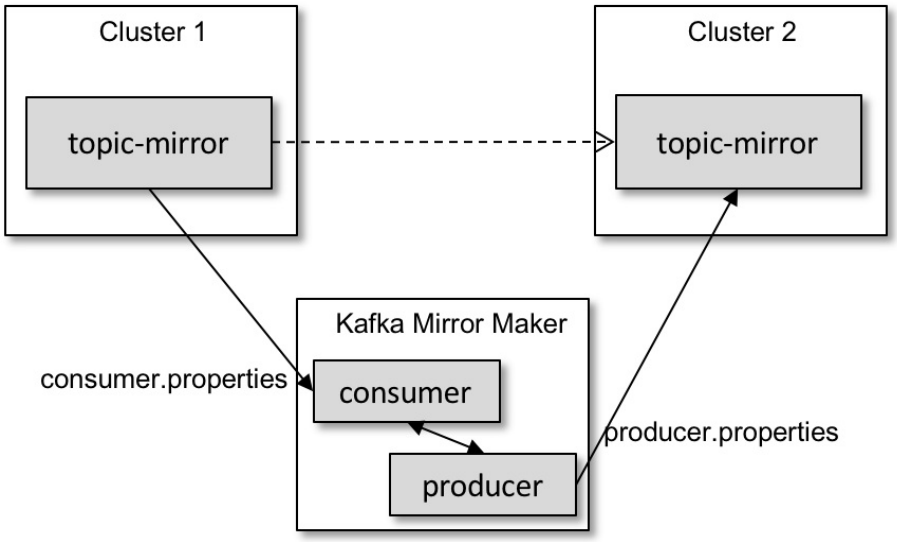
Kafka Mirror Maker

Kafka Mirror Maker 是用于在两个集群之间同步数据的一个工具，其实现原理是通过从源集群中消费消息，然后将消息生产到目标集群中，也就是普通的生产和消费消息。如果了解 RabbitMQ，那么会发现这个工具和 RabbitMQ 中的数据迁移插件 Federation/Shovel 的实现原理如出一辙。用户只需要在启动 Kafka Mirror Maker 时指定一些简单的消费端和生产端配置就可以实现准实时的数据同步。



如上图所示，我们需要将集群 Cluster 1 中的消息同步到集群 Cluster 2 中。通过 Kafka Mirror Maker 做一个中间的周转站，我们就可以很容易地实现跨集群的数据同步。

在上一节中，我们了解了 Kafka Connect 的相关用法，它和 Kafka Mirror Maker 的区别在于：Kafka Connect 用于其他数据存储系统与 Kafka 之间的数据复制，而不是 Kafka 与 Kafka 之间的数据复制。在第17节中，分区重分配可以实现 Kafka 与 Kafka 之间的数据复制，它与 Kafka Mirror Maker 的区别在于它是单个集群内部的数据复制，而不是跨集群之间的数据复制。

Kafka Mirror Maker 可以在两个不同的数据中心（两个集群位于不同的数据中心）中同步（镜像）数据。我们可以在两个不同的数据中心中部署一个集群，各个数据中心持有集群中的部分 broker 节点，通过将副本分散到不同的数据中心来实现不同数据中心的数据同步。但这样有一个严重的问题，即高延迟，这个问题会严重影响 Kafka 和 ZooKeeper 的性能，还有可能引发严重的异常。

下面我们来了解一下 Kafka Mirror Maker 的用法，它具体对应 Kafka 中的 kafka-mirror-maker.sh 脚本。参考上图，我们演示从 Cluster 1 中将主题 topic-mirror 的数据同步到 Cluster 2 中，首先创建并配置两个配置文件，参考如下：

```
# consumer.properties的配置
bootstrap.servers=cluster1:9092
group.id=groupIdMirror
client.id=sourceMirror
partition.assignment.strategy=org.apache.kafka.cl
ients.consumer.RoundRobinAssignor
# producer.properties的配置
bootstrap.servers=cluster2:9092
client.id=sinkMirror
```

consumer.properties 和 producer.properties 这两个配置文件中的配置对应消费者客户端和生产者客户端的配置，具体可以参考一下前面章节的内容。

