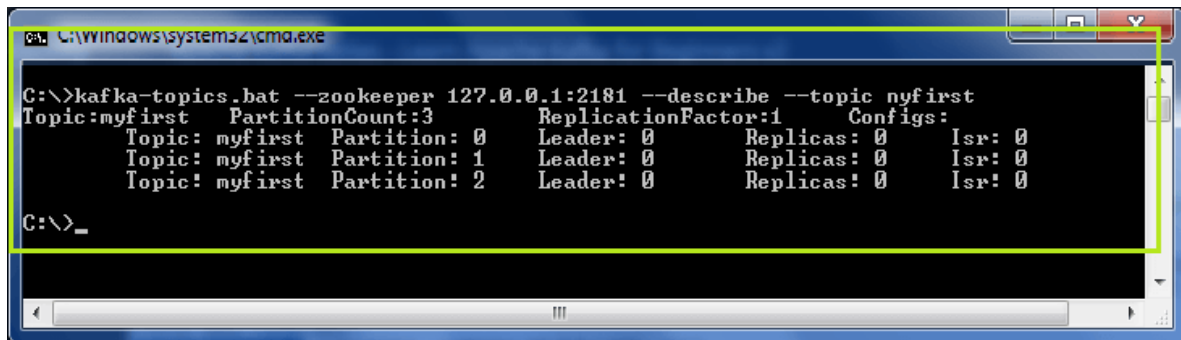


Exercise 2: Consumer Groups

Describing a topic

To describe a topic within the broker, use '**--describe**' command as:

'**kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic <topic_name>**'. This command gives the whole description of a topic with the number of partitions, leader, replicas and, ISR.



```
C:\>kafka-topics.bat --zookeeper 127.0.0.1:2181 --describe --topic myfirst
Topic:myfirst PartitionCount:3 ReplicationFactor:1 Configs:
Topic: myfirst Partition: 0 Leader: 0 Replicas: 0 Isr: 0
Topic: myfirst Partition: 1 Leader: 0 Replicas: 0 Isr: 0
Topic: myfirst Partition: 2 Leader: 0 Replicas: 0 Isr: 0
C:\>_
```

Sending data to Kafka Topics

Kafka Console Producer

Producer with Keys

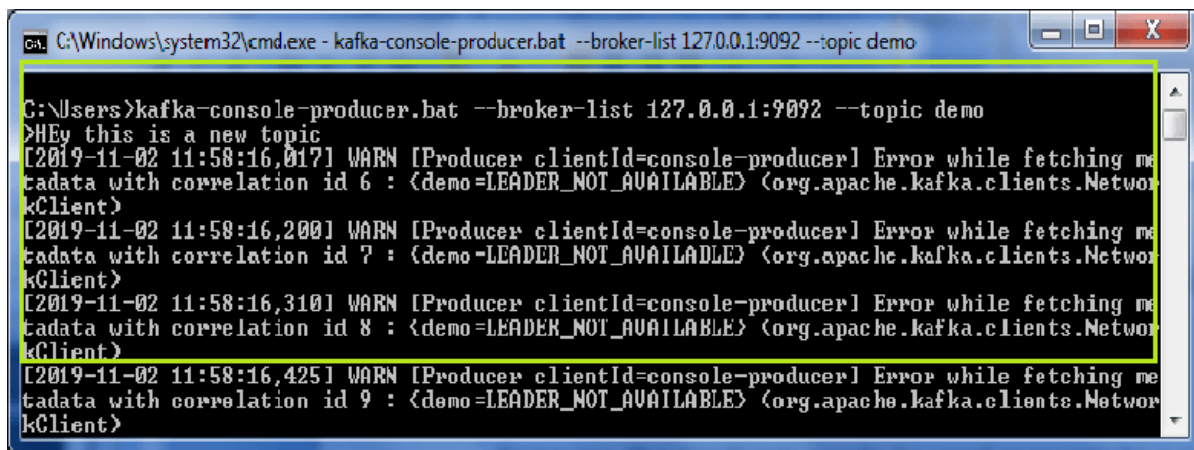
A Kafka producer can write data to the topic either with or without a key. If a producer does not specify a key, the data will be stored to any of the partitions with key=null, else the data will be stored to the specified partition only. A '**parse.key**' and a '**key.seperator**' is required to specify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

1. 'kafka-console-producer --bootstrap-server localhost:9092 --topic <topic_name> --property parse.key=**true** --property key.separator=,
2. > key,value
3. > another key,another value'

Here, key is the specific partition, and value is the message to be written by the producer to the topic.

When a topic does not exist?

