查看主题

第16节中提及了 kafka-topics.sh 脚本有5种指令类型：create、list、describe、alter 和 delete。其中 list 和 describe 指令可以用来方便地查看主题信息，在前面的内容中我们已经接触过了 describe 指令的用法，本节会对其做更细致的讲述。

通过 list 指令可以查看当前所有可用的主题，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka –list
__consumer_offsets
topic-create
topic-demo
topic-config
```

前面的章节我们都是通过 describe 指令来查看单个主题信息的，如果不使用 --topic 指定主题，则会展示出所有主题的详细信息。--topic 还支持指定多个主题，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-create,topic-demo
Topic:topic-create  PartitionCount:4
ReplicationFactor:2 Configs:
    Topic: topic-create Partition: 0    Leader: 2
Replicas: 2,0   Isr: 2,0
    Topic: topic-create Partition: 1    Leader: 0
Replicas: 0,1   Isr: 0,1
    Topic: topic-create Partition: 2    Leader: 2
Replicas: 1,2   Isr: 2,1
    Topic: topic-create Partition: 3    Leader: 2
Replicas: 2,1   Isr: 2,1
Topic:topic-demo    PartitionCount:4
ReplicationFactor:3 Configs:
    Topic: topic-demo   Partition: 0    Leader: 2
Replicas: 2,1,0 Isr: 2,0,1
    Topic: topic-demo   Partition: 1    Leader: 2
Replicas: 0,2,1 Isr: 2,0,1
    Topic: topic-demo   Partition: 2    Leader: 2
Replicas: 1,0,2 Isr: 2,0,1
    Topic: topic-demo   Partition: 3    Leader: 2
Replicas: 2,0,1 Isr: 2,0,1
```

在使用 describe 指令查看主题信息时还可以额外指定 topics-with-overrides、under-replicated-partitions 和 unavailable-partitions 这三个参数来增加一些附加功能。

增加 topics-with-overrides 参数可以找出所有包含覆盖配置的主题，它只会列出包含了与集群不一样配置的主题。注意使用 topics-with-overrides 参数时只显示原本只使用 describe 指令的第一行信息，参考示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topics-with-overrides
Topic:__consumer_offsets    PartitionCount:50
ReplicationFactor:1
Configs:segment.bytes=104857600,cleanup.policy=co
mpact,compression.type=producer
Topic:topic-config  PartitionCount:1
ReplicationFactor:1
Configs:cleanup.policy=compact,max.message.bytes=
10000
```

under-replicated-partitions 和 unavailable-partitions 参数都可以找出有问题的分区。通过 under-replicated-partitions 参数可以找出所有包含失效副本的分区。包含失效副本的分区可能正在进行同步操作，也有可能同步发生异常，此时分区的 ISR 集合小于AR 集合。对于通过该参数查询到的分区要重点监控，因为这很可能意味着集群中的某个 broker 已经失效或同步效率降低等。

举个例子，参照主题 topic-create 的环境，我们将集群中的 node2 节点下线，之后再通过这个参数来查看 topic-create 的信息，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-create --under-replicated-
partitions
    Topic: topic-create Partition: 1    Leader: 0
Replicas: 0,1   Isr: 0
    Topic: topic-create Partition: 2    Leader: 2
Replicas: 1,2   Isr: 2
    Topic: topic-create Partition: 3    Leader: 2
Replicas: 2,1   Isr: 2
```

我们再将 node2 节点恢复，执行同样的命令，可以看到没有任何信息显示：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-topics.sh --zookeeper localhost:2181/ kafka --describe --topic topic-create --under-replicated-partitions
```

通过 unavailable-partitions 参数可以查看主题中没有 leader 副本的分区，这些分区已经处于离线状态，对于外界的生产者和消费者来说处于不可用的状态。

举个例子，参考主题 topic-create 的环境，我们将集群中的 node2 和 node3 节点下线，之后再通过这个参数来查看 topic-create 的信息，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-create --unavailable-
partitions
    Topic: topic-create Partition: 2    Leader:
-1  Replicas: 1,2   Isr: 1
    Topic: topic-create Partition: 3    Leader:
-1  Replicas: 2,1   Isr: 1

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-create
Topic:topic-create  PartitionCount:4
ReplicationFactor:2 Configs:
    Topic: topic-create Partition: 0    Leader: 0
Replicas: 2,0   Isr: 0
    Topic: topic-create Partition: 1    Leader: 0
Replicas: 0,1   Isr: 0
    Topic: topic-create Partition: 2    Leader:
-1  Replicas: 1,2   Isr: 1
    Topic: topic-create Partition: 3    Leader:
-1  Replicas: 2,1   Isr: 1
```

