APACHE KAFKA

Rajesh Pasham

Apache Kafka - Basic Operations

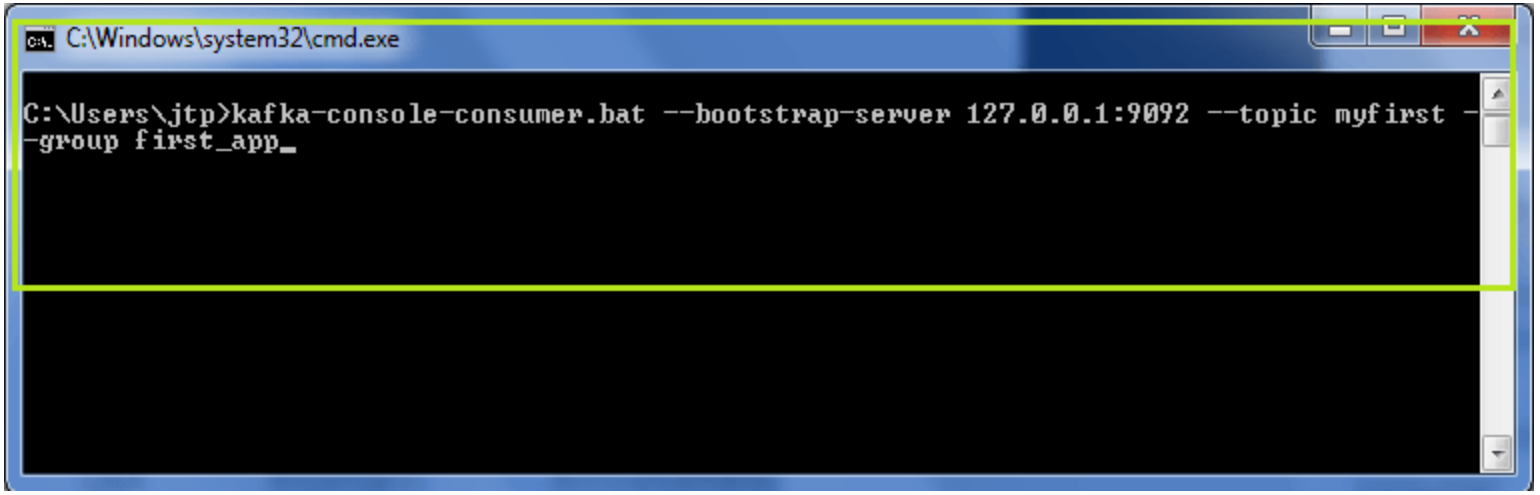
Kafka Consumer Group

- Generally, a Kafka consumer belongs to a particular consumer group.
- A consumer group basically represents the name of an application.
- In order to consume messages in a consumer group, '**-group**' command is used.

Apache Kafka - Basic Operations

- Use the '**-group**' command as:
  - **'kafka-console-consumer --bootstrap-server localhost:9092 --topic <topic_name> --group <group_name>'**.
  - Give some name to the group.