Suppose the producer wants to send messages to a new topic that does not exist yet. In such a situation, a warning will appear, as shown in the below snapshot, after producing a message. It is just a warning.



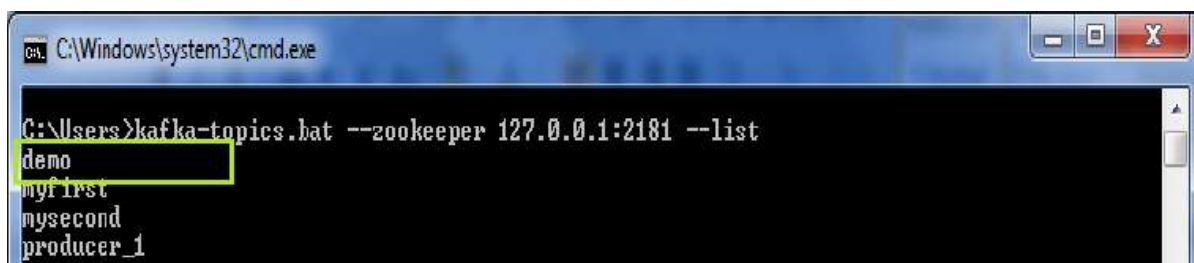
```
C:\Windows\system32\cmd.exe - kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic demo

C:\Users>kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic demo
>Hey this is a new topic
[2019-11-02 11:58:16,017] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 6 : {demo=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-11-02 11:58:16,200] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 7 : {demo=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-11-02 11:58:16,310] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 8 : {demo=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
[2019-11-02 11:58:16,425] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 9 : {demo=LEADER_NOT_AVAILABLE} (org.apache.kafka.clients.NetworkClient)
```

Why this warning?

The warning occurred because earlier the topic 'demo' didn't exist. But, as soon the producer wrote a message, Kafka somehow created that topic. Although, no leader election held for this unexpected topic, '**LEADER_NOT_AVAILABLE**' error could be seen. But, for the next time, the producer can continue to write more messages as no warning will appear again. It is because the topic comes in the existing list now.

The users can check using the '**--list**' command, as shown below:



```
C:\Windows\system32\cmd.exe

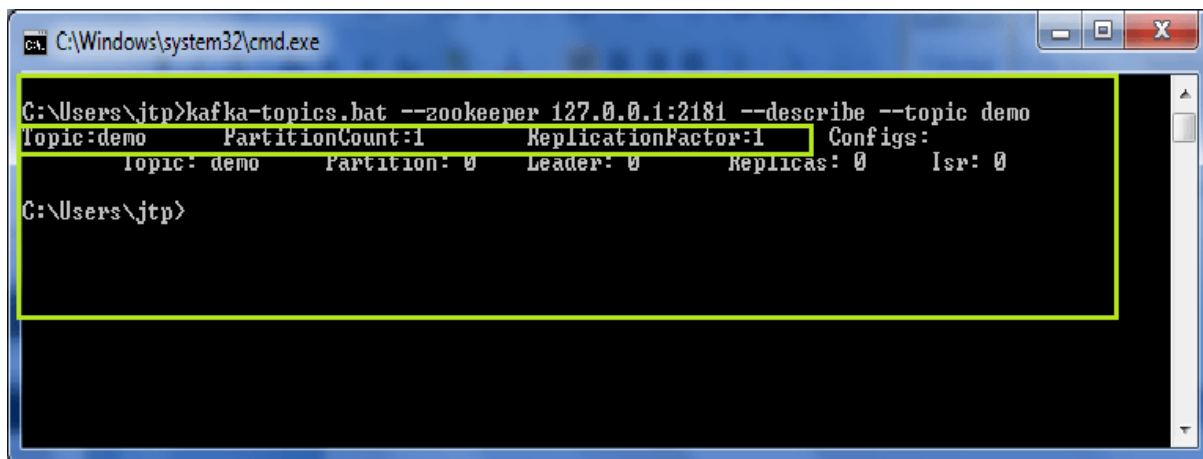
C:\Users>kafka-topics.bat --zookeeper 127.0.0.1:2181 --list
demo
myfirst
mysecond
producer_1
```

The topic 'demo' can be seen on the list.

Describing the new topic

As such topics which are created directly by the producer grabs the default number of partitions and its replication factor as 1.

For example,



```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-topics.bat --zookeeper 127.0.0.1:2181 --describe --topic demo
Topic:demo      PartitionCount:1      ReplicationFactor:1      Configs:
Topic: demo      Partition: 0      Leader: 0      Replicas: 0      Isr: 0

C:\Users\jtp>
```

The topic 'demo' when described using the '**--describe**' command, gives the value of 'PartitionCount' and 'ReplicationFactor' as 1(default value). Thus, it is always a better option to create a topic before producing messages to it.

