消费组管理

在 Kafka 中，我们可以通过 kafka-consumer-groups.sh 脚本查看或变更消费组的信息。我们可以通过 list 这个指令类型的参数来罗列出当前集群中所有的消费组名称，示例如下（这个功能对应 KafkaAdminClient 中的 listConsumerGroups() 方法）：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --list
console-consumer-98513
groupIdMonitor
console-consumer-49560
console-consumer-69403
console-consumer-66179
console-consumer-33348
console-consumer-82390
console-consumer-38225
```

注意，在之前的版本中还可以通过 zookeeper 参数来连接指定的 ZooKeeper 地址，因为在旧版的 Kafka 中可以将消费组的信息存储在 ZooKeeper 节点中，不过在2.0.0版本中已经将这个参数删除了，目前只能通过正统的 bootstrap-server 参数来连接 Kafka 集群以此来获取消费者的相应信息。

kafka-consumer-groups.sh 脚本还可以配合 describe 这个指令类型的参数来展示某一个消费组的详细信息，不过要完成此功能还需要配合 group 参数来一同实现，group 参数用来指定特定消费组的名称。下面的示例中展示了消费组 groupIdMonitor 的详细信息

（这个功能对应 KafkaAdminClient 中的 describeConsumerGroups(Collection<String> groupIds) 方法）：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor

TOPIC           PARTITION  CURRENT-OFFSET  LOG-
END-OFFSET  LAG             CONSUMER-ID
HOST            CLIENT-ID
topic-monitor   0          668             668
0               consumer-1-063cdec2-b525-4ba3-
bbfe-db9a92e3b21d /192.168.0.2  consumer-1
topic-monitor   1          666             666
0               consumer-1-063cdec2-b525-4ba3-
bbfe-db9a92e3b21d /192.168.0.2  consumer-1
topic-monitor   2          666             666
0               consumer-1-273faaf0-c950-44a8-
8a11-41a116f79fd4 /192.168.0.2  consumer-1
```

在展示的结果中包含多个字段的信息，其中 TOPIC 表示消费组订阅的主题名称；PARTITION 表示对应的分区编号；CURRENT-OFFSET 表示消费组最新提交的消费位移；LOG-END-OFFSET 表示的是HW（高水位）；LAG表示消息滞后的数量，是 LOG-END-OFFSET 与 CURRENT-OFFSET 的数值之差，详细内容还可以参考32节。CUNSUMER_ID 表示消费组的成员ID，对应于 member_id；HOST 表示消费者的 host 信息；CLIENT-ID 对应于消费者客户端中的 clientId。

消费组一共有 Dead、Empty、PreparingRebalance、CompletingRebalance、Stable 这几种状态，正常情况下，一个具有消费者成员的消费组的状态为 Stable。我们可以通过 state 参数

来查看消费组当前的状态，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
--state

COORDINATOR (ID)          ASSIGNMENT-STRATEGY
STATE                #MEMBERS
192.168.0.4:9092 (2)     range
Stable               2
```

如果消费组内没有消费者，那么对应的状态为 Empty，示例如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
--state
Consumer group 'groupIdMonitor' has no active
members.

COORDINATOR (ID)          ASSIGNMENT-STRATEGY
STATE                #MEMBERS
192.168.0.4:9092 (2)
Empty                0
```

我们还可以通过 members 参数罗列出消费组内的消费者成员信息，参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
--members

CONSUMER-ID
HOST            CLIENT-ID          #PARTITIONS
consumer-1-273faaf0-c950-44a8-8a11-41a116f79fd4
/192.168.0.2  consumer-1      1
consumer-1-063cdec2-b525-4ba3-bbfe-db9a92e3b21d
/192.168.0.2  consumer-1      2
```

如果在此基础上再增加一个 verbose 参数，那么还会罗列出每个消费者成员的分配情况，如下所示。

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
--members --verbose

CONSUMER-ID
HOST            CLIENT-ID       #PARTITIONS
ASSIGNMENT
consumer-1-063cdec2-b525-4ba3-bbfe-db9a92e3b21d
/192.168.0.2  consumer-1      2          topic-
monitor(0,1)
consumer-1-b5bb268b-d077-4db8-b525-9d60cd0ee06b
/192.168.0.2  consumer-1      1          topic-
monitor(2)
```

我们可以通过 delete 这个指令类型的参数来删除一个指定的消费组，不过如果消费组中有消费者成员正在运行，则删除操作会失败，详细参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --delete --group groupIdMonitor
Error: Deletion of some consumer groups failed:
* Group 'groupIdMonitor' could not be deleted due
to: NON_EMPTY_GROUP

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --delete --group groupIdMonitor
Deletion of requested consumer groups
('groupIdMonitor') was successful.