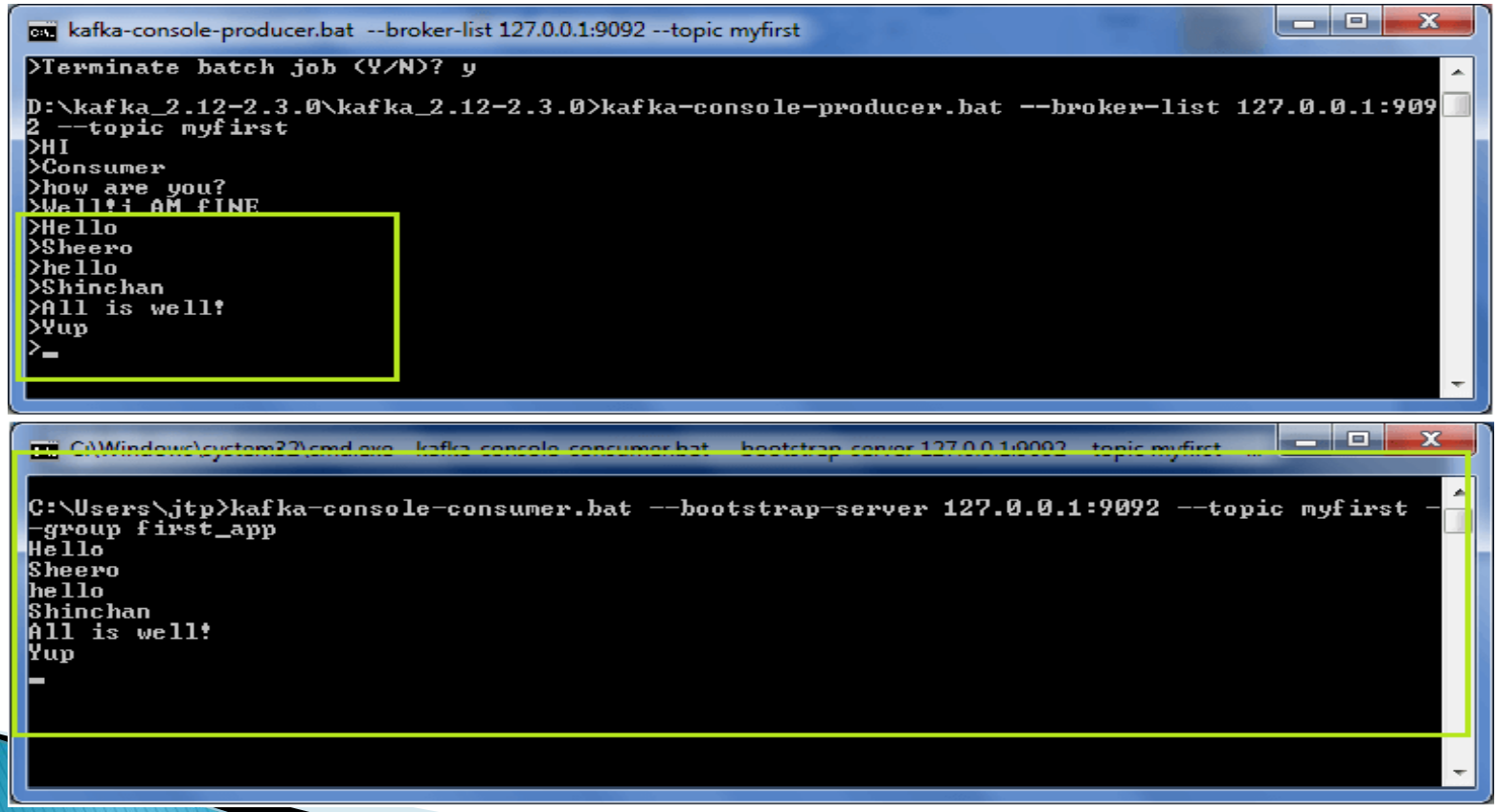
Kafka Consumer Group



- In the above snapshot, the name of the group is '**first_app**'.
- It is seen that no messages are displayed because no new messages were produced to this topic.
- If '**--from-beginning**' command will be used, all the previous messages will be displayed.

Kafka Consumer Group

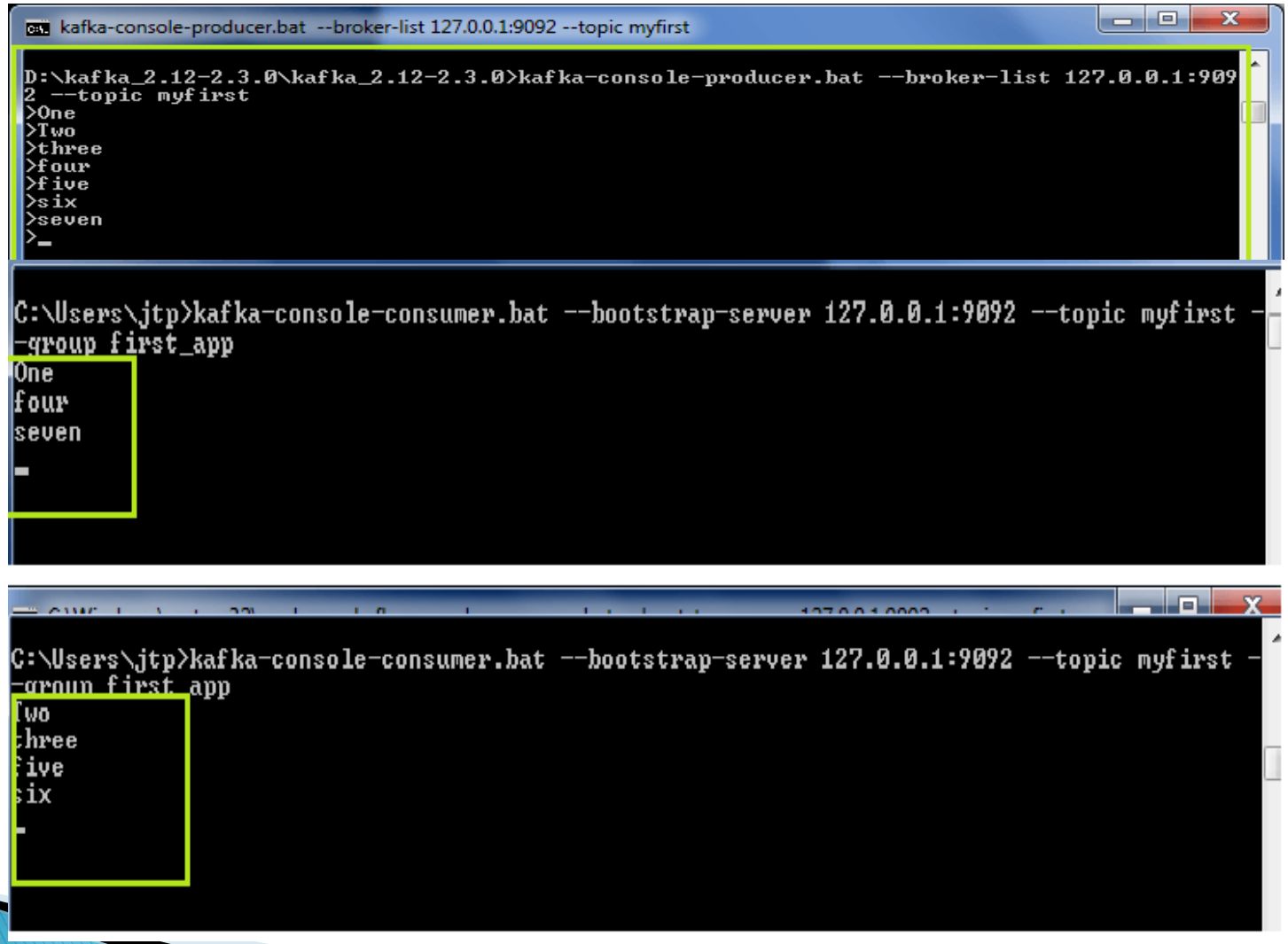
- To view some new messages, produce some instant messages from the producer console.