Changing the Default Values

Follow the below steps to change the default values for the new topic:

1. Open '**server.properties**' file using Notepad++, or any other text editor.
2. Edit the value of num.partitions=1 to a new value. Let it be 3. So, whenever such new topics are introduced, the number of PartitionCount and ReplicationFactor will be 3(whatever the user has set).
3. Save the file.

But, always create topics before.

Kafka Console Consumer

Reading whole messages

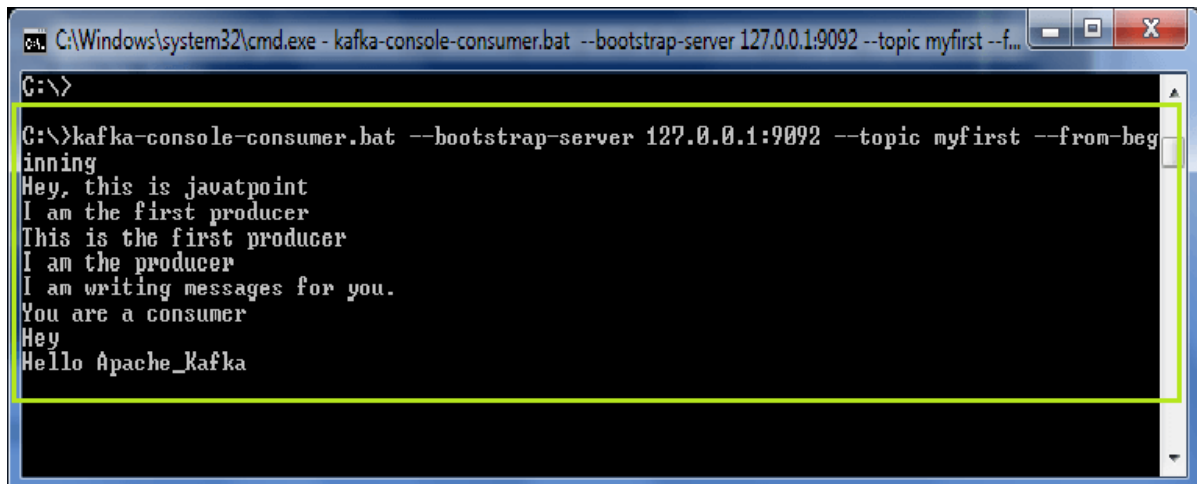
Apache Kafka allows to produce millions of messages. Sometimes, a consumer may require to read whole messages from a particular topic.

To do so, use '**--from-beginning**' command with the above kafka console consumer command as:

'kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --from-beginning'.

This command tells the Kafka topic to allow the consumer to read all the messages from the beginning(i.e., from the time when the consumer was inactive).

For example,

A screenshot of a Windows command prompt window. The title bar shows the command: `C:\Windows\system32\cmd.exe - kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --from-beg...`. The command prompt shows the command `kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --from-beg` being executed. The output messages are: `inning`, `Hey, this is javatpoint`, `I am the first producer`, `This is the first producer`, `I am the producer`, `I am writing messages for you.`, `You are a consumer`, `Hey`, and `Hello Apache_Kafka`.

In the above snapshot, it is clear that all messages are displayed from the beginning.

Note: The order of the messages is not the 'total'. It is because the sequence is at the partition level only(as studied in the Kafka introduction section).

For this topic 'myfirst', we had three partitions. So, if a user wishes to see the order, create a topic with a single partition value. It will display whole messages in a sequence.

After completing the message exchange process, press 'Ctrl+C' and stop.

So, several messages can be consumed either from the beginning or from that state when the user wants the consumer to read.

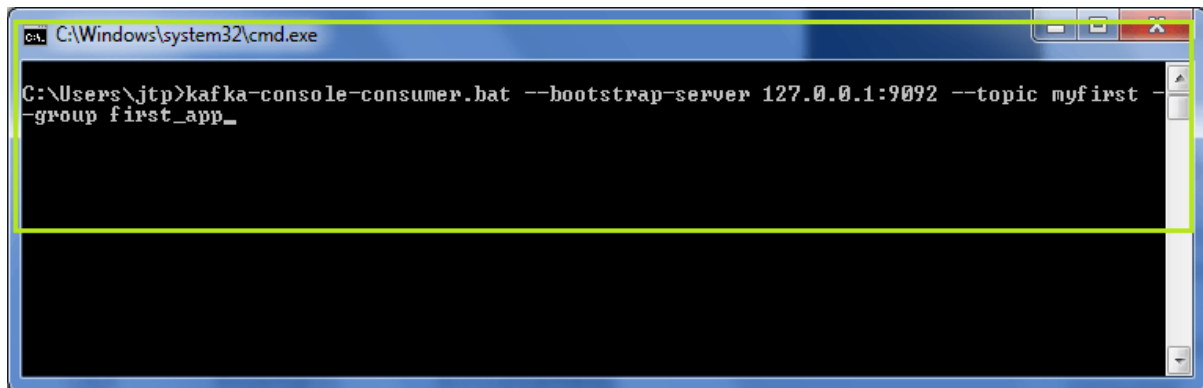
Kafka Consumer Group CLI

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.

