Session 23:
KAFKA INTRODUCTION
Assignment 1

- Prachi Mohite

Below are some Brief about Kafka

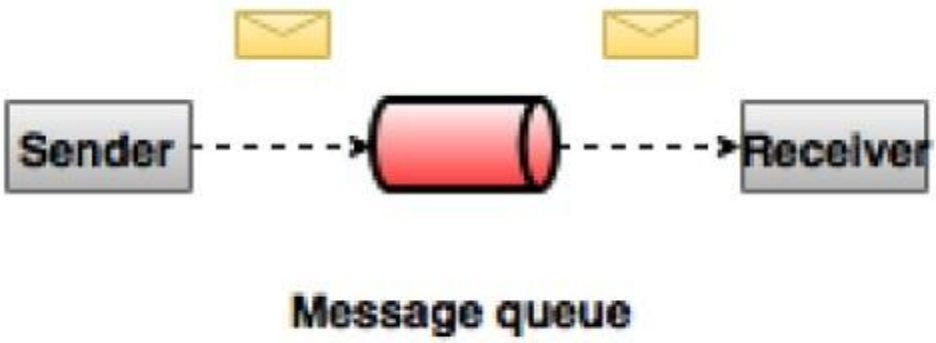
Kafka

Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables to pass messages from one end-point to another. Kafka is suitable for both offline and online message consumption. Kafka messages are persisted on the disk and replicated within the cluster to prevent data loss. Kafka is built on top of the ZooKeeper synchronization service. It integrates very well with Apache Storm and Spark or real-time streaming data analysis.

Two types of Messaging

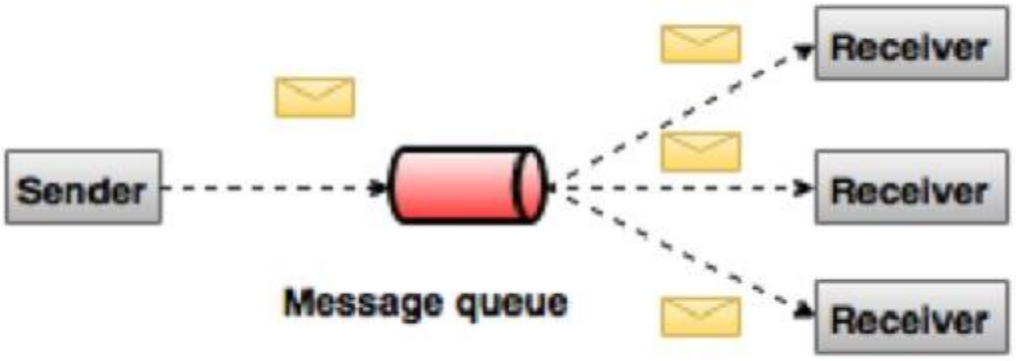
Point to Point

In a point-to-point system, messages are persisted in a queue. One or more consumers can consume the messages in the queue, but a particular message can be consumed by a maximum of one consumer only. Once a consumer reads a message in the queue, it disappears from that queue. E.g Order taking system



Publish-Subscribe Messaging System

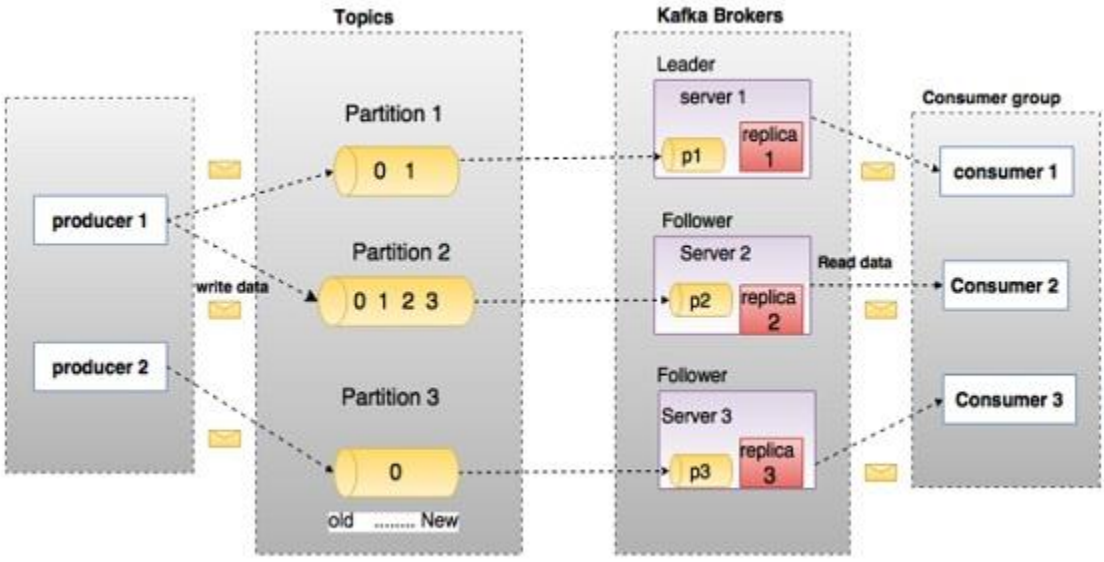
In the publish-subscribe system, messages are persisted in a topic. Unlike point-to-point system, consumers can subscribe to one or more topic and consume all the messages in that topic. In the Publish-Subscribe system, message producers are called publishers and message consumers are called subscribers. Real life example is Dish TV.



