Confluent KAFKA Administration

Rajesh Pasham

Apache Kafka - Basic Operations

- First let us start implementing single node-single broker configuration and we will then migrate our setup to single node-multiple brokers configuration.

- Before moving to the Kafka Cluster Setup, first you would need to start your ZooKeeper because Kafka Cluster uses ZooKeeper.

- Start ZooKeeper

```
$bin/zookeeper-server-start etc/kafka/zookeeper.properties
```

Apache Kafka - Basic Operations

- To start Kafka Broker, type the following command −

```
$bin/kafka-server-start etc/kafka/server.properties
```

- After starting Kafka Broker, type the command jps on terminal and you would see the following response −

```
821 QuorumPeerMain
928 Kafka
931 Jps
```

- Now you could see two daemons running on the terminal where QuorumPeerMain is ZooKeeper daemon and another one is Kafka daemon.

Apache Kafka - Basic Operations

Single Node-Single Broker Configuration

- In this configuration you have a single ZooKeeper and broker id instance.
- Following are the steps to configure it –
- Creating a Kafka Topic – Kafka provides a command line utility named ***kafka-topics*** to create topics on the server. Open new terminal and type the below example.

Apache Kafka - Basic Operations

Single Node-Single Broker Configuration

- Syntax

```
$bin/kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic topic-name
```

- Example

```
$bin/kafka-topics --create --bootstrap-server localhost:9092
  --replication-factor 1   --partitions 1 --topic Hello-Kafka
```

- We just created a topic named Hello-Kafka with a single partition and one replica factor.

Apache Kafka - Basic Operations

Single Node-Single Broker Configuration

- The above created output will be similar to the following output –

  Output – Created topic Hello-Kafka

- Once the topic has been created, you can get the notification in Kafka broker terminal window and the log for the created topic specified in “/tmp/kafka-logs/“ in the config/server.properties file.

Apache Kafka - Basic Operations

Topic already exists

- In case the user creates another topic with the same name as the existing topic, an error " **topic <topic_name> already exists**" will be thrown.

Apache Kafka - Basic Operations

List of Topics

- To get a list of topics in Kafka server, you can use the following command −

  Syntax : $bin/kafka-topics --list --bootstrap-server localhost:9092

- Output

```
        Hello-Kafka
```

Apache Kafka - Basic Operations

Describing a topic

- To describe a topic within the broker, use '--describe' command as:
  - 'kafka-topics --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.

Apache Kafka - Basic Operations

Start Producer to Send Messages

- Syntax : $bin/kafka-console-producer --broker-list localhost:9092 --topic topic-name
- From the above syntax, two main parameters are required for the producer command line client −
- Broker-list − The list of brokers that we want to send the messages to. In this case we only have one broker.

Apache Kafka - Basic Operations

Start Producer to Send Messages

- The etc/kafka/server.properties file contains broker port id, since we know our broker is listening on port 9092, so you can specify it directly.
- Topic name – Here is an example for the topic name.
- Example
  - $bin/kafka-console-producer --broker-list localhost:9092 --topic Hello-Kafka

Apache Kafka - Basic Operations

Start Producer to Send Messages

- The producer will wait on input from stdin and publishes to the Kafka cluster.
- By default, every new line is published as a new message then the default producer properties are specified in etc/kafka/producer.properties file.
- Now you can type a few lines of messages in the terminal as shown below.

Apache Kafka - Basic Operations

Start Producer to Send Messages

- Output

```
$ bin/kafka-console-producer --broker-list localhost:9092  --topic Hello-Kafka

[2016-01-16 13:50:45,931]  WARN property topic is not valid (kafka.utils.Verifia-bleProperties)
```

  - Hello
  - My first message
  - My second message

Apache Kafka - Basic Operations

Start Consumer to Receive Messages

- Similar to producer, the default consumer properties are specified in etc/kafka/consumer.properties file. Open a new terminal and type the below syntax for consuming messages.

- **Syntax:** $bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic topic-name --from-beginning

- **Example :** $bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic Hello-Kafka --from-beginning

Apache Kafka - Basic Operations

Start Consumer to Receive Messages

- **Output**
  - Hello
  - My first message
  - My second message

- Finally, you are able to enter messages from the producer's terminal and see them appearing in the consumer's terminal.

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>'.

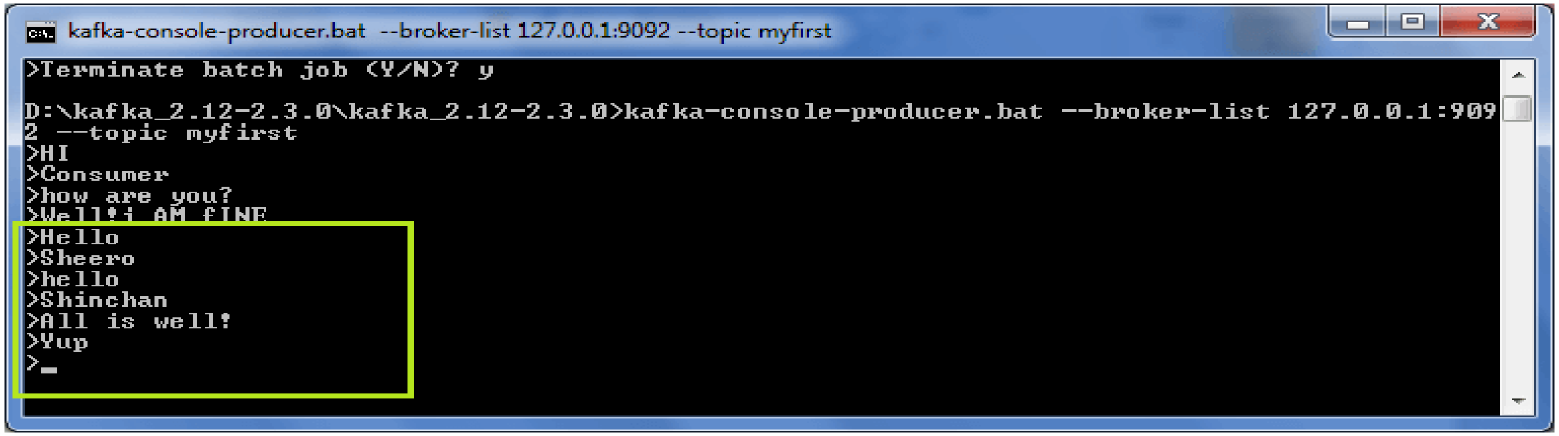
  - Give some name to the group.