Let's see how consumers will consume messages from Kafka topics:

Step1: Open the Windows command prompt.

Step2: Use the '**--group**' command as: '**kafka-console-consumer --bootstrap-server localhost:9092 --topic --group <group_name>**'. Give some name to the group. Press enter.

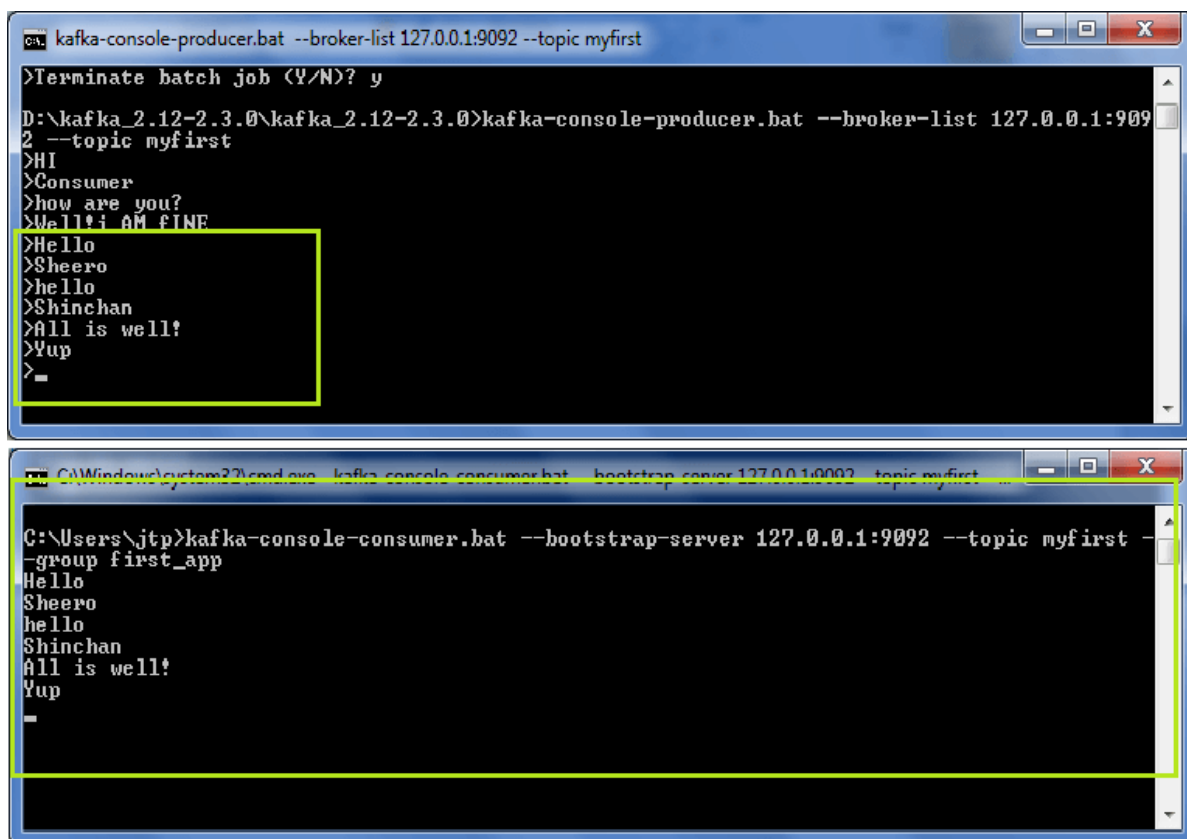


```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group first_app_
```

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.

Step3: To view some new messages, produce some instant messages from the producer console(as did in the previous section).



```
C:\Windows\system32\cmd.exe kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst

>Terminate batch job (Y/N)? y
D:\kafka_2.12-2.3.0\kafka_2.12-2.3.0>kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst
>HI
>Consumer
>how are you?
>Well! i AM FINE
>Hello
>Sheero
>hello
>Shinchan
>All is well!
>Yup
>_

C:\Windows\system32\cmd.exe kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group first_app
Hello
Sheero
hello
Shinchan
All is well!
Yup
_
```

So, the new messages produced by the producer can be seen in the consumer's console.