Kafka Consumer Group

- But, it was a single consumer reading data in the group.
- Let's create more consumers to understand the power of a consumer group.
- For that, open a new terminal and type the exact same consumer command as:
  - 'kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic <topic_name> --group <group_name>'.

Kafka Consumer Group



Kafka Consumer Group

- In the above snapshot, it is clear that the producer is sending data to the Kafka topics.
- The two consumers are consuming the messages.
- Look at the sequence of the messages.
- As there were three partitions created for 'myfirst' topic, so messages are split in that sequence only.

Kafka Consumer Group

- We can further create more consumers under the same group, and each consumer will consume the messages according to the number of partitions.
- Try yourself to understand better.
- The group id should be the same, then only the messages will be split between the consumers.

Kafka Consumer Group

- However, if any of the consumers is terminated, the partitions will be reassigned to the active consumers, and these active consumers will receive the messages.

- So, in this way, various consumers in a consumer group consume the messages from the Kafka topics.

Producer with Keys

- A Kafka producer can write data to the topic either with or without a key.
- If a producer does not specify a key, the data will be stored to any of the partitions with key=null, else the data will be stored to the specified partition only.
- A '**parse.key**' and a '**key.seperator**' is required to specify a key for the topic.

Producer with Keys

- The command used is:
  - 'kafka-console-producer --broker -list localhost:9092 --topic <topic_name> --property parse.key=**true** --property key.separator=,
  - >key,value
  - >another key,another value‘
- Here, key is the specific partition, and value is the message to be written by the producer to the topic.

Consumer with Keys

- When a producer has attached a key value with the data, it will get stored to that specified partition.
- If no key value is specified, the data will move to any partition.
- So, when a consumer reads the message with a key, it will be displayed null, if no key was specified.
- A '**print.key**' and a '**key.seperator**' sre required to consume messages from the Kafka topics.

Consumer with Keys

- The command used is:

- 'kafka-console-consumer --bootstrap-server localhost:9092 --topic <topic_name> --from-beginning --property print.key=true --property key.seperator=,'

- Using the above command, the consumer can read data with the specified keys.