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
Error: Consumer group 'groupIdMonitor' does not
exist.
```

在 KafkaAdminClient 中也有一个 deleteConsumerGroups(Collection<String> groupIds) 方法用来删除指定的消费组。

消费位移管理

kafka-consumer-groups.sh 脚本还提供了重置消费组内消费位移的功能，具体是通过 reset-offsets 这个指令类型的参数来实施的，不过实现这一功能的前提是消费组内没有正在运行的消费者成员。下面的示例将消费组中的所有分区的消费位移都置为0，详细参考如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
```

```
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --all-
topics --reset-offsets --to-earliest --execute
Error: Assignments can only be reset if the group
'groupIdMonitor' is inactive, but the current
state is Stable.

TOPIC                          PARTITION  NEW-
OFFSET

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --all-
topics --reset-offsets --to-earliest --execute

TOPIC                          PARTITION  NEW-
OFFSET
topic-monitor                  1          0
topic-monitor                  0          0
topic-monitor                  2          0

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
Consumer group 'groupIdMonitor' has no active
members.

TOPIC        PARTITION   CURRENT-OFFSET   LOG-
END-OFFSET    LAG       CONSUMER-ID       HOST
CLIENT-ID
topic-monitor    1           0
999             999             -               -
-
```

```
topic-monitor     0              0
1001            1001              -                 -
-
topic-monitor     2              0
1000            1000              -                 -
-
```

可以通过将 --all-topics 修改为 --topic 来实现更加细粒度的消费位移的重置，all-topics 参数指定了消费组中所有主题，而 topic 参数可以指定单个主题，甚至可以是主题中的若干分区。下面的示例将主题 topic-monitor 分区2的消费位移置为分区的末尾：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --topic
topic-monitor:2 --reset-offsets --to-latest --
execute

TOPIC                          PARTITION  NEW-
OFFSET
topic-monitor                  2          1000

[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
Consumer group 'groupIdMonitor' has no active
members.

TOPIC          PARTITION    CURRENT-OFFSET   LOG-
END-OFFSET   LAG          CONSUMER-ID     HOST
CLIENT-ID
topic-monitor    1               0
999             999               -
-               -
topic-monitor    0               0
1001            1001              -
-               -
topic-monitor    2            1000
1000               0               -
-               -
```

前面的两个示例中各自使用了 to-earliest 和 to-latest 参数来分别将消费位移调整到分区的开头和末尾。除此之外，kafka-consumer-groups.sh 脚本还提了更多的选择。

- by-duration <String: duration>：将消费位移调整到距离当前时间指定间隔的最早位移处。duration 的格式为“PnDTnHnMnS”。
- from-file <String: path to CSV file>：将消费位移重置到CSV文件中定义的位置。
- shift-by <Long: number-of-offsets>：把消费位移调整到当前位移 + number-of-offsets 处，number-of-offsets 的值可以为负数。
- to-current：将消费位移调整到当前位置处。
- to-datetime <String: datatime>：将消费位移调整到大于给定时间的最早位移处。datatime 的格式为“YYYY-MM-DDTHH:mm:SS.sss”。
- to-offset <Long: offset>：将消费位移调整到指定的位置。

kafka-consumer-groups.sh 脚本中还有两个参数 dry-run 和 export，dry-run 是只打印具体的调整方案而不执行，export 是将位移调整方案以 CSV 的格式输出到控制台，而 execute 才会执行真正的消费位移重置。下面的示例演示了 execute、dry-run、export、to-current、shift-by、from-file 的具体用法：

```
# 查看当前消费组的消费位移
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092 --describe --group groupIdMonitor
Consumer group 'groupIdMonitor' has no active
members.

TOPIC           PARTITION   CURRENT-OFFSET   LOG-
END-OFFSET   LAG        CONSUMER-ID      HOST
CLIENT-ID
topic-monitor      1           999
999             0               -
-               -
```

```
topic-monitor      0          1001
1001             0                -
-                -
topic-monitor      2          1000
1000             0                -
-                -
# 将消费位移往前调整10，但是不执行
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --topic
topic-monitor --reset-offsets --shift-by -10 --
dry-run