Step4: 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.bat --bootstrap-server 127.0.0.1:9092 --topic <topic_name> --group <group_name>**'.

```
cmd: kafka-console-producer.bat --broker-list 127.0.0.1:9092
D:\kafka_2.12-2.3.0\kafka_2.12-2.3.0>kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst
>One
>Two
>three
>four
>five
>six
>seven
>_
```

```
C:\Users\jtp>kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group first_app
One
four
seven
_
```

```
C:\Users\jtp>kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group first_app
Two
three
five
six
_
```

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 (discussed earlier), so messages are split in that sequence only.

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.

Note: The group id should be the same, then only the messages will be split between the consumers.

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.

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.separator**' are required to consume messages from the Kafka topics. The command used is:

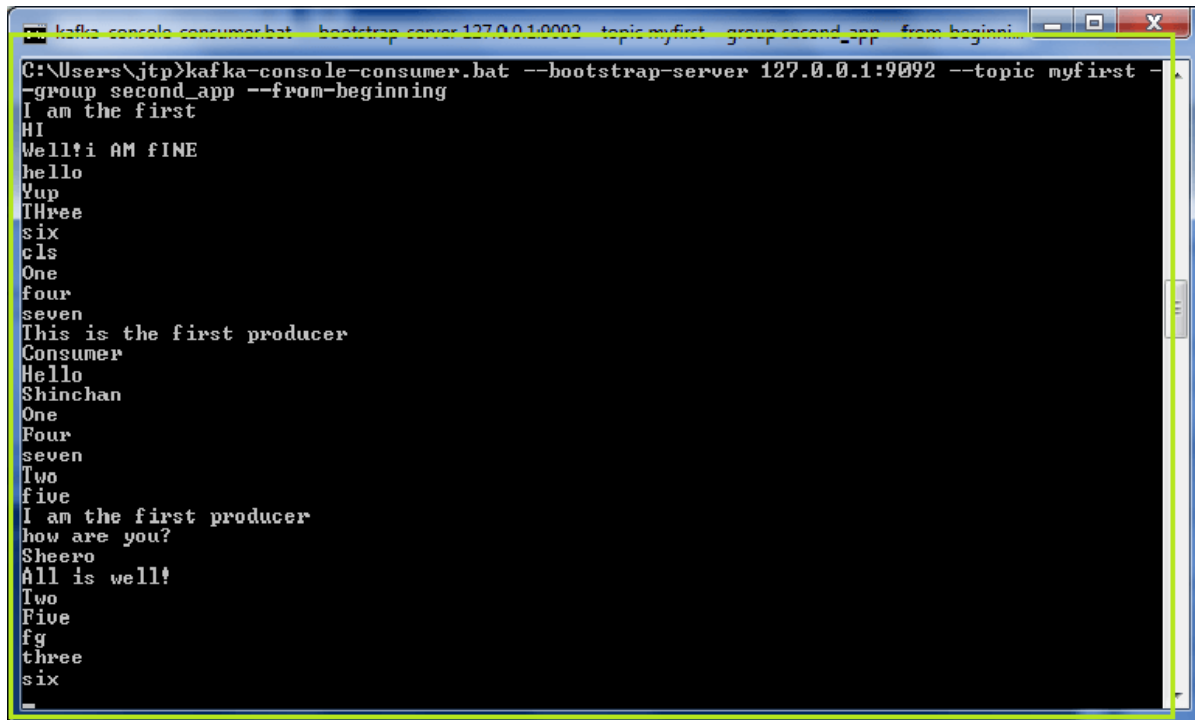
```
'kafka-console-consumer --bootstrap-server localhost:9092 --topic <topic_name> --from-beginning --property print.key=true --property key.separator=,'
```