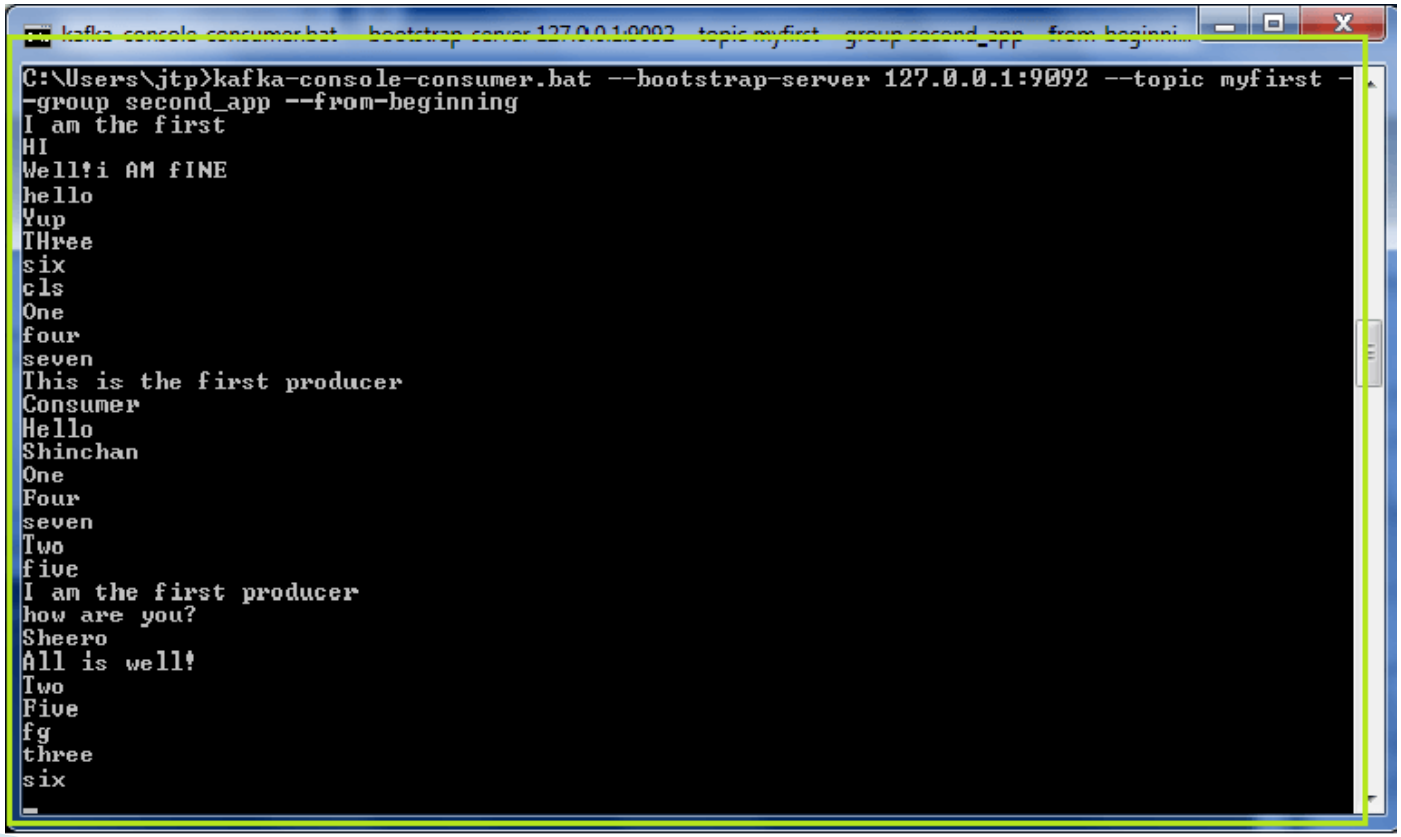
More about Consumer Group

'--from-beginning' command

- This command is used to read the messages from the starting(discussed earlier).
- Thus, using it in a consumer group will give the following output:

More about Consumer Group

'--from-beginning' command



More about Consumer Group

'--from-beginning' command

- It can be noticed that a new consumer group 'second_app' is used to read the messages from the beginning.
- If one more time the same command will run, it will not display any output.
- It is because offsets are committed in Apache Kafka.
- So, once a consumer group has read all the until written messages, next time, it will read the new messages only.

More about Consumer Group

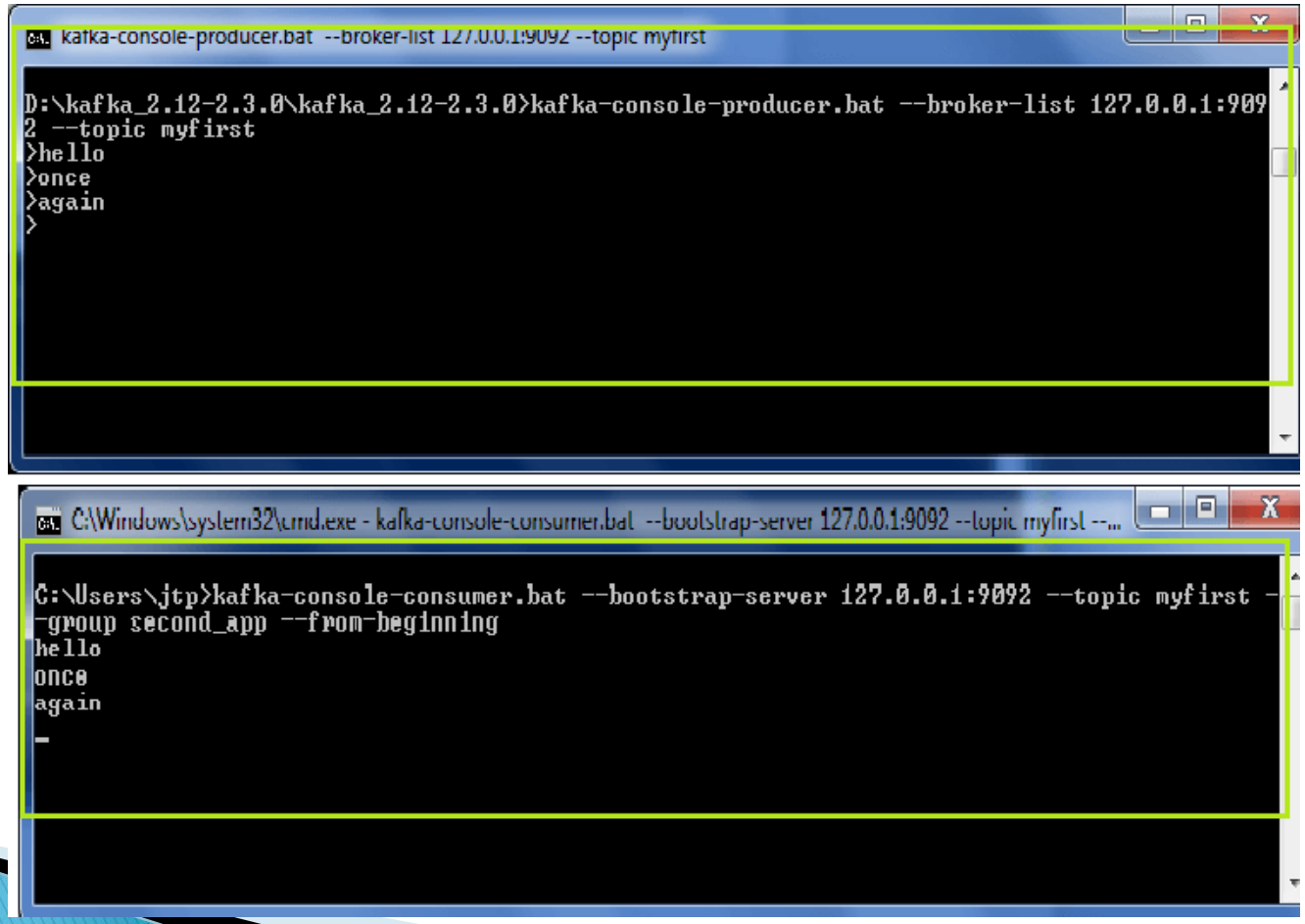
'--from-beginning' command

- For example, in the below snapshot, when '**--from-beginning**' command is used again, only the new messages are read.

- It is because all the previous messages were consumed earlier only.