TOPIC                          PARTITION  NEW-
OFFSET
topic-monitor                  2          990
topic-monitor                  1          989
topic-monitor                  0          991
# 将消费位移调整为当前位移并将结果输出到控制台，但是也不执
行
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --topic
topic-monitor --reset-offsets --to-current --
export -dry-run
topic-monitor,2,1000
topic-monitor,1,999
topic-monitor,0,1001
# 将消费位移再次往前调整20并输出结果，但是不执行
[root@node1 kafka_2.11-2.0.0]# bin/kafka-
consumer-groups.sh --bootstrap-server
localhost:9092  --group groupIdMonitor --topic
topic-monitor --reset-offsets --shift-by -20 --
```

```
export --dry-run
topic-monitor,2,980
topic-monitor,1,979
topic-monitor,0,981
# 中间步骤：将上面的输出结果保存到offsets.csv文件中
# 通过from-file参数从offsets.csv文件中获取位移重置策略，并且执行
[root@node1 kafka_2.11-2.0.0]# bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092  --group groupIdMonitor --topic topic-monitor --reset-offsets --from-file offsets.csv --execute

TOPIC                          PARTITION  NEW-OFFSET
topic-monitor                  2          980
topic-monitor                  1          979
topic-monitor                  0          981
# 最终消费位移都往前重置了20
[root@node1 kafka_2.11-2.0.0]# bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group groupIdMonitor
Consumer group 'groupIdMonitor' has no active members.

TOPIC           PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG        CONSUMER-ID     HOST            CLIENT-ID
topic-monitor   1          979             999             20         -               -               -
topic-monitor   0          981             1001            20         -               -               -
```

```
-                    -
topic-monitor    2              980
1000                 20                  -
-                    -
```

手动删除消息

kafka-delete-records.sh 这个脚本可以用来删除指定位置前的消息。当一个分区被创建的时候，它的起始位置（logStartOffset）为0。我们可以通过 KafkaConsumer 中的 beginningOffsets() 方法来查看分区的起始位置，参考代码清单27-1：

```
//代码清单27-1 查看分区起始位置
KafkaConsumer<String, String> kafkaConsumer =
createNewConsumer();
List<PartitionInfo> partitions =
kafkaConsumer.partitionsFor("topic-monitor");
List<TopicPartition> tpList = partitions.stream()
        .map(pInfo -> new
TopicPartition(pInfo.topic(), pInfo.partition()))
        .collect(toList());
Map<TopicPartition, Long> beginningOffsets =
        kafkaConsumer.beginningOffsets(tpList);
System.out.println(beginningOffsets);
```

输出结果如下：

```
{topic-monitor-0=0, topic-monitor-1=0, topic-
monitor-2=0}
```

下面使用 kafka-delete-records.sh 脚本来删除部分消息。在执行具体的删除动作之前需要先配置一个 JSON 文件，用来指定所要删除消息的分区及对应的位置。我们需要分别删除主题 topic-monitor

下分区0中偏移量为10、分区1中偏移量为11和分区2中偏移量为12的消息：

```
{
    "partitions": [
        {
            "topic": "topic-monitor",
            "partition": 0,
            "offset": 10
        },
                {
            "topic": "topic-monitor",
            "partition": 1,
            "offset": 11
        },
                {
            "topic": "topic-monitor",
            "partition": 2,
            "offset": 12
        }
    ],
    "version": 1
}
```

之后将这段内容保存到文件中，比如取名为 delete.json，在此之后，我们就可以通过 kafka-delete-records.sh 脚本中的 offset-json-file 参数来指定这个 JSON 文件。具体的删除操作如下：

```
[root@node1 kafka_2.11-2.0.0]# bin/kafka-delete-
records.sh --bootstrap-server localhost:9092 --
offset-json-file delete.json
Executing records delete operation
Records delete operation completed:
partition: topic-monitor-0  low_watermark: 10
partition: topic-monitor-1  low_watermark: 11
partition: topic-monitor-2  low_watermark: 12
```

我们再次执行代码清单27-1，可以发现最后的运行结果已经变为：

```
{topic-monitor-0=10, topic-monitor-1=11, topic-
monitor-2=12}
```

kafka-delete-records.sh 脚本内部是通过调用 KafkaAdminClient 中的 deleteRecords() 方法来实现的，这个方法的具体定义如下所示。

```
public DeleteRecordsResult deleteRecords(
        Map<TopicPartition, RecordsToDelete>
recordsToDelete)
```

deleteRecords() 方法最终还需要通过发送 DeleteRecordsRequest 请求来通知 Kafka 完成相应的“删除”动作。其实 Kafka 并不会直接删除消息，它在收到 DeleteRecordsRequest 请求之后，会将指定分区的 logStartOffset 置为相应的请求值（比如分区0的偏移量10），最终的删除消息的动作还是交由日志删除任务来完成的。