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

More about Consumer Group

'--from-beginning' command

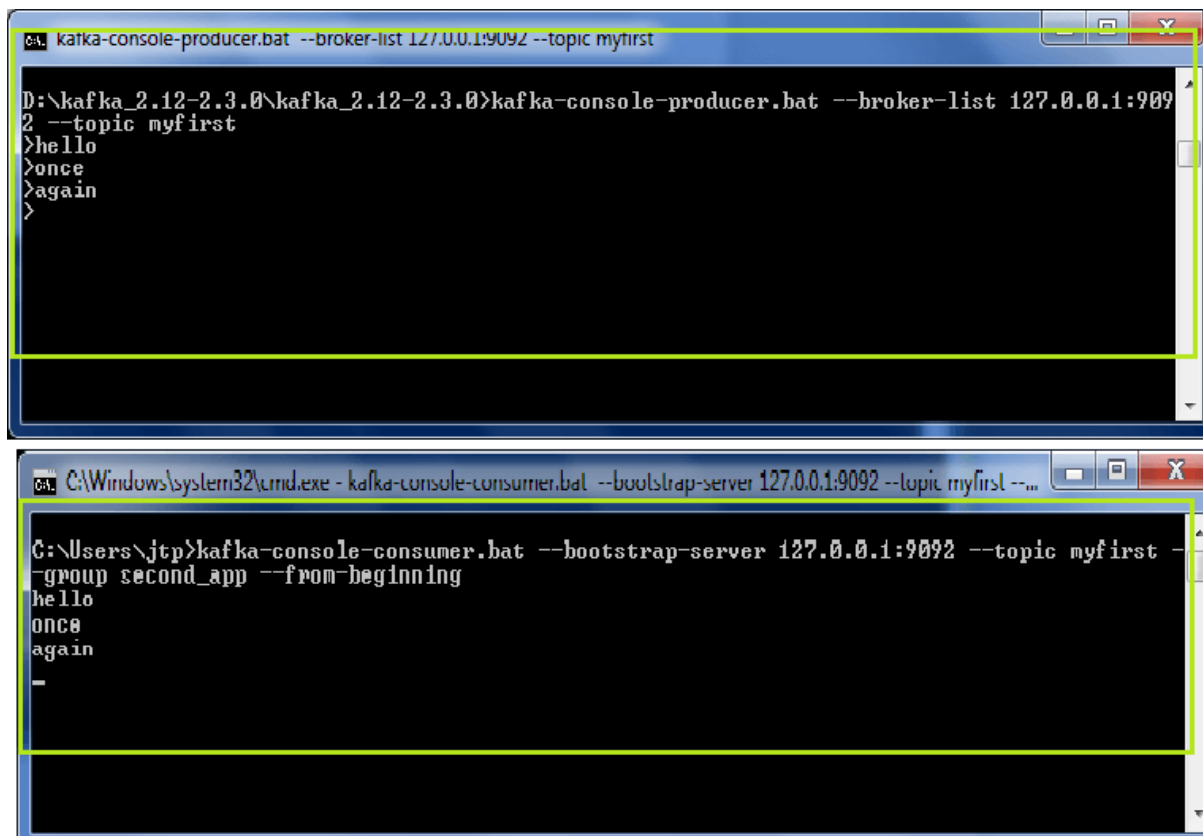
This command is used to read the messages from the starting(discussed earlier). Thus, using it in a consumer group will give the following output:



```
kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group second_app --from-beginning
C:\Users\jtp>kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group second_app --from-beginning
I am the first
HI
Well!i AM fine
hello
Vup
THree
six
cls
One
four
seven
This is the first producer
Consumer
Hello
Shinchan
One
Four
seven
Two
five
I am the first producer
how are you?
Sheero
All is well!
Two
Five
fg
three
six
```

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.

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.



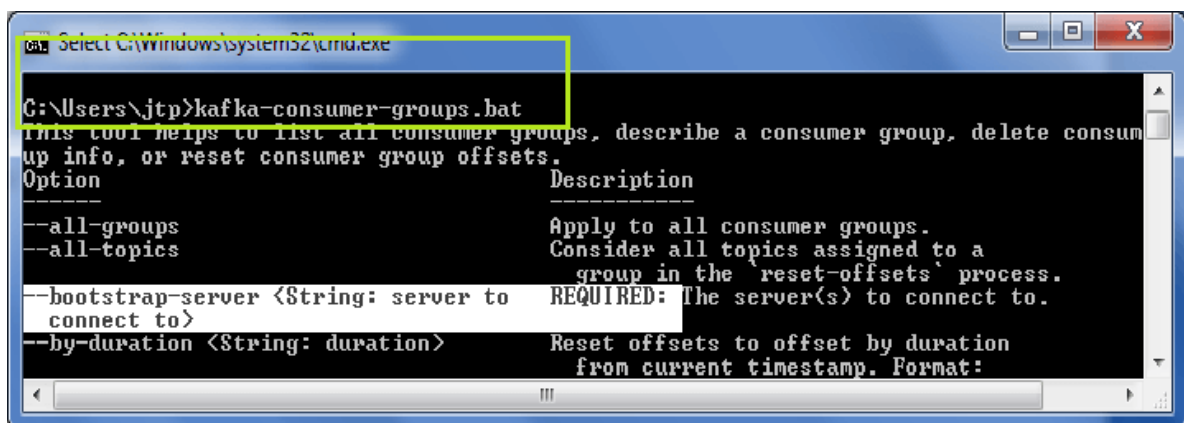
```
C:\> kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst

D:\kafka_2.12-2.3.0\kafka_2.12-2.3.0>kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst
>hello
>once
>again
>

C:\Windows\system32\cmd.exe - kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group second_app --from-beginning
hello
once
again
-
```