More about Consumer Group

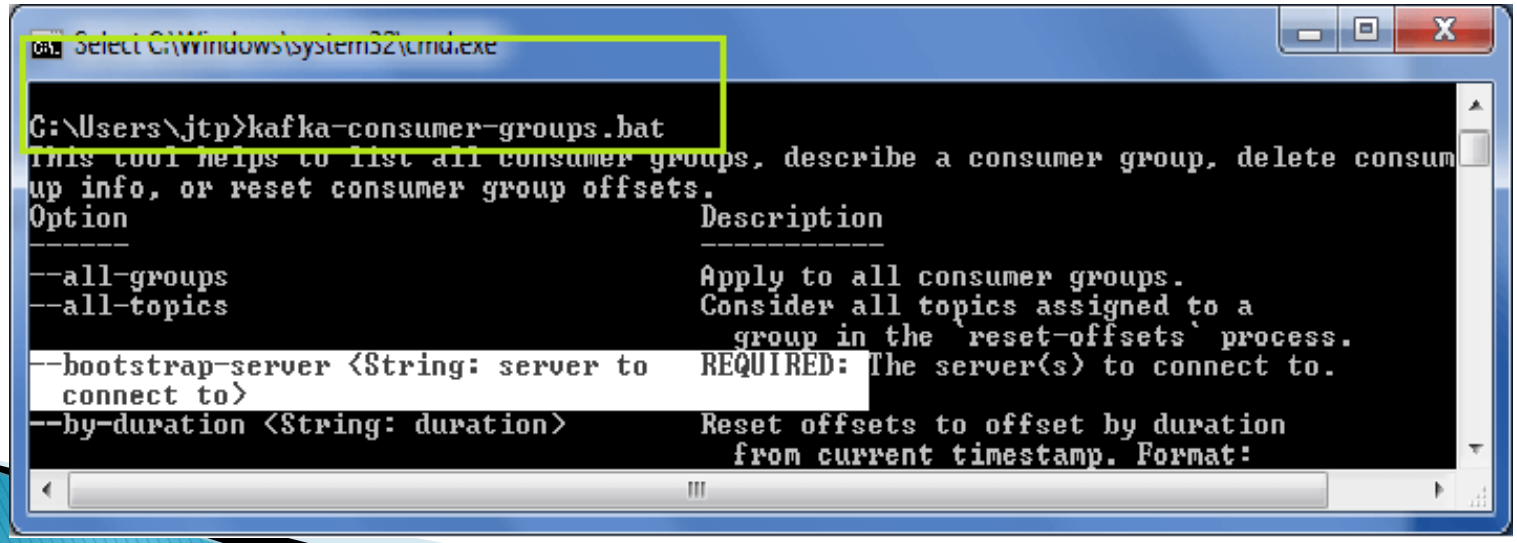
'--from-beginning' command



More about Consumer Group

'kafka-consumer-groups' command

- This command gives the whole documentation to list all the groups, describe the group, delete consumer info, or reset consumer group offsets.



More about Consumer Group

Listing Consumer Groups

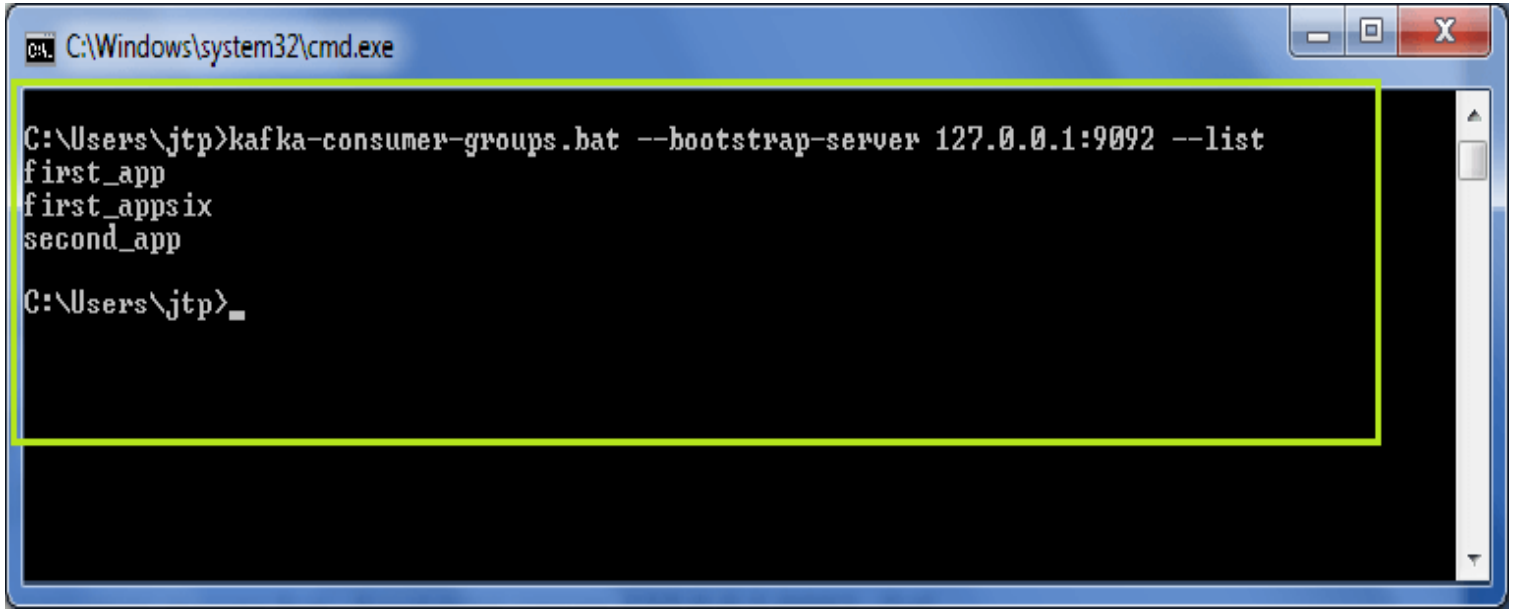
- A '--list' command is used to list the number of consumer groups available in the Kafka Cluster.

