Geetha Madhuri Chittuluri

DataEng: Data Transport Activity

[this lab activity references tutorials at confluence.com]

A. Initialization

1. Get your cloud.google.com account up and running
    a. Redeem your GCP coupon - **Redeemed**
    b. Login to your GCP console
    c. Create a new, separate VM instance – **created new instance 'kafka'**

2. Follow the Kafka tutorial from project assignment #1
    a. Create a separate topic for this in-class activity – created 'transport' topic
    b. Make it "small" as you will not want to use many resources for this activity. By "small" I mean that you should choose medium or minimal options when asked for any configuration decisions about the topic, cluster, partitions, storage, anything. GCP/Confluent will ask you to choose the configs, and because you are using a free account you should opt for limited resources where possible. - **done**
    c. Get a basic producer and consumer working with a Kafka topic as described in the tutorials. – **done**

3. Create a sample breadcrumb data file (named bcsample.json) consisting of a sample of 1000 breadcrumb records. These can be any records because we will not be concerned with the actual contents of the breadcrumb records during this assignment.
   Created bcsample.json file with 1000 records and uploaded to git repository

4. Update your producer to parse your sample.json file and send its contents, one record at a time, to the kafka topic.

   Updated producer file and written messages to kafka topic – 'transport'

5. Use your consumer.py program (from the tutorial) to consume your records.

   Consumed records

B. Kafka Monitoring

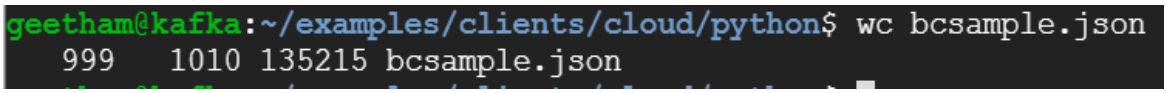
1. Find the Kafka monitoring console for your topic. Briefly describe its contents. Do the measured values seem reasonable to you?

   Yes, Kafka monitoring console is in Confluent cloud. Each topic has production and consumption and Consumer lag tabs which has a throughput metric displayed in bytes/second

2. Use this monitoring feature as you do each of the following exercises.

C. Kafka Storage

1. Run the linux command “wc bcsample.json”. Record the output here so that we can verify that your sample data file is of reasonable size.



2. What happens if you run your consumer multiple times while only running the producer once?

   Once consumer reads the produced messages, it won’t be reading again and again, it just waits for message or event in poll

3. Before the consumer runs, where might the data go, where might it be stored?

   Data is stored in the ‘topic’ till the consumer runs

4. Is there a way to determine how much data Kafka/Confluent is storing for your topic? Do the Confluent monitoring tools help with this?

   Yes, In confluent cloud, there is a throughput metric which bytes produces and consumed.

5. Create a “topic_clean.py” consumer that reads and discards all records for a given topic. This type of program can be very useful during debugging.

   created

D. Multiple Producers

1. Clear all data from the topic
2. Run two versions of your producer concurrently, have each of them send all 1000 of your sample records. When finished, run your consumer once. Describe the results.

   Consumer reads 2000 records produced by both producers concurrently 1000 records from each.

E. Multiple Concurrent Producers and Consumers

1. Clear all data from the topic
2. Update your Producer code to include a 250 msec sleep after each send of a message to the topic.
3. Run two or three concurrent producers and two concurrent consumers all at the same time.
4. Describe the results.

   I see producer is writing message with 0.25sec delay and two concurrent consumers consumers also consuming the messages

F. Varying Keys

1. Clear all data from the topic

So far you have kept the “key” value constant for each record sent on a topic. But keys can be very useful to choose specific records from a stream.

2. Update your producer code to choose a random number between 1 and 5 for each record’s key.
3. Modify your consumer to consume only records with a specific key (or subset of keys).
4. Attempt to consume records with a key that does not exist. E.g., consume records with key value of “100”. Describe the results
5. Can you create a consumer that only consumes specific keys? If you run this consumer multiple times with varying keys then does it allow you to consume messages out of order while maintaining order within each key?

G. Producer Flush

The provided tutorial producer program calls “producer.flush()” at the very end, and presumably your new producer also calls producer.flush().

1. What does Producer.flush() do?

   Wait for all messages in the Producer queue to be delivered. Also flush() will block until the previously sent messages have been delivered which makes the producer synchronous.

2. What happens if you do not call producer.flush()?

   Total number of messages might be lesser than the actual number of messages produced.

3. What happens if you call producer.flush() after sending each record?

   After sending each record, calling producer.flush() will effectively implement synchronous producer.

4. What happens if you wait for 2 seconds after every 5th record send, and you call flush only after every 15 record sends, and you have a consumer running concurrently? Specifically, does the consumer receive each message immediately? only after a flush? Something else?

   Consumer consumed each message after it gets flushed and displayed in a group of 5 records after displaying 5records at a time it is waiting for message and then again displaying a group of 5 messages

H. Consumer Groups

1. Create two consumer groups with one consumer program instance in each group.
2. Run the producer and have it produce all 1000 messages from your sample file.
3. Run each of the consumers and verify that each consumer consumes all of the 50 messages.
4. Create a second consumer within one of the groups so that you now have three consumers total.
5. Rerun the producer and consumers. Verify that each consumer group consumes the full set of messages but that each consumer within a consumer group only consumes a portion of the messages sent to the topic.

Geetha Madhuri Chittuluri

I. Kafka Transactions

6. Create a new producer, similar to the previous producer, that uses transactions.
7. The producer should begin a transaction, send 4 records in the transactions, then wait for 2 seconds, then choose True/False randomly with equal probability. If True then finish the transaction successfully with a commit. If False is picked then cancel the transaction.
8. Create a new transaction-aware consumer. The consumer should consume the data. It should also use the Confluent/Kaka transaction API with a “read_committed” isolation level. (I can’t find evidence of other isolation levels).
9. Transaction across multiple topics. Create a second topic and modify your producer to send two records to the first topic and two records to the second topic before randomly committing or canceling the transaction. Modify the consumer to consume from the two queues. Verify that it only consumes committed data and not uncommitted or canceled data.