'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.



```
C:\Users\jtp>kafka-consumer-groups.bat
This tool helps to list all consumer groups, describe a consumer group, delete consumer info, or reset consumer group offsets.
Option      Description
-----
--all-groups      Apply to all consumer groups.
--all-topics      Consider all topics assigned to a group in the 'reset-offsets' process.
--bootstrap-server <String: server to connect to>  REQUIRED: The server(s) to connect to.
--by-duration <String: duration>  Reset offsets to offset by duration from current timestamp. Format:
```

It requires a bootstrap server for the clients to perform different functions on the 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'.

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


```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-consumer-groups.bat --bootstrap-server 127.0.0.1:9092 --list
first_app
first_appsix
second_app

C:\Users\jtp>_
```

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>'

```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-consumer-groups.bat --bootstrap-server 127.0.0.1:9092 --describe --group
p first_app

Consumer group 'first_app' has no active members.

GROUP      TOPIC      PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG
CONSUMER-ID HOST        CLIENT-ID
first_app  myfirst    2          12              12              0
-
first_app  myfirst    1          11              11              0
-
first_app  myfirst    0          11              11              0
-

C:\Users\jtp>
```

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.

While resetting the offsets, the user needs to choose three arguments:

1. An execution option
2. Reset Specifications
3. 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.

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.

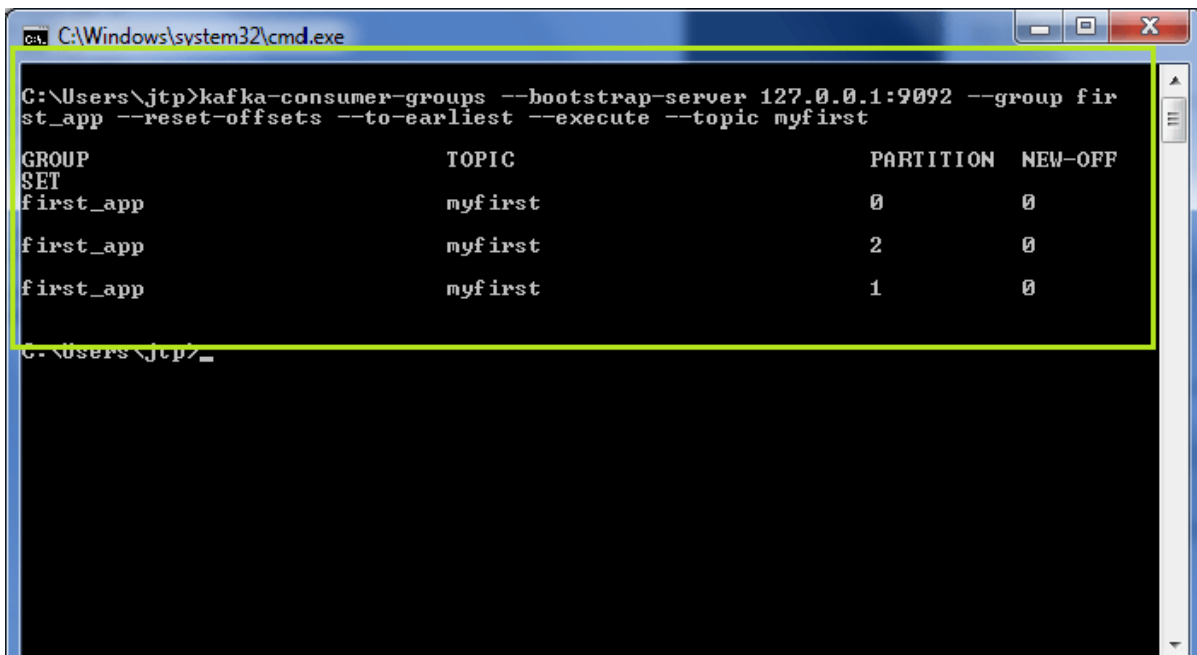
There are two scopes available to define:

'--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.