  - Example: ‘kafka-console-consumer --bootstrap-server localhost:9092 --topic myfirst --group first_app

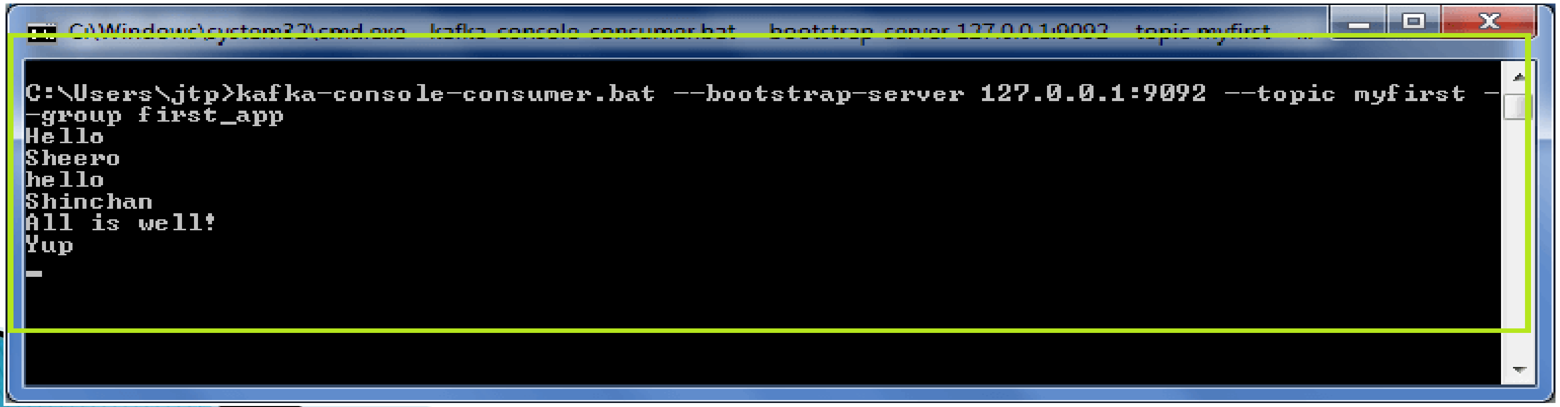
  - In the above example, the name of the group is “first_app”

Kafka Consumer Group

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



```
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 127.0.0.1:909
2 --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 ...

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

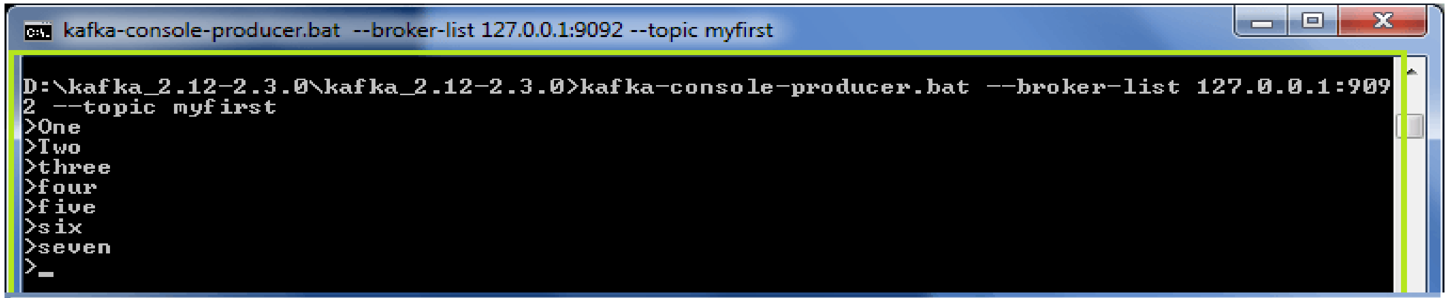
Kafka Consumer Group

- But, it was a single consumer reading data in the group.

- Let's create more consumers to understand the power of a consumer group.

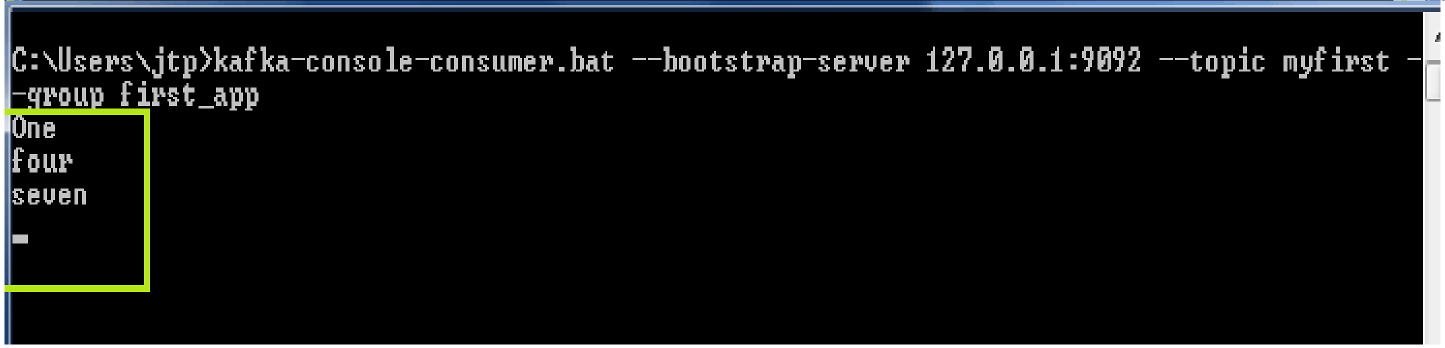
- For that, open a new terminal and type the exact same consumer command as:
  - 'kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic <topic_name> --group <group_name>'.

Kafka Consumer Group

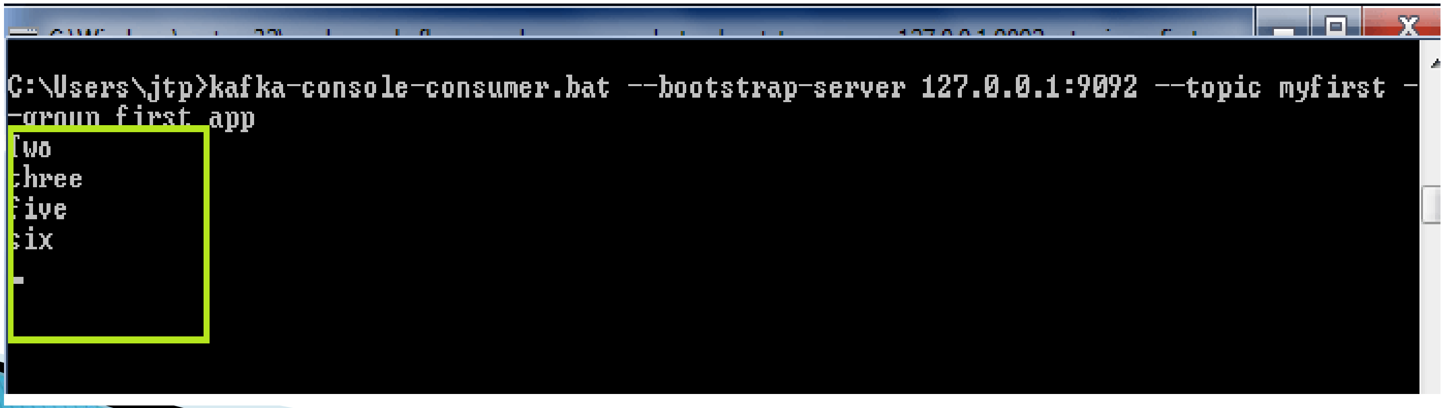


kafka-console-producer.bat --broker-list 127.0.0.1:9092 --topic myfirst

