机所在的可用区。	vsw-rj9dfeeir1xl8px4g****
跨可用区部署	是否跨可用区部署。实例的规格类型为专业版时，可选择配置该参数。 ○ **是**：可以跨可用区部署。跨可用区部署，可以使实例具备较高的容灾能力，可以抵御机房级别的故障。 ○ **否**：不跨可用区部署，实例将在当前可用区中部署。	不涉及
版本	您要部署的消息队列Kafka版实例的版本。版本号与开源版本号对应。	2.2.0
消息保留时间	在磁盘容量充足的情况下，消息的最长保留时间。磁盘容量不足（即磁盘水位达到85%）时，将提前删除旧的消息，以确保服务可用性。	72
最大消息大小	消息队列Kafka版实例能收发的消息的最大值。配置前，请确认是否匹配生产和消费客户端相应配置。	1
消费位点保留时间	消息消费位点的保留时间。默认保留时间为7天，即10080分钟。取值范围为1440分钟～43200分钟。	10080
自定义用户名密码	选择是否自定义实例用户名和密码。公网/VPC实例类型可以选择配置该参数。 ○ **是**：自定义用户名和密码，适用于多个实例共享同一用户名和密码的场景。 ○ **否**：使用系统为每个实例分配的默认用户名和密码。	**否**

参数	说明	示例
ACL 功能	选择是否启用实例的ACL功能。 ○ **启用**：开启实例ACL功能。借助消息队列Kafka版的ACL功能，您可以按需为SASL用户赋予向消息队列Kafka版收发消息的权限，从而实现权限分割。关于SASL用户授权具体操作，请参见SASL用户授权。 ○ **禁用**：不开启实例ACL功能，使用实例的默认SASL用户仅提供身份校验。	不涉及

在**迁移上云**页面可以看到任务状态为**迁移中**。迁移完成后，即可开始使用迁移上云的Topic和Group。

删除迁移任务

在**迁移上云**页面，找到待删除的目标任务，在其右侧**操作**列，单击**删除**。

后续步骤

查看迁移进度

3.5. 查看迁移进度

本文说明如何查看自建Kafka集群迁移到消息队列Kafka版的进度。

前提条件

您已开始自建Kafka集群迁移上云。迁移类型包括：

- 通过命令行迁移
  - 迁移Topic上云
  - 迁移Group上云
  - 迁移数据上云
- 通过元数据文件迁移上云
- 迁移自建Kafka集群至新实例
- 位点迁移

> **注意** 目前不支持将自建Kafka集群的位点迁移到消息队列Kafka版。

操作步骤

> **注意** 消息队列Kafka版不支持数据迁移和位点迁移信息上报，即如果您已经开始数据迁移或位点迁移，您目前无法在消息队列Kafka版控制台查看迁移进度。

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在左侧导航栏，单击**迁移上云**。
4. 在**迁移上云**页面，显示该地域下的所有迁移上云任务。选择要查看的迁移任务，查看**Topic 迁移进度**和**Group 迁移进度**，单击**详情**查看详细信息。

后续步骤

验证迁移结果

3.6. 验证迁移结果

迁移成功后，目标消息队列Kafka版实例下会创建对应的Topic和Group。您可以在**Topic 管理**和**Group 管理**页面的列表中查看。

前提条件

您已将自建Kafka集群迁移至消息队列Kafka版实例中。迁移类型包括：

- 通过命令行迁移
  - 迁移Topic上云
  - 迁移Group上云
  - 迁移数据上云
- 通过元数据文件迁移上云
- 迁移自建Kafka集群至新实例

操作步骤

1. 登录消息队列Kafka版控制台。
2. 在**概览**页面的**资源分布**区域，选择地域。
3. 在**实例列表**页面，单击目标实例名称。
4. 查看资源列表。
   - 在左侧导航栏，单击**Topic 管理**，在**Topic 管理**页面的Topic列表中查看已创建的Topic。
   - 在左侧导航栏，单击**Group 管理**，在**Group 管理**页面的Group列表中查看已创建的Group。

4.云上资源迁移

4.1. 云上迁移Topic

本文介绍如何使用消息队列Kafka版提供的迁移工具将某个消息队列Kafka版实例的Topic迁移到另一个消息队列Kafka版实例。

前提条件

您已完成以下操作：

- 下载JDK 8
- 下载迁移工具kafka-migration-assessment.jar.zip
- 购买并部署消息队列Kafka版实例：
  - 购买并部署VPC实例
  - 购买并部署公网/VPC实例

> **注意**
>
> - 迁移不会删除源消息队列Kafka版实例的Topic，只是在目标消息队列Kafka版实例创建相同配置的Topic。
> - 迁移内容仅为Topic配置，不包含Topic中存储的数据。