- The command is used as:
  - 'kafka-consumer-groups.bat --bootstrap-server localhost:9092 --list'.

More about Consumer Group

Listing Consumer Groups

- A snapshot is shown below, there are three consumer groups present.



More about Consumer Group

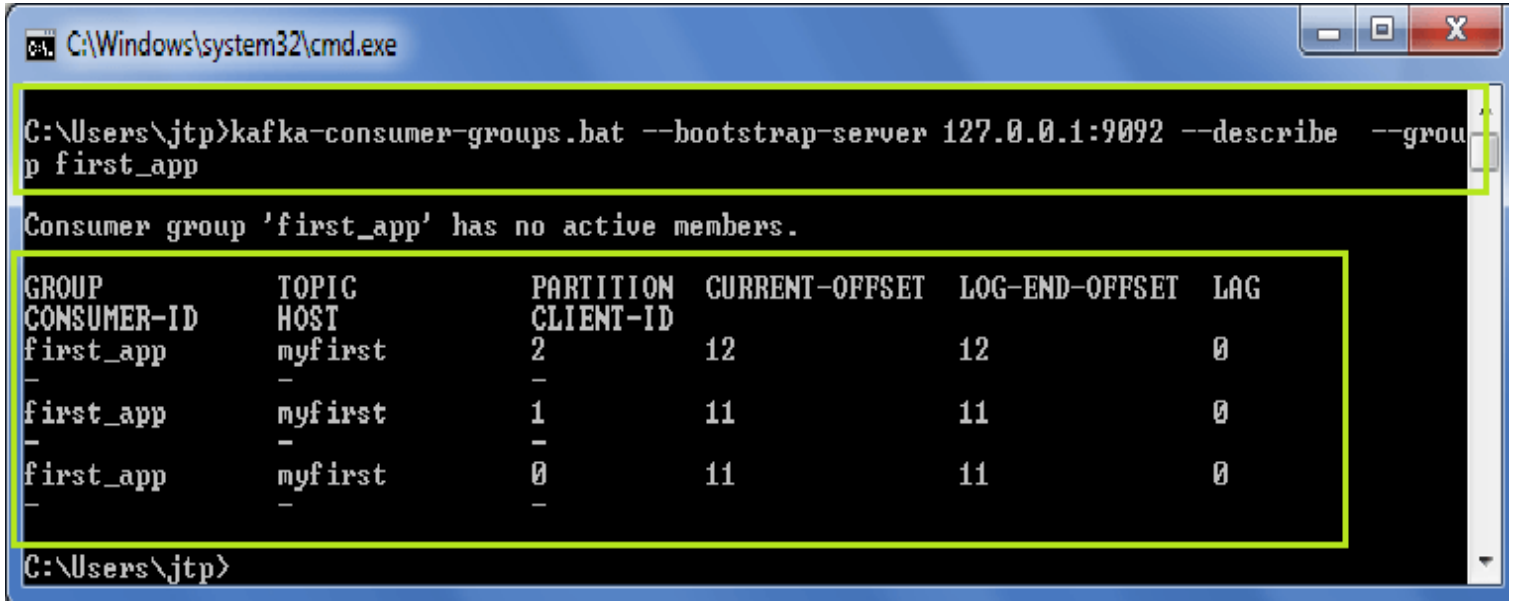
Describing a Consumer Group

- A '**--describe**' command is used to describe a consumer group.

- The command is used as:
  - **'kafka-consumer-groups.bat --bootstrap-server localhost:9092 --describe --group <group_name>'**

More about Consumer Group

Describing a Consumer Group



- This command describes whether any active consumer is present, the current offset value, lag value is 0 -indicates that the consumer has read all the data.

Resetting the Offsets

- Offsets are committed in Apache Kafka.
- Therefore, if a user wants to read the messages again, it is required to reset the offsets value.
- '**Kafka-consumer-groups**' command offers an option to reset the offsets.
- Resetting the offset value means defining the point from where the user wants to read the messages again.
- It supports only one consumer group at a time, and there should be no active instances for the group.

Resetting the Offsets

- While resetting the offsets, the user needs to choose three arguments:
  - An execution option
  - Reset Specifications
  - Scope

- There are two executions options available:
  - '--dry-run': It is the default execution option. This option is used to plan those offsets that need to be reset.
  - '**--execute':** This option is used to update the offset values.

Resetting the Offsets

- There are following reset specifications available:
  - '--to-datetime': It reset the offsets on the basis of the offset from datetime. The format used is: 'YYYY-MM-DDTHH:mm:SS.sss'.
  - '--to-earliest': It reset the offsets to the earliest offset.
  - '--to-latest': It reset the offsets to the latest offset.
  - '--shift-by': It reset the offsets by shifting the current offset value by 'n'. The value of 'n' can be positive or negative.
  - '--from-file': It resets the offsets to the values defined in the CSV file.
  - ' --to-current': It reset the offsets to the current offset.