```
D:\kafka_2.12-2.3.0\kafka_2.12-2.3.0>kafka-console-producer.bat --broker-list 127.0.0.1:909
2 --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
```

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.

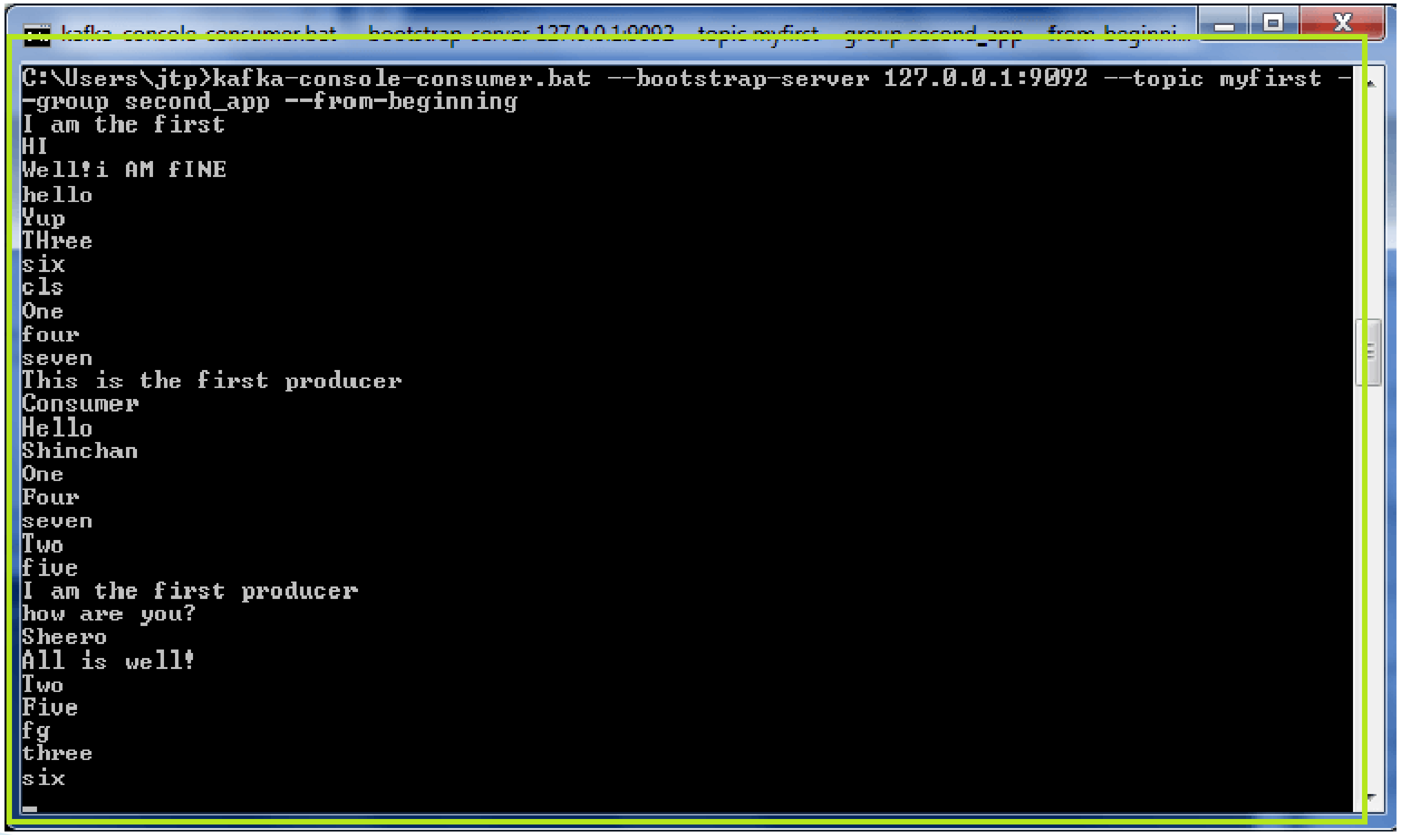
More about Consumer Group

'--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


kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --group second_app --from-beginni...