我们再将 node2 和 node3 恢复，执行同样的命令，可以看到没有任何信息：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-create --unavailable-
partitions
```

修改主题

当一个主题被创建之后，依然允许我们对其做一定的修改，比如修改分区个数、修改配置等，这个修改的功能就是由 kafka-topics.sh 脚本中的 alter 指令提供的。

我们首先来看如何增加主题的分区数。以前面的主题 topic-config 为例，当前分区数为1，修改为3，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-config --partitions 3
WARNING: If partitions are increased for a topic
that has a key, the partition logic or ordering
of the messages will be affected
Adding partitions succeeded!

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/kafka --
describe --topic topic-config
Topic:topic-config  PartitionCount:3
ReplicationFactor:1 Configs:
    Topic: topic-config Partition: 0    Leader: 2
Replicas: 2 Isr: 2
    Topic: topic-config Partition: 1    Leader: 0
Replicas: 0 Isr: 0
    Topic: topic-config Partition: 2    Leader: 1
Replicas: 1 Isr: 1
```

注意上面提示的告警信息：当主题中的消息包含 key 时（即 key 不为 null），根据 key 计算分区的行为就会受到影响。当 topic-config 的分区数为1时，不管消息的 key 为何值，消息都会发往这一个分区；当分区数增加到3时，就会根据消息的 key 来计算分区号，原本发往分区0的消息现在有可能会发往分区1或分区2。如此还

会影响既定消息的顺序，所以在增加分区数时一定要三思而后行。对于基于 key 计算的主题而言，建议在一开始就设置好分区数量，避免以后对其进行调整。

目前 Kafka 只支持增加分区数而不支持减少分区数。比如我们再将主题 topic-config 的分区数修改为1，就会报出 InvalidPartitionException 的异常，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-config --partitions 1
WARNING: If partitions are increased for a topic
that has a key, the partition logic or ordering
of the messages will be affected
Error while executing topic command : The number
of partitions for a topic can only be increased.
Topic topic-config currently has 3 partitions, 1
would not be an increase.
[2018-09-10 19:28:40,031] ERROR
org.apache.kafka.common.errors.InvalidPartitionsE
xception: The number of partitions for a topic
can only be increased. Topic topic-config
currently has 3 partitions, 1 would not be an
increase.
 (kafka.admin.TopicCommand$)
```

为什么不支持减少分区?
按照 Kafka 现有的代码逻辑，此功能完全可以实现，不过也会使代码的复杂度急剧增大。实现此功能需要考虑的因素很多，比如删除的分区中的消息该如何处理？如果随着分区一起消失则消息的可靠性得不到保障；如果需要保留则又需要考虑如何保留。直接存储到现有分区的尾部，消息的时间戳就不会递增，如此对于 Spark、Flink 这类需要消息时间戳（事件时间）的组件将会受到影响；如果分散插入现有的分区，那么在消息量很大的时候，内部的数据复制会占用很大的资源，而且在复制期间，此主题的可用性又如何得到保障？与此同时，顺序性问题、事务性问题，以及分区和副本的状态机切换问题都是不得不面对的。反观这个功能的收益点却是很低的，如果真的需要实现此类功能，则完全可以重新创建一个分区数较小的主题，然后将现有主题中的消息按照既定的逻辑复制过去即可。

在创建主题时有一个 if-not-exists 参数来忽略一些异常，在这里也有对应的参数，如果所要修改的主题不存在，可以通过 if-exists 参数来忽略异常。下面修改一个不存在的主题 topic-unknown 的分区，会报出错误信息“Topic topic-unknown does not exist”，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-unknown --partitions 4
Error while executing topic command : Topic
topic-unknown does not exist on ZK path
localhost:2181/kafka
[2018-09-11 11:14:55,458] ERROR
java.lang.IllegalArgumentException: Topic topic-
unknown does not exist on ZK path
localhost:2181/kafka
    at
kafka.admin.TopicCommand$.alterTopic(TopicCommand
.scala:123)
    at
kafka.admin.TopicCommand$.main(TopicCommand.scala:
65)
    at
kafka.admin.TopicCommand.main(TopicCommand.scala)
 (kafka.admin.TopicCommand$)

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-unknown --partitions 4 --if-
exists
```