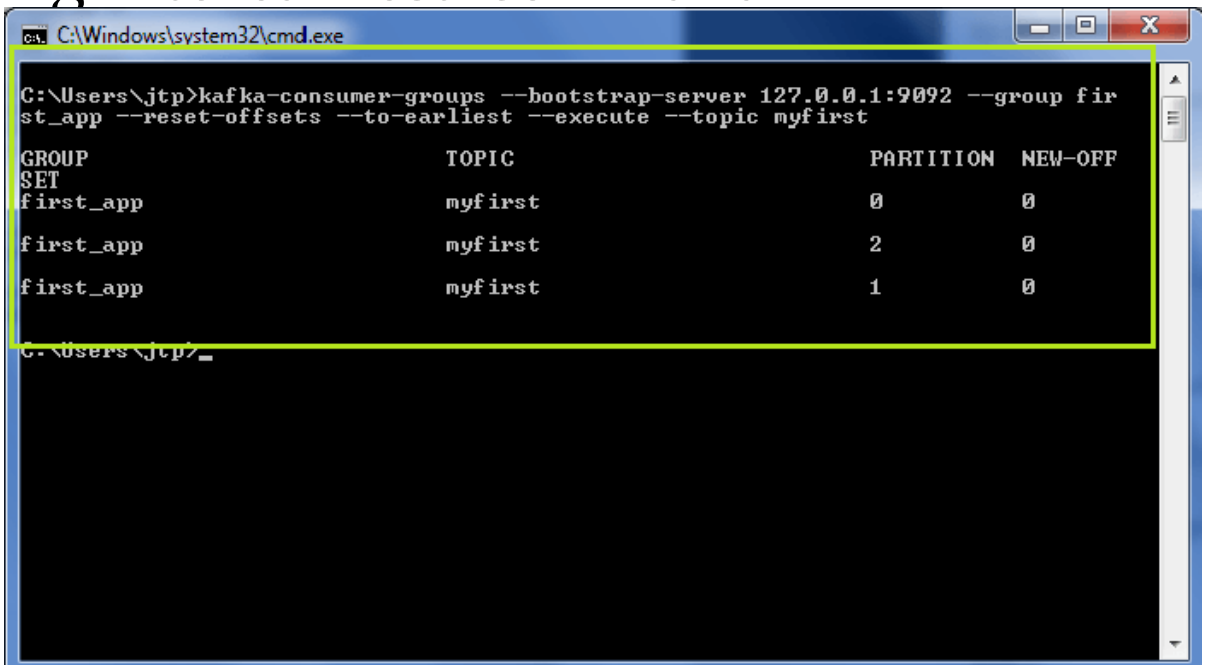
Resetting the Offsets

- There are two scopes available to define:
  - '--all-topics': It reset the offset value for all the available topics within a group.
  - '--topics': It reset the offset value for the specified topics only. The user needs to specify the topic name for resetting the offset value.

Resetting the Offsets

Let's try and see:

- 1) Using '--to-earliest' command

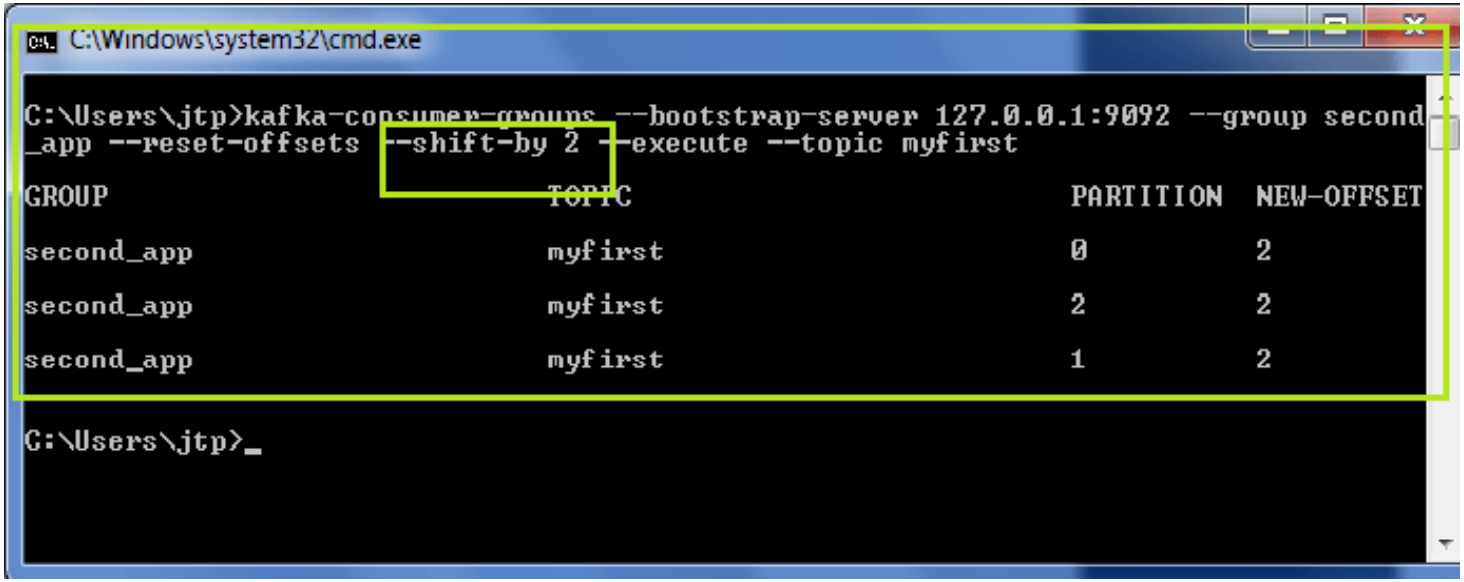


In the above snapshot, the offsets are reset to the new offset as 0. It is because '**--to-earliest**' command is used, which has reset the offset value to 0.