```
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
Yup
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
_
```

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

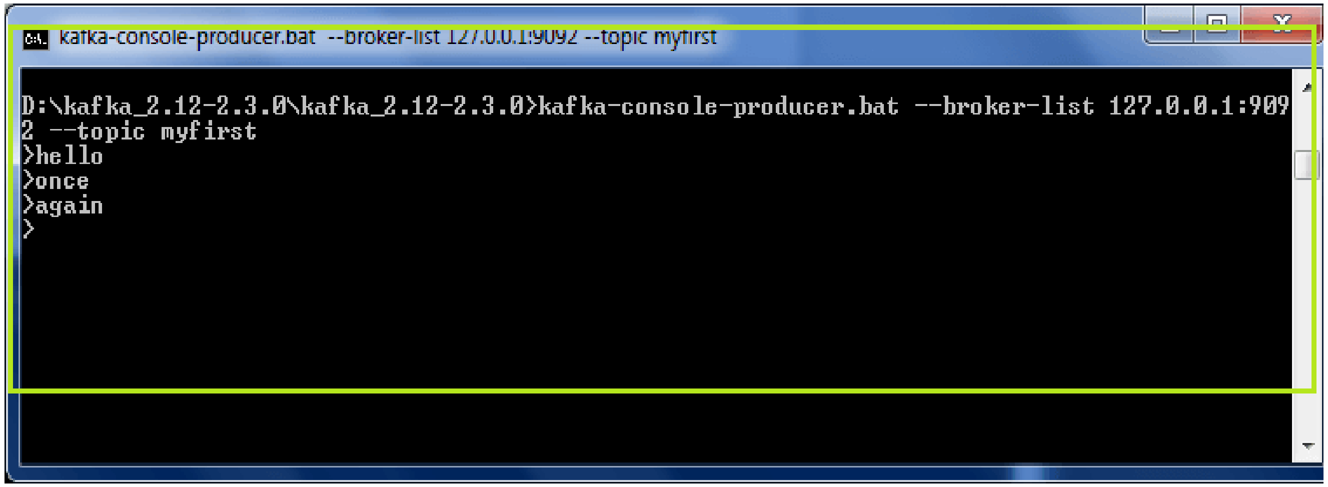
'--from-beginning' command

- 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.

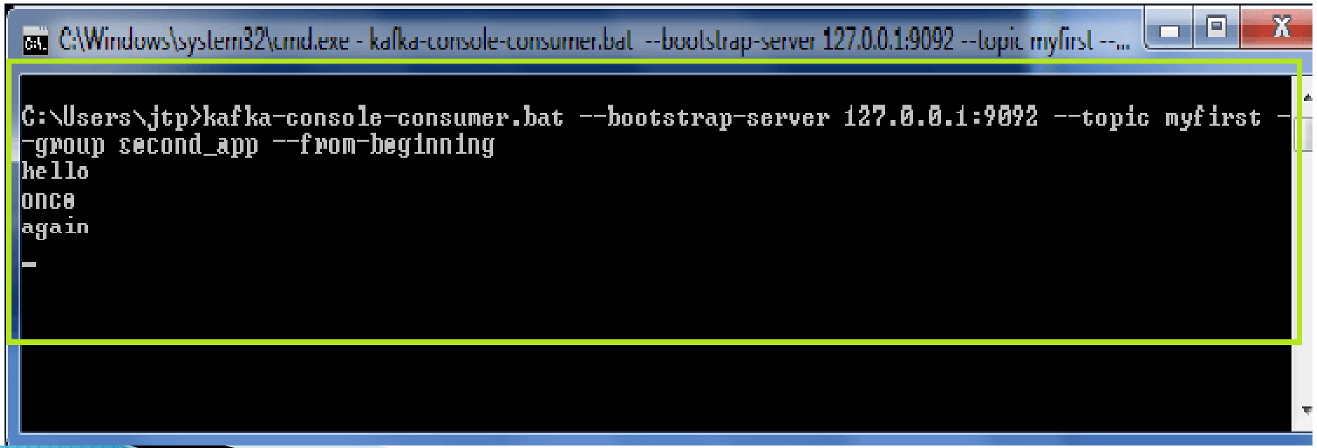
More about Consumer Group

'--from-beginning' command



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 127.0.0.1:909
2 --topic myfirst
>hello
>once
>again
>
```



C:\Windows\system32\cmd.exe - kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst --...

```
C:\Users\jtp>kafka-console-consumer.bat --bootstrap-server 127.0.0.1:9092 --topic myfirst -
-group second_app --from-beginning
hello
once
again
_
```