Let's try and see:

1) Using '--to-earliest' command



```
C:\Windows\system32\cmd.exe

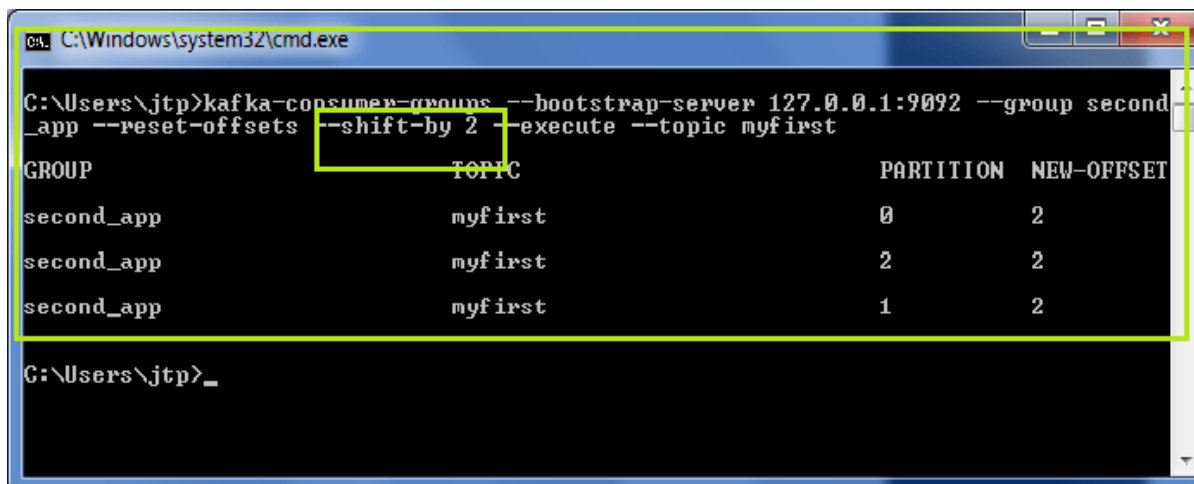
C:\Users\jtp>kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --group fir
st_app --reset-offsets --to-earliest --execute --topic myfirst

GROUP          TOPIC          PARTITION  NEW-OFF
SET
first_app      myfirst        0          0
first_app      myfirst        2          0
first_app      myfirst        1          0

C:\Users\jtp>_
```

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.

2) Using '--shift-by' command

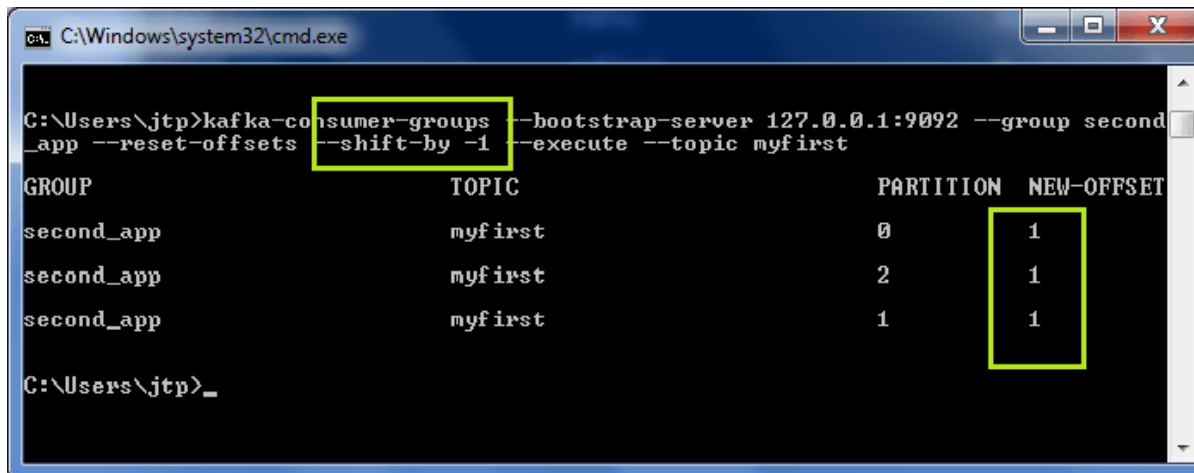


```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --group second_app --reset-offsets --shift-by 2 --execute --topic myfirst

GROUP          TOPIC          PARTITION  NEW-OFFSET
second_app     myfirst        0          2
second_app     myfirst        2          2
second_app     myfirst        1          2

C:\Users\jtp>_
```



```
C:\Windows\system32\cmd.exe

C:\Users\jtp>kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --group second_app --reset-offsets --shift-by -1 --execute --topic myfirst

GROUP          TOPIC          PARTITION  NEW-OFFSET
second_app     myfirst        0          1
second_app     myfirst        2          1
second_app     myfirst        1          1

C:\Users\jtp>_
```

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'.