除了修改分区数，我们还可以使用 kafka-topics.sh 脚本的 alter 指令来变更主题的配置。在创建主题的时候我们可以通过 config 参数来设置所要创建主题的相关参数，通过这个参数可以覆盖原本的默认配置。在创建完主题之后，我们还可以通过 alter 指令配合 config 参数增加或修改一些配置以覆盖它们配置原有的值。

下面的示例中演示了将主题 topic-config 的 max.message.bytes 配置值从10000修改为20000，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper 1ocalhost:2181/ kafka --
describe --topic topic-config
Topic:topic-config  PartitionCount:1
ReplicationFactor:1
Configs:cleanup.policy=compact,max.message.bytes=
10000
        Topic: topic-config Partition: 0
Leader: 2   Replicas: 2 Isr: 2

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper 1ocalhost:2181/ kafka --
alter --topic topic-config --config
max.message.bytes=20000
WARNING: Altering topic configuration from this
script has been deprecated and may be removed in
future releases.
         Going forward, please use kafka-
configs.sh for this functionality
Updated config for topic "topic-config".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper 1ocalhost:2181/ kafka --
describe --topic topic-config
Topic:topic-config  PartitionCount:1
ReplicationFactor:1
Configs:max.message.bytes=20000,cleanup.policy=co
mpact
        Topic: topic-config Partition: 0
Leader: 2   Replicas: 2 Isr: 2
```

我们再次覆盖主题 topic-config 的另一个配置 segment.bytes（看上去相当于增加动作），示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-config --config
segment.bytes=1048577
WARNING: Altering topic configuration from this
script has been deprecated and may be removed in
future releases.
         Going forward, please use kafka-
configs.sh for this functionality
Updated config for topic "topic-config".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-config
Topic:topic-config  PartitionCount:1
ReplicationFactor:1
Configs:segment.bytes=1048577,cleanup.policy=comp
act,max.message.bytes=20000
    Topic: topic-config Partition: 0    Leader: 2
Replicas: 2 Isr: 2
```

我们可以通过 delete-config 参数来删除之前覆盖的配置，使其恢复原有的默认值。下面的示例将主题 topic-config 中所有修改过的3个配置都删除：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-config --delete-config
segment.bytes
WARNING: Altering topic configuration from this
script has been deprecated and may be removed in
future releases.
         Going forward, please use kafka-
configs.sh for this functionality
Updated config for topic "topic-config".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
alter --topic topic-config --delete-config
max.message.bytes --delete-config cleanup.policy
WARNING: Altering topic configuration from this
script has been deprecated and may be removed in
future releases.
         Going forward, please use kafka-
configs.sh for this functionality
Updated config for topic "topic-config".

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
describe --topic topic-config
Topic:topic-config  PartitionCount:1
ReplicationFactor:1 Configs:
    Topic: topic-config Partition: 0    Leader: 2
Replicas: 2 Isr: 2
```

注意到在变更（增、删、改）配置的操作执行之后都会提示一段告警信息，指明了使用 kafka-topics.sh 脚本的 alter 指令来变更主题配置的功能已经过时（deprecated），将在未来的版本中删除，并且

推荐使用 kafka-configs.sh 脚本来实现相关功能。

删除主题

如果确定不再使用一个主题，那么最好的方式是将其删除，这样可以释放一些资源，比如磁盘、文件句柄等。kafka-topics.sh 脚本中的 delete 指令就可以用来删除主题，比如删除一个主题 topic-delete：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
delete --topic topic-delete
Topic topic-delete is marked for deletion.
Note: This will have no impact if
delete.topic.enable is not set to true.
```

可以看到在执行完删除命令之后会有相关的提示信息，这个提示信息和 broker 端配置参数 delete.topic.enable 有关。必须将 delete.topic.enable 参数配置为 true 才能够删除主题，这个参数的默认值就是 true，如果配置为 false，那么删除主题的操作将会被忽略。在实际生产环境中，建议将这个参数的值设置为 true。