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

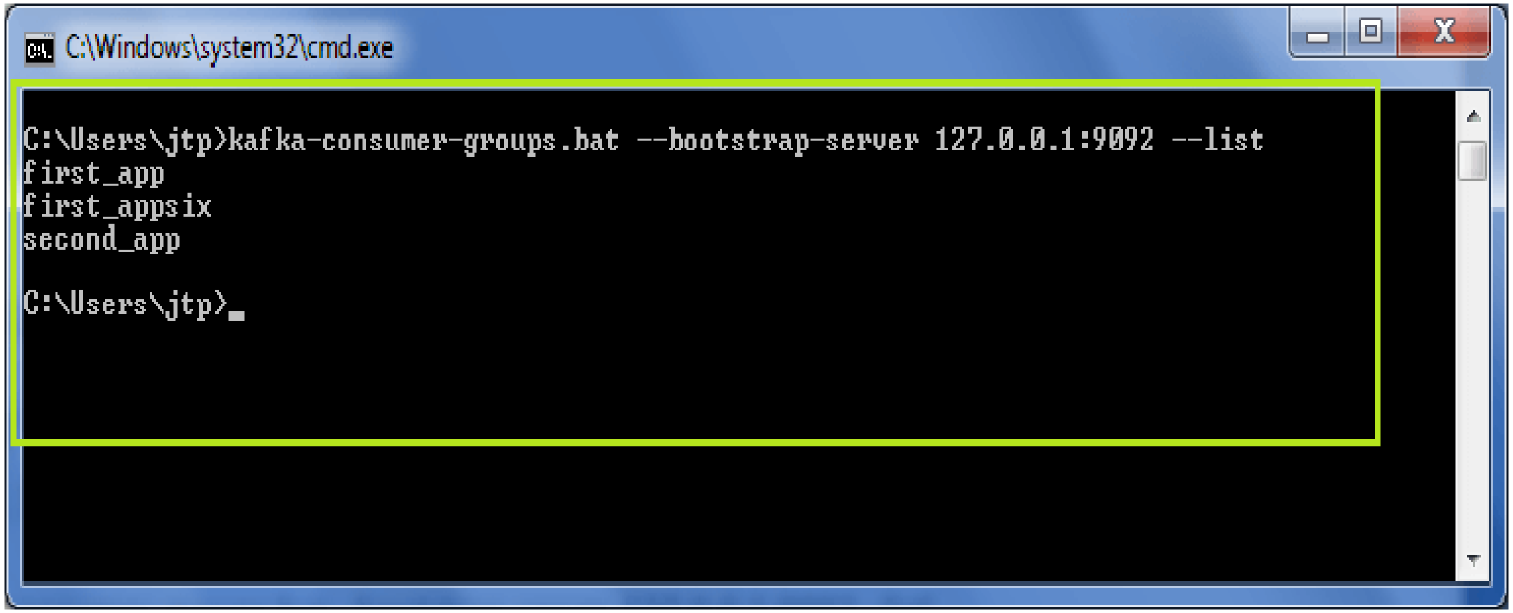
Listing Consumer Groups

- 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 --bootstrap-server localhost:9092 --list'.

More about Consumer Group

Listing Consumer Groups

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



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

C:\Users\jtp>_
```

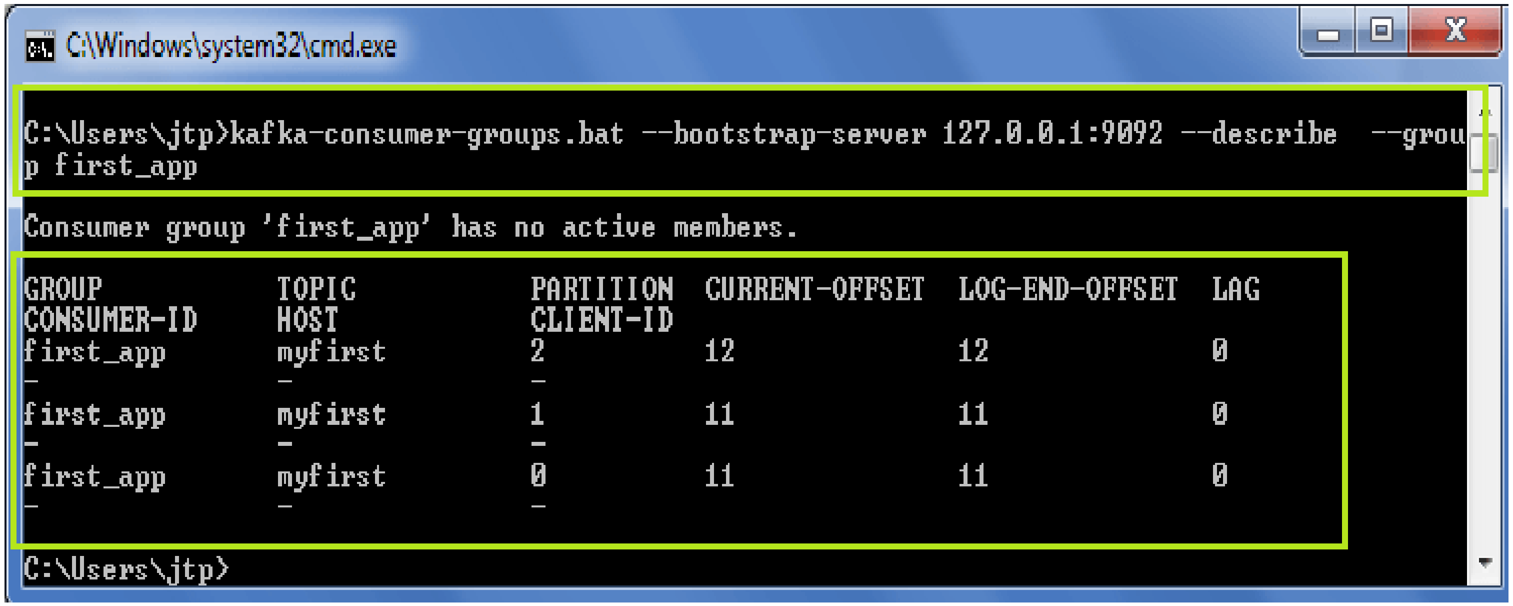
More about Consumer Group

Describing a Consumer Group

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

More about Consumer Group

Describing a Consumer Group



```
C:\Users\jtp>kafka-consumer-groups.bat --bootstrap-server 127.0.0.1:9092 --describe  --grou
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.