下面就可以启动 Kafka Mirror Maker 了，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-mirror-
maker.sh --consumer.config consumer.properties --
producer.config producer.properties --whitelist
'topic-mirror'
```

kafka-mirror-maker.sh 脚本中有多个可配置的参数，如下表所示。

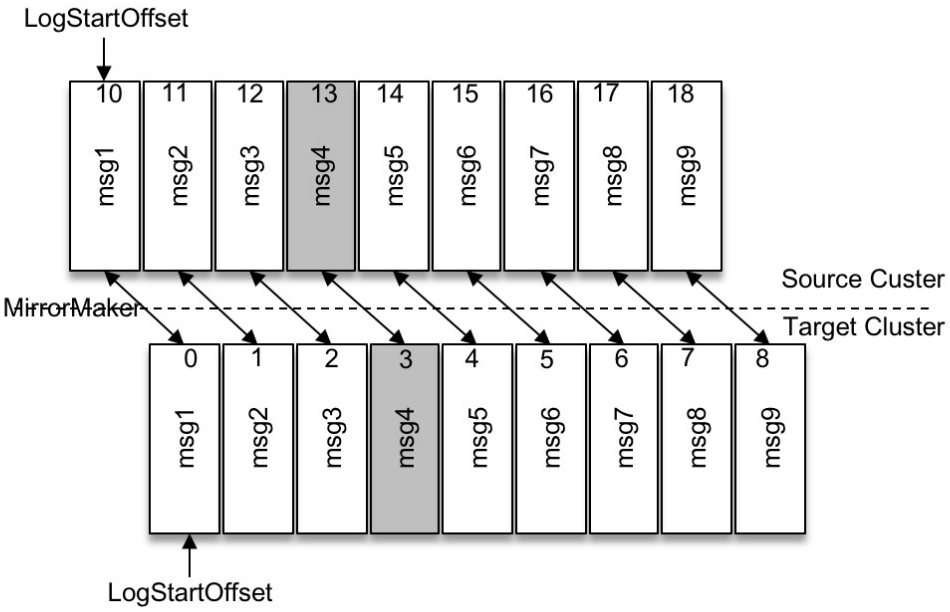
参 数	释 义
abort.on.send.failure	默认为 true
consumer.config	用于指定消费者的配置文件，配置文件里有两个必填的参数：boostrap.servers 和 group.id
consumer.rebalance.listener	指定再均衡监听器
help	打印帮助信息
message.handler	指定消息的处理器。这个处理器会在消费者消费到消息之后且在生产者发送消息之前被调用
message.handler.args	指定消息处理器的参数，同 message.handler 一起使用
num.streams	指定消费线程的数量
offset.commit.interval.ms	指定消费位移提交间隔
producer.config	用于指定生产者的配置文件，配置文件里唯一必填的参数是 bootstrap.servers
rebalance.listener.args	指定再均衡监听器的参数，同 consumer.rebalance.listener 一起使用

b表示复制源集群中主题a和主

whitelist	指定需要复制的源集群中的主题。这个参数可以指定一个正则表达式，比如a	题b，的数据。为了方便使用，这里也允许将“

注意，不要在单个集群的内部使用 Kafka Mirror Maker，否则会循环复制。如果在配置文件 consumer.properties 中配置的 bootstrap.servers 和在配置文件 producer.properties 中配置的 bootstrap.servers 的 broker 节点地址列表属于同一个集群，启动 Kafka Mirror Maker 之后，只要往主题 topic-mirror 中输入一条数据，那么这条数据会在这个主题内部无限循环复制，直至 Kafka Mirror Maker 关闭。

由于 kafka-mirror-maker.sh 脚本是启动一个生产者和一个消费者进行数据同步操作的，因此数据同步完成后，该命令依然在等待新的数据进行同步，也就是需要用户自己查看数据是否同步完成，在保证数据同步完成后手动关闭该命令。同时，用户可以在目标集群中创建主题，主题的分区数及副本因子可以与源集群中该主题对应的分区数及副本因子不一致。可以将目标集群中的 auto.create.topics.enable 参数配置为 true，以确保在同步操作时有对应的主题，不过建议在同步之前先确认是否有相关的主题，如果没有则手工创建，或者采用自定义的元数据同步工具进行创建。



源集群和目标集群是两个完全独立的实体。对每个主题而言，两个集群之间的分区数可能不同；就算分区数相同，那么经过消费再生产之后消息所规划到的分区号也有可能不同；就算分区数相同，消息所规划到的分区号也相同，那么消息所对应的 offset 也有可能不相同。

参考上图，源集群中由于执行了某次日志清理操作，某个分区的 logStartOffset 值变为10，而目标集群中对应分区的 logStartOffset 还是0，那么从源集群中原封不动地复制到目标集群时，同一条消息的 offset 也不会相同。如果要实现客户端生产消费的迁移（将通信链路从源集群中切换到目标集群中），在数据同步完成之后，也不可能不做任何改变就能实现完美的切换。

不过，如果能够做到源集群中的消息除 offset 外都在目标集群中一致（比如消息的分区号相同，主题的分区数相同），那么可以试着通过 kafka-consumer-group.sh 脚本重置消费位移（参考第27节）来实现合理的客户端迁移切换。或者先将生产者的链路切换到目标集群，然后等待消费者消费完源集群中的消息之后再将它的链路切换到目标集群。

kafka-mirror-maker.sh 脚本对应的实现类是 kafka.tools.MirrorMaker，它只有500多行代码，很多时候我们会把它与同类产品 uReplicator 进行对比，笔者觉得这样有失稳妥，前者的定位只是一个工具，而后者是一个完备的工程项目，它们都有各自的适用场景。不过话又说回来，uReplicator 底层也是基于 MirrorMaker 进行构建的，并针对 MirrorMaker 做了大量的调优及工程化改造，具体的内容可以参考官网介绍：http://eng.uber.com/ureplicator/。