Kafka Usecase

Gabriel D Hofer

July 22, 2021

Using single-node multi-broker setup , do the following: Create a topic called “logs” with 3 partitions and a replication factor of 3.

```
$ bin/kafka-topics.sh \
  --zookeeper localhost:2181 \
  --create --topic logs \
  --partitions 3 \
  --replication-factor 3
```

Describe the topic to understand in broker are the leader and in-sync replicas of your topic.

```
$ bin/kafka-topics.sh --zookeeper localhost:2181 --describe --topic logs
Topic: logs TopicId: Y7fJeVWCRzWNGLWHApE_bw PartitionCount: 3 ReplicationFactor: 3  Configs:
  Topic: logs Partition: 0  Leader: 3 Replicas: 3,2,1 Isr: 3,2,1
  Topic: logs Partition: 1  Leader: 1 Replicas: 1,3,2 Isr: 1,3,2
  Topic: logs Partition: 2  Leader: 2 Replicas: 2,1,3 Isr: 2,1,3
```

Produce some log messages to the “logs” topic as key-value pairs where log-level is the key and log message is the value. Ex: A message can look like “WARN: Message not properly formatted”. Use Kafka console producer for doing this.

```
$ bin/kafka-console-producer.sh --broker-list localhost:9092,localhost:9093,localhost:9094
    --sync --topic logs
>ERROR: php 21/05/2015
DEBUG:  php 11/01/2016
ERROR:  RailsApp  05/08/2015
ERROR:  php 19/05/2015
DONE: mysql 23/01/2016
ERROR:  php 13/02/2016
ERROR:  php 22/11/2014
ERROR:  RailsApp  25/12/2015
ERROR:  mysql 18/03/2015
DEBUG:  php 22/08/2015
ERROR:  RailsApp  06/05/2015
ERROR:  php 09/03/2015
DONE: mysql 28/06/2015
ERROR:  php 21/05/2015
DONE: php 11/01/2016
WARN: RailsApp  05/08/2015
INFO: php 19/05/2015
WARN: mysql 23/01/2016
INFO: php 13/02/2016
ERROR:  php 22/11/2014
ERROR:  RailsApp  25/12/2015
ERROR:  mysql 18/03/2015
WARN: php 22/08/2015
ERROR:  RailsApp  06/05/2015
ERROR:  php 09/03/2015
WARN: mysql 28/06/2015>
>>>>>>>>>>>>>>>>>>>>>>>>
```

Launch three instances of Kafka Console Consumers as a consumer group and demonstrate how messages are distributed across three partitions and how each consumer in the group is consuming from each of the three partitions.

Consumer instance one:

```
$ ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic logs --group
    logs-group
ERROR:  php 21/05/2015
ERROR:  RailsApp  05/08/2015
ERROR:  php 13/02/2016
DEBUG:  php 22/08/2015
DONE: mysql 28/06/2015
DONE: php 11/01/2016
ERROR:  php 22/11/2014
WARN: mysql 28/06/2015
```

Consumer instance two:

```
$ ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic logs --group
    logs-group
DEBUG:  php 11/01/2016
ERROR:  php 19/05/2015
ERROR:  RailsApp  25/12/2015
ERROR:  RailsApp  06/05/2015
INFO: php 19/05/2015
INFO: php 13/02/2016
ERROR:  mysql 18/03/2015
ERROR:  RailsApp  06/05/2015
```

Consumer instance three:

```
$ ./bin/kafka-console-consumer.sh --bootstrap-server localhost:9094 --topic logs --group
    logs-group
DONE: mysql 23/01/2016
ERROR:  php 22/11/2014
ERROR:  mysql 18/03/2015
ERROR:  php 09/03/2015
ERROR:  php 21/05/2015
WARN: RailsApp  05/08/2015
WARN: mysql 23/01/2016
ERROR:  RailsApp  25/12/2015
WARN: php 22/08/2015
ERROR:  php 09/03/2015
```

Write a simple Python client application using Kafka Producer API to producer messages to the “logs” topic. Using Fire-and-Forget approach for fastest message delivery. Send only “ERROR” messages.

```
from kafka import KafkaProducer
from time import sleep
import json
from datetime import datetime
import random

producer = KafkaProducer(bootstrap_servers=['localhost:9092'])
producer.send('logs', b'Client application using Kafka Producer API')

error_messages = [
    '500 Internal Server Error',
    '501 Not Implemented',
    '502 Bad Gateway',
    '503 Service Unavailable',
    '504 Gateway Timeout',
    '505 HTTP Version Not Supported',
    '506 Variant Also Negotiates (RFC 2295)',
    '507 Insufficient Storage (WebDAV; RFC 4918)',
    '508 Loop Detected (WebDAV; RFC 5842)',
    '510 Not Extended (RFC 2774)',
    '511 Network Authentication Required (RFC 6585)',
]

for i in range(50):
    message = " ERROR : {} ".format(str(  random.choice(error_messages)  ))
    producer.send('logs', json.dumps(message).encode('utf-8'))
    sleep(0.5)
    print("logs message ",i)
```

Write a simple Python client application using Kafka Consumer API to consume the data from the “logs” topic. Print the log messages on the Console. Turn-off auto-commit and explicit commit the messages from the application.

```
from kafka import KafkaConsumer
consumer = KafkaConsumer('logs', bootstrap_servers=['localhost:9092'])

# Turn off auto-commit messages
KafkaConsumer(enable_auto_commit=False, auto_commit_interval_ms=False)

for message in consumer:
    print(message.value)
```