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.

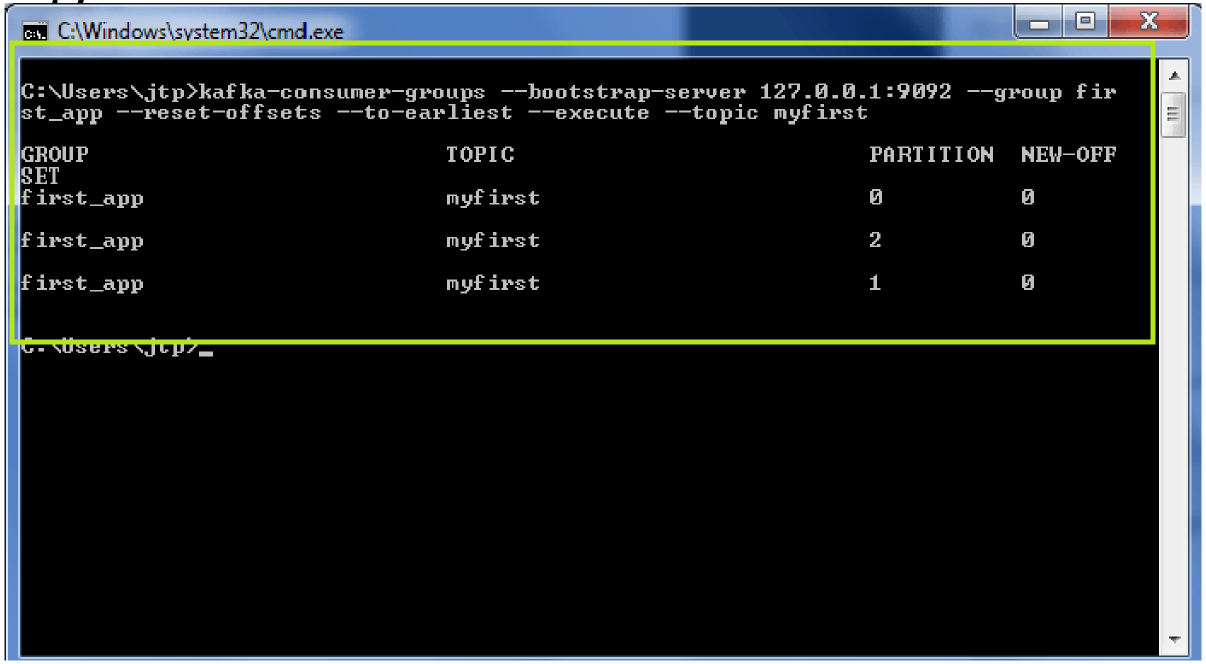
Resetting the Offsets

- 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



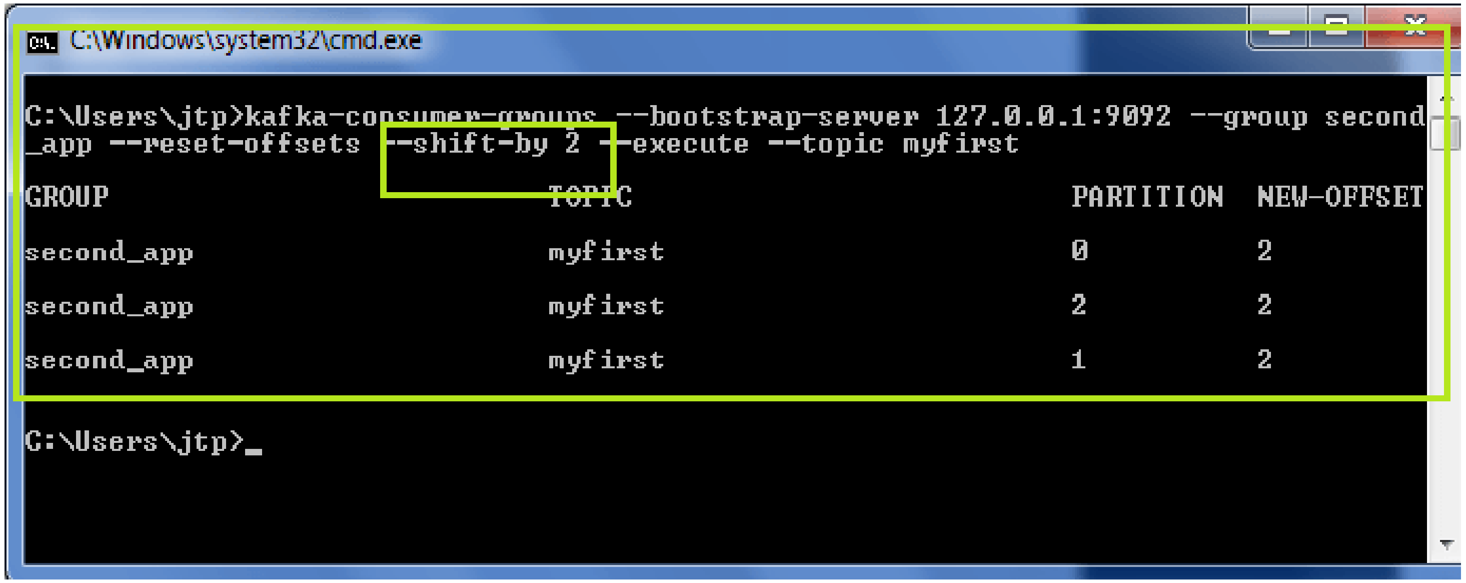
```
C:\Users\jtp>kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --group fir
st_app --reset-offsets --to-earliest --execute --topic myfirst

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.

Resetting the Offsets

Let's try and see:

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



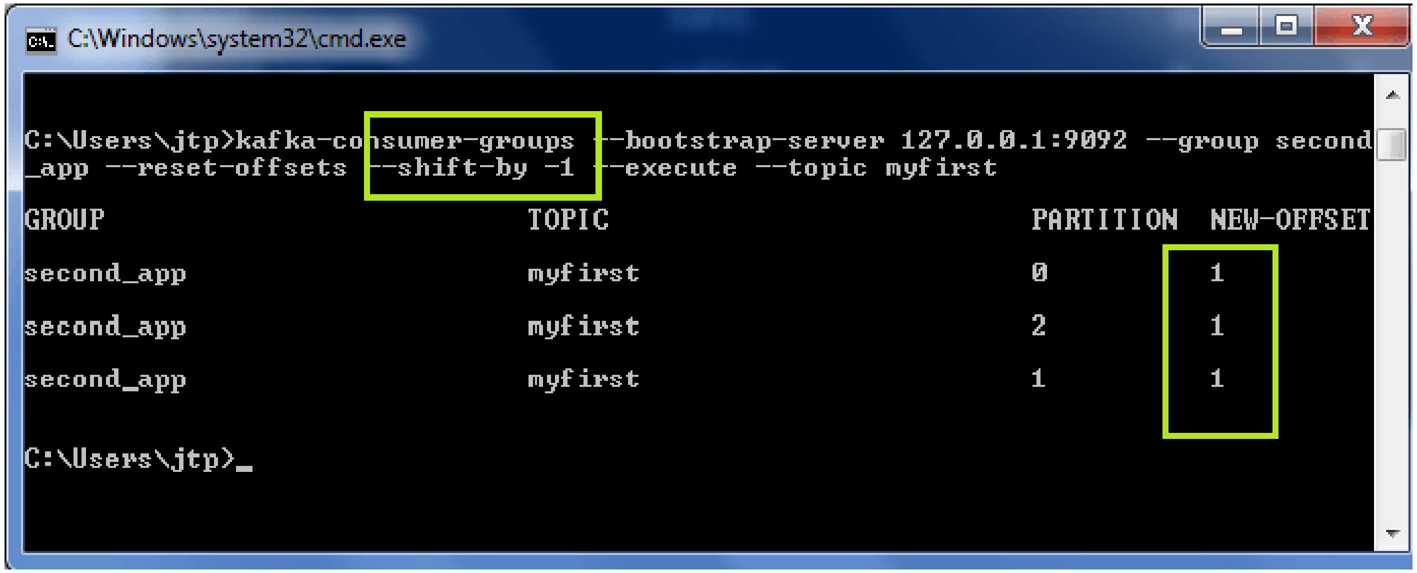
C:\Windows\system32\cmd.exe

