http://www.cnblogs.com/huxi2b/p/6061110.html

众所周知，由于Zookeeper并不适合大批量的频繁写入操作，新版Kafka已推荐将consumer的位移信息保存在Kafka内部的topic中，即\_\_consumer\_offsets topic，并且默认提供了kafka\_consumer\_groups.sh脚本供用户查看consumer信息。

　　不过依然有很多用户希望了解\_\_consumer\_offsets topic内部到底保存了什么信息，特别是想查询某些consumer group的位移是如何在该topic中保存的。针对这些问题，本文将结合一个实例探讨如何使用kafka-simple-consumer-shell脚本来查询该内部topic。

**1. 创建topic “test”**

bin/kafka-topics.sh --zookeeper localhost:2181 --create --topic test --replication-factor 3 --partitions 3

**2. 使用kafka-console-producer.sh脚本生产消息**

　　由于默认没有指定key，所以根据round-robin方式，消息分布到不同的分区上。 (**本例中生产了64条消息**)

**3. 验证消息生产成功**

bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092,localhost:9093,localhost:9094 --topic test --time -1

结果输出表明64条消息全部生产成功！

*test:2:21*

*test:1:21*

*test:0:22*

**4. 创建一个console consumer group**

bin/kafka-console-consumer.sh --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --topic test --from-beginning --new-consumer

**5. 获取该consumer group的group id(后面需要根据该id查询它的位移信息)**

bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --list --new-consumer

输出： console-consumer-46965  (记住这个id！)

**6. 查询\_\_consumer\_offsets topic所有内容**

**注意：运行下面命令前先要在consumer.properties中设置exclude.internal.topics=false**

0.11.0.0之前版本

bin/kafka-console-consumer.sh --topic **\_\_consumer\_offsets** --zookeeper localhost:2181 --formatter "kafka.coordinator.GroupMetadataManager\$OffsetsMessageFormatter" --consumer.config config/consumer.properties --from-beginning

**0.11.0.0之后版本(含)**

bin/kafka-console-consumer.sh --topic **\_\_consumer\_offsets** --zookeeper localhost:2181 --formatter "kafka.coordinator.group.GroupMetadataManager\$OffsetsMessageFormatter" --consumer.config config/consumer.properties --from-beginning

默认情况下\_\_consumer\_offsets有50个分区，如果你的系统中consumer group也很多的话，那么这个命令的输出结果会很多。

**7. 计算指定consumer group在\_\_consumer\_offsets topic中分区信息**

这时候就用到了第5步获取的group.id(本例中是console-consumer-46965)。Kafka会使用下面公式计算该group位移保存在\_\_consumer\_offsets的哪个分区上：

Math.abs(groupID.hashCode()) % numPartitions

所以在本例中，对应的分区=Math.abs("console-consumer-46965".hashCode()) % 50 = 11，即\_\_consumer\_offsets的分区11保存了这个consumer group的位移信息，下面让我们验证一下。

**8. 获取指定consumer group的位移信息**

**0.11.0.0版本之前**

bin/kafka-simple-consumer-shell.sh --topic \_\_consumer\_offsets **--partition 11** --broker-list localhost:9092,localhost:9093,localhost:9094 --formatter "kafka.coordinator.GroupMetadataManager\$OffsetsMessageFormatter"

**0.11.0.0版本以后(含)**

bin/kafka-simple-consumer-shell.sh --topic \_\_consumer\_offsets **--partition 11** --broker-list localhost:9092,localhost:9093,localhost:9094 --formatter "kafka.coordinator.group.GroupMetadataManager\$OffsetsMessageFormatter"

下面是输出结果：

[复制代码](javascript:void(0);)

...

[console-consumer-46965,test,2]::[OffsetMetadata[21,NO\_METADATA],CommitTime 1479092279434,ExpirationTime 1479178679434]

[console-consumer-46965,test,1]::[OffsetMetadata[21,NO\_METADATA],CommitTime 1479092284246,ExpirationTime 1479178684246]

[console-consumer-46965,test,0]::[OffsetMetadata[22,NO\_METADATA],CommitTime 1479092284246,ExpirationTime 1479178684246]

[console-consumer-46965,test,2]::[OffsetMetadata[21,NO\_METADATA],CommitTime 1479092284246,ExpirationTime 1479178684246]

[**console-consumer-46965,test,1]::[OffsetMetadata[21**,NO\_METADATA],CommitTime 1479092284436,ExpirationTime 1479178684436]

[**console-consumer-46965,test,0]::[OffsetMetadata[22**,NO\_METADATA],CommitTime 1479092284436,ExpirationTime 1479178684436]

[**console-consumer-46965,test,2]::[OffsetMetadata[21**,NO\_METADATA],CommitTime 1479092284436,ExpirationTime 1479178684436]

...

[复制代码](javascript:void(0);)

　　上图可见，该consumer group果然保存在分区11上，且位移信息都是对的(这里的位移信息是已消费的位移，严格来说不是第3步中的位移。由于我的consumer已经消费完了所有的消息，所以这里的位移与第3步中的位移相同)。另外，可以看到\_\_consumer\_offsets topic的每一日志项的格式都是：[Group, Topic, Partition]::[OffsetMetadata[Offset, Metadata], CommitTime, ExpirationTime]