操作步骤

1. 打开命令行工具。
2. 使用**cd**命令将路径切换到迁移工具所在目录。
3. 执行以下命令确认要迁移的Topic。

```
java -jar kafka-migration.jar TopicMigrationFromAliyun --sourceAk <yoursourceAccessKeyId> --sourceSk <yoursourceAccessKeySecret> --sourceRegionId <yoursourceRegionId> --sourceInstanceId <yoursourceInstanceId> --destAk <yourdestAccessKeyId> --destSk <yourdestAccessKeySecret> --destRegionId <yourdestRegionId> --destInstanceId <yourdestInstanceId>
```

参数	描述
sourceAk	源消息队列Kafka版实例所属阿里云账号的AccessKey ID
sourceSk	源消息队列Kafka版实例所属阿里云账号的AccessKey Secret
sourceRegionId	源消息队列Kafka版实例的地域ID
sourceInstanceId	源消息队列Kafka版实例的ID
destAk	目标消息队列Kafka版实例所属阿里云账号的AccessKey ID
destSk	目标消息队列Kafka版实例所属阿里云账号的AccessKey Secret

参数	描述
destRegionId	目标消息队列Kafka版实例的地域ID
destInstanceId	目标消息队列Kafka版实例的ID

待确认的返回结果示例如下：

```
15:13:12 INFO - cmd=TopicMigrationFromAliyun, request=null, response={"total":4,"code":
200,"requestId":"1CBAB340-2146-43A3-8470-84D77DB8B43E","success":true,"pageSize":10000,
"currentPage":1,"message":"operation success.","topicList":[{"instanceId":"alikafka_pre
-cn-0pp1cng20***","localTopic":false,"createTime":1578558314000,"regionId":"cn-hangzhou
","statusName":"服务中","topic":"agdagasdg","remark":"agdadgdasg","partitionNum":12,"com
pactTopic":false,"status":0,"tags":[]},{"instanceId":"alikafka_pre-cn-0pp1cng20***","lo
calTopic":false,"createTime":1578558294000,"regionId":"cn-hangzhou","statusName":"服务中
","topic":"135215","remark":"1315215","partitionNum":12,"compactTopic":false,"status":0
,"tags":[]},{"instanceId":"alikafka_pre-cn-0pp1cng20***","localTopic":false,"createTime
":1578558214000,"regionId":"cn-hangzhou","statusName":"服务中","topic":"1332","remark":"
13414","partitionNum":12,"compactTopic":false,"status":0,"tags":[]},{"instanceId":"alik
afka_pre-cn-0pp1cng20***","localTopic":false,"createTime":1578558141000,"regionId":"cn-
hangzhou","statusName":"服务中","topic":"aete","remark":"est","partitionNum":12,"compact
Topic":false,"status":0,"tags":[]}]}
15:13:12 INFO - Will create topic:agdagasdg, isCompactTopic:false, partition number:12
15:13:12 INFO - Will create topic:135215, isCompactTopic:false, partition number:12
15:13:12 INFO - Will create topic:1332, isCompactTopic:false, partition number:12
15:13:12 INFO - Will create topic:aete, isCompactTopic:false, partition number:12
```

4. 执行以下命令提交要迁移的Topic。

```
 java -jar kafka-migration.jar TopicMigrationFromAliyun --sourceAk <yoursourceAccessKeyId>
--sourceSk <yoursourceAccessKeySecret> --sourceRegionId <yoursourceRegionId> --sourceInsta
nceId <yoursourceInstanceId> --destAk <yourdestAccessKeyId> --destSk <yourdestAccessKeySec
ret> --destRegionId <yourdestRegionId> --destInstanceId <yourdestInstanceId> --commit
```

参数	描述
commit	提交迁移

提交迁移的返回结果示例如下：

```
16:38:30 INFO - cmd=TopicMigrationFromAliyun, request=null, response={"code":200,"reque
stId":"A0CA4D70-46D4-45CF-B9E0-B117610A26DB","success":true,"message":"operation succes
s"}
16:38:30 INFO - TopicCreate success, topic=agdagasdg, partition number=12, isCompactTop
ic=false
16:38:36 INFO - cmd=TopicMigrationFromAliyun, request=null, response={"code":200,"reque
stId":"05E88C75-64B6-4C87-B962-A63D906FD993","success":true,"message":"operation succes
s"}
16:38:36 INFO - TopicCreate success, topic=135215, partition number=12, isCompactTopic=
false
16:38:42 INFO - cmd=TopicMigrationFromAliyun, request=null, response={"code":200,"reque
stId":"9D54F6DB-6FA0-4F6D-B19A-09109F70BDDA","success":true,"message":"operation succes
s"}
16:38:42 INFO - TopicCreate success, topic=1332, partition number=12, isCompactTopic=fa
lse
16:38:49 INFO - cmd=TopicMigrationFromAliyun, request=null, response={"code":200,"reque
stId":"6C265013-D15E-49AE-BE55-BF7657ADA1B7","success":true,"message":"operation succes
s"}
16:38:49 INFO - TopicCreate success, topic=aete, partition number=12, isCompactTopic=fa
lse
```