```
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>_
```

Resetting the Offsets

Let's try and see:

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



```
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>_
```

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

Single Node-Multiple Brokers Configuration

- Before moving on to the multiple brokers cluster setup, first start your ZooKeeper server.
- Create Multiple Kafka Brokers − We have one Kafka broker instance already in etc/kafka/server.properties.
- Now we need multiple broker instances, so copy the existing server.properties file into two new config files and rename it as server-one.properties and server-two.properties.

Apache Kafka - Basic Operations

Single Node-Multiple Brokers Configuration

- Then edit both new files and assign the following changes −
  - etc/kafka/server-one.properties

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

# The port the socket server listens on
port=9093

# A comma seperated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-1
```

Apache Kafka - Basic Operations

Single Node-Multiple Brokers Configuration

- etc/kafka/server-two.properties

```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=2

# The port the socket server listens on
port=9094

# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs-2
```

Apache Kafka - Basic Operations

Single Node-Multiple Brokers Configuration

- Start Multiple Brokers– After all the changes have been made on three servers then open three new terminals to start each broker one by one.
- Broker1 : $bin/kafka-server-start etc/kafka/server.properties
- Broker2 : $bin/kafka-server-start etc/kafka/server-one.properties

Apache Kafka - Basic Operations

Single Node-Multiple Brokers Configuration

- Broker3 : $bin/kafka-server-start etc/kafka/server-two.properties

- Now we have three different brokers running on the machine.

- Try it by yourself to check all the daemons by typing jps on the terminal, then you would see the response.

Apache Kafka - Basic Operations

Creating a Topic

- Let us assign the replication factor value as three for this topic because we have three different brokers running.
- If you have two brokers, then the assigned replica value will be two.
- Syntax : $bin/kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 3 --partitions 1 --topic topic-name

Apache Kafka - Basic Operations

Creating a Topic

- Example: $bin/kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 3 --partitions 1 --topic Multibrokerapplication

- Output
  - created topic “Multibrokerapplication”

Apache Kafka - Basic Operations

Creating a Topic

- The Describe command is used to check which broker is listening on the current created topic as shown below −

```
$bin/kafka-topics --describe --bootstrap-server localhost:9092 --topic Multibrokerapplication
```

- Output

```
Topic:Multibrokerapplication PartitionCount:1 ReplicationFactor:3
```

Apache Kafka - Basic Operations

Creating a Topic

```
Configs:
Topic:Multibrokerapplication Partition:0 Leader:0 Replicas:0,2,1 Isr:0,2,1
```

- From the above output, we can conclude that first line gives a summary of all the partitions, showing topic name, partition count and the replication factor that we have chosen already.
- In the second line, each node will be the leader for a randomly selected portion of the partitions.

Apache Kafka - Basic Operations

Creating a Topic

- In our case, we see that our first broker (with broker.id 0) is the leader. Then Replicas:0,2,1 means that all the brokers replicate the topic finally Isr is the set of in-sync replicas.
- Well, this is the subset of replicas that are currently alive and caught up by the leader.