如果要删除的主题是 Kafka 的内部主题，那么删除时就会报错。截至 Kafka 2.0.0，Kafka 的内部一共包含2个主题，分别为 __consumer_offsets和__transaction_state。下面的示例中尝试删除内部主题__consumer_offsets：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
delete --topic __consumer_offsets
Error while executing topic command : Topic
__consumer_offsets is a kafka internal topic and
is not allowed to be marked for deletion.
[2018-09-11 11:30:32,635] ERROR
kafka.admin.AdminOperationException: Topic
__consumer_offsets is a kafka internal topic and
is not allowed to be marked for deletion.
… (省略若干项)
```

尝试删除一个不存在的主题也会报错。比如下面的示例中尝试删除一个不存在的主题 topic-unknown：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
delete --topic topic-unknown
Error while executing topic command : Topic
topic-unknown does not exist on ZK path
localhost:2181/kafka
[2018-09-11 23:43:22,186] ERROR
java.lang.IllegalArgumentException: Topic topic-
unknown does not exist on ZK path
localhost:2181/kafka
… (省略若干项)
```

这里同 alter 指令一样，也可以通过 if-exists 参数来忽略异常，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
topics.sh --zookeeper localhost:2181/ kafka --
delete --topic topic-unknown --if-exists
[root@node1 kafka_2.11-2.0.0]#
```

使用 kafka-topics.sh 脚本删除主题的行为本质上只是在 ZooKeeper 中的/admin/delete_topics路径下创建一个与待删除主题同名的节点，以此标记该主题为待删除的状态。与创建主题相同的是，真正删除主题的动作也是由 Kafka 的控制器负责完成的。

了解这一原理之后，我们可以直接通过 ZooKeeper 的客户端来删除主题。下面示例中使用 ZooKeeper 客户端 zkCli.sh 来删除主题 topic-delete：

```
[zk: localhost:2181/kafka (CONNECTED) 15] create
/admin/delete_topics/topic- delete ""
Created /admin/delete_topics/topic-delete
```

我们还可以通过手动的方式来删除主题。主题中的元数据存储在 ZooKeeper 中的/brokers/topics和/config/topics路径下，主题中的消息数据存储在 log.dir 或 log.dirs 配置的路径下，我们只需要手动删除这些地方的内容即可。下面的示例中演示了如何删除主题 topic-delete，总共分3个步骤，第一步和第二步的顺序可以互换。

第一步，删除 ZooKeeper 中的节点/config/topics/topic-delete。

```
[zk: localhost:2181/kafka (CONNECTED) 7] delete
/config/topics/topic-delete
```

第二步，删除 ZooKeeper 中的节点/brokers/topics/topic-delete 及其子节点。

```
[zk: localhost:2181/kafka (CONNECTED) 8] rmr
/brokers/topics/topic-delete
```

第三步，删除集群中所有与主题 topic-delete 有关的文件。

```
#集群中的各个broker节点中执行rm -rf /tmp/kafka-
logs/topic-delete*命令来删除与主题topic-delete有关的
文件
[root@node1 kafka_2.11-2.0.0]# rm -rf /tmp/kafka-
logs/topic-delete*
[root@node2 kafka_2.11-2.0.0]# rm -rf /tmp/kafka-
logs/topic-delete*
[root@node3 kafka_2.11-2.0.0]# rm -rf /tmp/kafka-
logs/topic-delete*
```

注意，删除主题是一个不可逆的操作。一旦删除之后，与其相关的所有消息数据会被全部删除，所以在执行这一操作的时候也要三思而后行。

介绍到这里，基本上 kafka-topics.sh 脚本的使用也就讲完了，为了方便读者查阅，下表列出了所有 kafka-topics.sh 脚本中的参数。读者也可以通过执行无任何参数的 kafka-topics.sh 脚本，或者执行 kafka-topics.sh -help 来查看帮助信息。

参 数 名 称	释 义
alter	用于修改主题，包括分区数及主题的配置
config <键值对>	创建或修改主题时，用于设置主题级别的参数
create	创建主题
delete	删除主题
delete-config <配置名称>	删除主题级别被覆盖的配置
describe	查看主题的详细信息

disable-rack-aware	创建主题时不考虑机架信息
help	打印帮助信息
if-exists	修改或删除主题时使用，只有当主题存在时才会执行动作
if-not-exists	创建主题时使用，只有主题不存在时才会执行动作
list	列出所有可用的主题
partitions <分区数>	创建主题或增加分区时指定分区数
replica-assignment <分配方案>	手工指定分区副本分配方案
replication-factor <副本数>	创建主题时指定副本因子
topic <主题名称>	指定主题名称
topics-with-overrides	使用 describe 查看主题信息时，只展示包含覆盖配置的主题
unavailable-partitions	使用 describe 查看主题信息时，只展示包含没有 leader 副本的分区
under-replicated-partitions	使用 describe 查看主题信息时，只展示包含失效副本的分区
zookeeper	指定连接的 ZooKeeper 地址信息（必填项）