5. 确认Topic迁移是否成功。
   i. 登录消息队列Kafka版控制台。
   ii. 在**概览**页面的**资源分布**区域，选择地域。
   iii. 在**实例列表**页面，单击目标实例名称。
   iv. 在左侧导航栏，单击**Topic 管理**。
   v. 在**Topic 管理**页面的Topic列表中，显示成功迁移的Topic。

4.2. 云上迁移Group

本文介绍如何使用消息队列Kafka版提供的迁移工具将某个消息队列Kafka版实例的Group迁移到另一个消息队列Kafka版实例。

前提条件

您已完成以下操作：

- 下载JDK 8
- 下载迁移工具JAR文件

> **注意**
> - 迁移不会删除源消息队列Kafka版实例的Group，只是在目标消息队列Kafka版实例创建相同配置的Group。
> - 迁移内容仅为Group配置，不包含Group消费的Topic及位点信息。

操作步骤

1. 打开命令行工具。
2. 使用**cd**命令将路径切换到迁移工具所在目录。

3. 确认要迁移的Group。

```
java -jar kafka-migration.jar ConsumerGroupMigrationFromAliyun --sourceAk <yoursourceAccessKeyId> --sourceSk <yoursourceAccessKeySecret> --sourceRegionId <yoursourceRegionId> --sourceInstanceId <yoursourceInstanceId> --destAk <yourdestAccessKeyId> --destSk <yourdestAccessKeySecret> --destRegionId <yourdestRegionId> --destInstanceId <yourdestInstanceId>
```

参数	描述
sourceAk	源消息队列Kafka版实例所属阿里云账号的AccessKey ID
sourceSk	源消息队列Kafka版实例所属阿里云账号的AccessKey Secret
sourceRegionId	源消息队列Kafka版实例的地域ID
sourceInstanceId	源消息队列Kafka版实例的ID
destAk	目标消息队列Kafka版实例所属阿里云账号的AccessKey ID
destSk	目标消息队列Kafka版实例所属阿里云账号的AccessKey Secret
destRegionId	目标消息队列Kafka版实例的地域ID
destInstanceId	目标消息队列Kafka版实例的ID

待确认的返回结果示例如下：

```
10:54:26 INFO - cmd=ConsumerGroupMigrationFromAliyun, request=null, response={"code":200,"requestId":"9793DADB-55A5-4D4E-9E9C-D4DA8B35370C","success":true,"consumerList":[{"instanceId":"alikafka_post-cn-0pp1h0uv6***","regionId":"cn-hangzhou","consumerId":"Demo","tags":[{"value":"","key":"migration"}]}],"message":"operation success."}
10:54:26 INFO - Will create consumer groups:[Demo]
```

4. 执行以下命令提交要迁移的Group。

```
java -jar kafka-migration.jar ConsumerGroupMigrationFromAliyun --sourceAk LTAI4FwQ5aK1mFYCspJ1**** --sourceSk wvDxjjRQ1tHPiL0oj7Y2Z7WDNk**** --sourceRegionId cn-hangzhou --sourceInstanceId alikafka_post-cn-0pp1h0uv**** --destAk LTAI4FwQ5aK1mFYCspJ1**** --destSk wvDxjjRQ1tHPiL0oj7Y2Z**** --destRegionId cn-hangzhou --destInstanceId alikafka_pre-cn-v0h1cng0**** --commit
```

参数	说明
commit	提交迁移

提交迁移的返回结果示例如下：

```
10:54:40 INFO - cmd=ConsumerGroupMigrationFromAliyun, request=null, response={"code":20
0,"requestId":"49E53B79-3C2C-4BCF-8BC8-07B0BB14B52A","success":true,"consumerList":[{"i
nstanceId":"alikafka_post-cn-0pp1h0uv6***","regionId":"cn-hangzhou","consumerId":"Demo"
,"tags":[{"value":"","key":"migration"}]}],"message":"operation success."}
10:54:41 INFO - cmd=ConsumerGroupMigrationFromAliyun, request=null, response={"code":20
0,"requestId":"5AEEFB13-2A6B-4265-97CB-902CFA483339","success":true,"message":"operatio
n success"}
10:54:41 INFO - ConsumerCreate success, consumer group=Demo
```

5. 确认Group迁移是否成功。
   i. 登录消息队列Kafka版控制台。
   ii. 在**概览**页面的**资源分布**区域，选择地域。
   iii. 在**实例列表**页面，单击目标实例名称。
   iv. 在左侧导航栏，单击**Group 管理**。
   v. 在**Group 管理**页面的Group列表显示成功迁移的Group。