Apache Kafka - Basic Operations

Start Producer to Send Messages

- This procedure remains the same as in the single broker setup.
- Example : $bin/kafka-console-producer --broker-list localhost:9092 --topic Multibrokerapplication
- Output

```
[2016-01-20 19:27:21,045] WARN Property topic is not valid
(kafka.utils.VerifiableProperties)
This is single node-multi broker demo
This is the second message
```

Apache Kafka - Basic Operations

Start Consumer to Receive Messages

- This procedure remains the same as shown in the single broker setup.
- Example: $bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic Multibrokerapplication --from-beginning
- Output
  - This is single node-multi broker demo
  - This is the second message

Basic Topic Operations

Modifying a Topic

- Now let us modify a created topic using the following command

- Syntax: $bin/kafka-topics --bootstrap-server localhost:9092
  --alter --topic topic_name --partitions count

- Example: $bin/kafka-topics --bootstrap-server
localhost:9092 --alter --topic Hello-kafka --partitions 2

Basic Topic Operations

Modifying a Topic

- We have already created a topic “Hello-Kafka” with single partition count and one replica factor.
- Now using “alter” command we have changed the partition count.
- Output

WARNING: If partitions are increased for a topic that has a key, the partition logic or ordering of the messages will be affected Adding partitions succeeded!

Basic Topic Operations

Deleting a Topic

- To delete a topic, you can use the following syntax.
- Syntax: $bin/kafka-topics --bootstrap-server localhost:9092
  --delete --topic topic_name
- Example: $bin/kafka-topics --bootstrap-server localhost:9092 --delete --topic Hello-kafka
- Output: > Topic Hello-kafka marked for deletion

Multiple node – multiple broker cluster

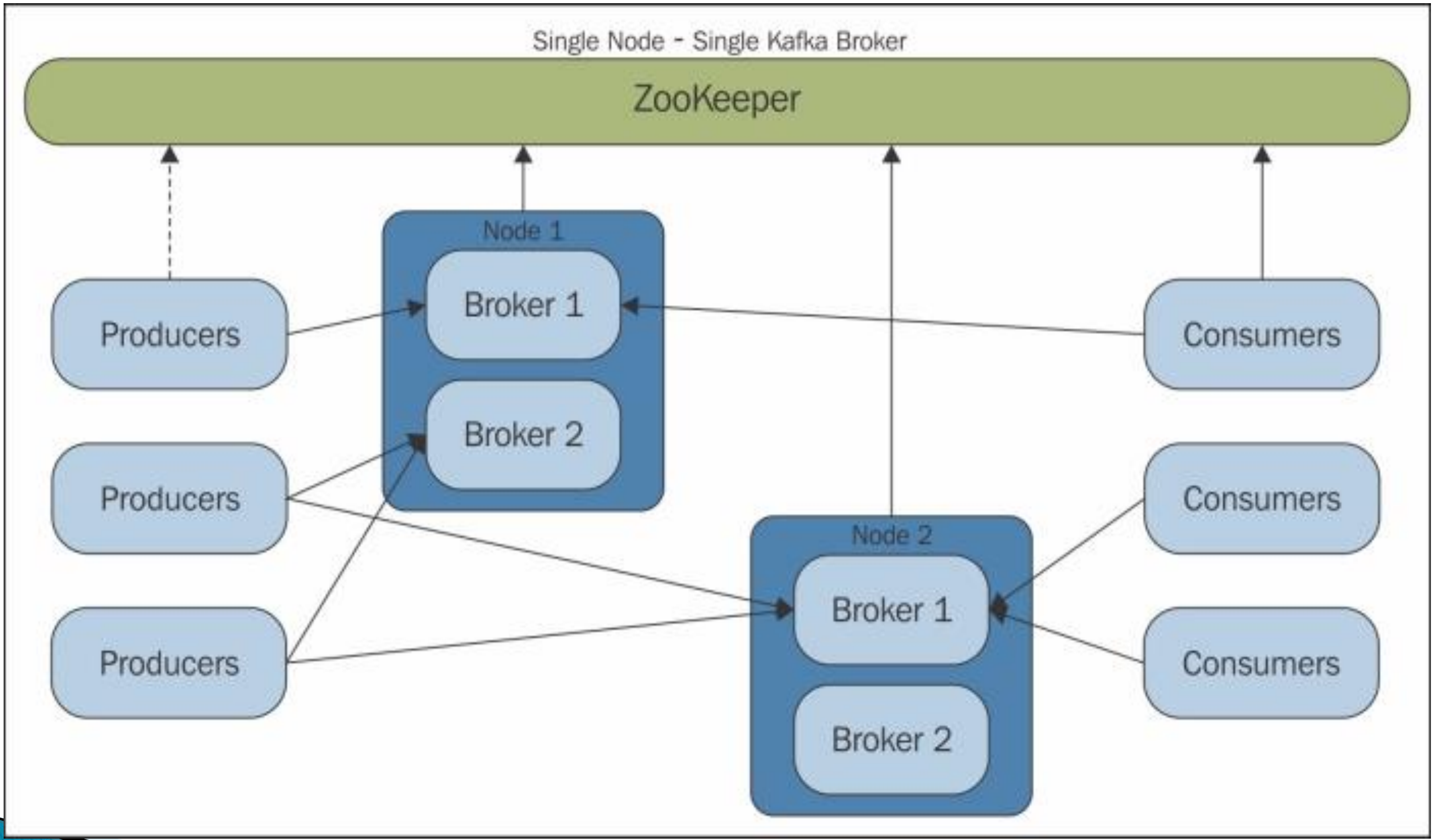
- As in the case of multiple-node Kafka cluster, where we set up multiple brokers on each node, we should install Kafka on each node of the cluster, and all the brokers from the different nodes need to connect to the same ZooKeeper.

- 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


Single Node - Single Kafka Broker
ZooKeeper
Node 1
Broker 1
Broker 2
Producers
Producers
Producers
Node 2
Broker 1
Broker 2
Consumers
Consumers
Consumers