Resetting the Offsets

Let's try and see:

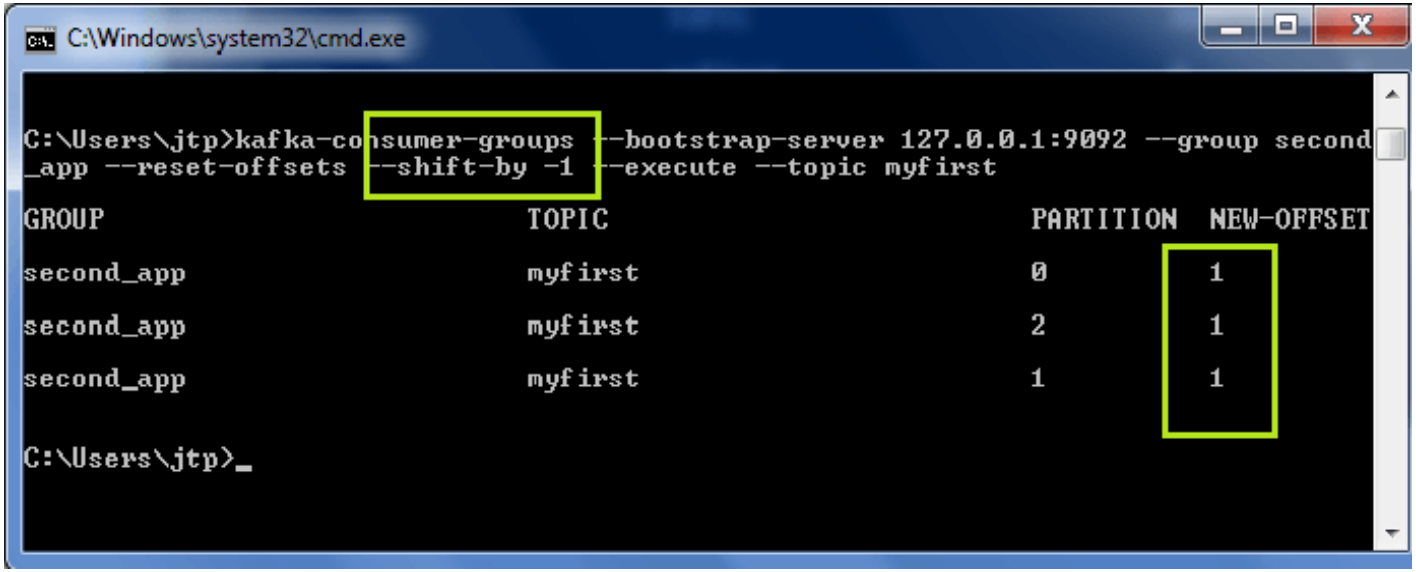
- 2) Using '--shift-by' command



Resetting the Offsets

Let's try and see:

- 2) Using '--shift-by' command



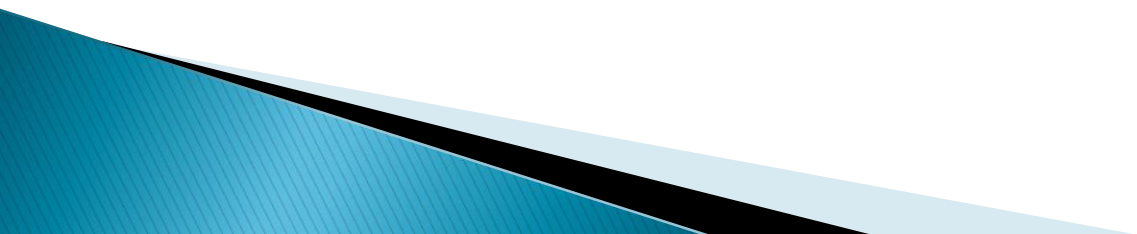
Resetting the Offsets

Let's try and see:

- 2) Using '--shift-by' command
  - In the first snapshot, the offset value is shifted from '0' to '+2'. In the second one, the offset value is shifted from '2' to '-1'.

Apache Kafka - Basic Operations

Exercise: 3 Kafka Producer, Consumer and Consumer Groups



Multiple node – multiple broker cluster

- As in the case of multiple-node Kafka cluster, where we set up multiple brokers on each node, we should install Kafka on each node of the cluster, and all the brokers from the different nodes need to connect to the same ZooKeeper.

- For testing purposes, all the commands will remain identical to the ones we used in the single node – multiple brokers cluster.

Multiple node – multiple broker cluster

- The diagram in the next slide shows the cluster scenario where multiple brokers are configured on multiple nodes (**Node 1** and **Node 2** in this case), and the producers and consumers are getting connected in different combinations:

Multiple node – multiple broker cluster