Benefits

Following are a few benefits of Kafka –

- **Reliability** − Kafka is distributed, partitioned, replicated and fault tolerance.
- **Scalability** − Kafka messaging system scales easily without down time..
- **Durability** − Kafka uses Distributed commit log which means messages persists on disk as fast as possible, hence it is durable..
- **Performance** − Kafka has high throughput for both publishing and subscribing messages. It maintains stable performance even many TB of messages are stored.



Some definitions

Topics - A stream of messages belonging to a particular category is called a topic. Data is stored in topics. Topics are split into partitions. For each topic, Kafka keeps a mini-mum of one partition. Each such partition contains messages in an immutable ordered sequence. A partition is implemented as a set of segment files of equal sizes.

Partition - Topics may have many partitions, so it can handle an arbitrary amount of data.

Partition offset- Each partitioned message has a unique sequence id called as offset.

Replicas of partition- Replicas are nothing but backups of a partition. Replicas are never read or write data. They are used to prevent data loss.

Brokers- Brokers are simple system responsible for maintaining the pub-lished data. Each broker may have zero or more partitions per topic. Assume, if there are N partitions in a topic and N number of brokers, each broker will have one partition.

Kafka Cluster- Kafka's having more than one broker are called as Kafka cluster. A Kafka cluster can be expanded without downtime. These clusters are used to manage the persistence and replication of message data.

Producers- Producers are the publisher of messages to one or more Kafka topics. Producers send data to Kafka brokers. Every time a producer pub-lishes a message to a broker, the broker simply appends the message to the last segment file. Actually, the message will be appended to a partition. Producer can also send messages to a partition of their choice.

Consumers- Consumers read data from brokers. Consumers subscribes to one or more topics and consume published messages by pulling data from the brokers.

Leader- Leader is the node responsible for all reads and writes for the given partition. Every partition has one server acting as a leader.

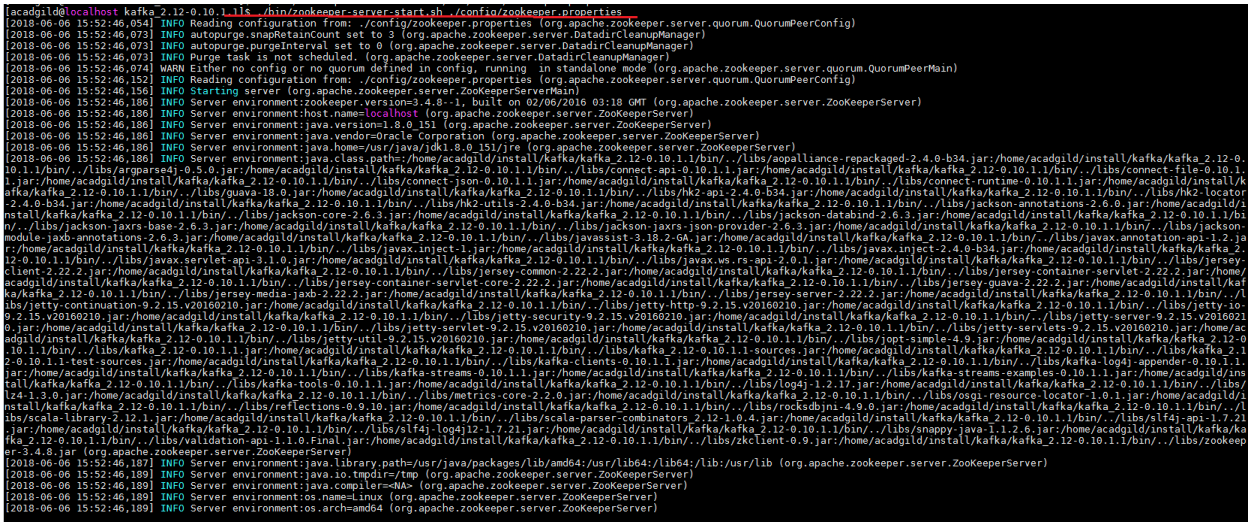
Follower- Node which follows leader instructions are called as follower. If the leader fails, one of the follower will automatically become the new leader. A follower acts as normal consumer, pulls messages and up-dates its own data store.

For this assignment we have to make sure that ZooKeeper and Kafka is running

1. Start the ZooKeeper with below command

Commands

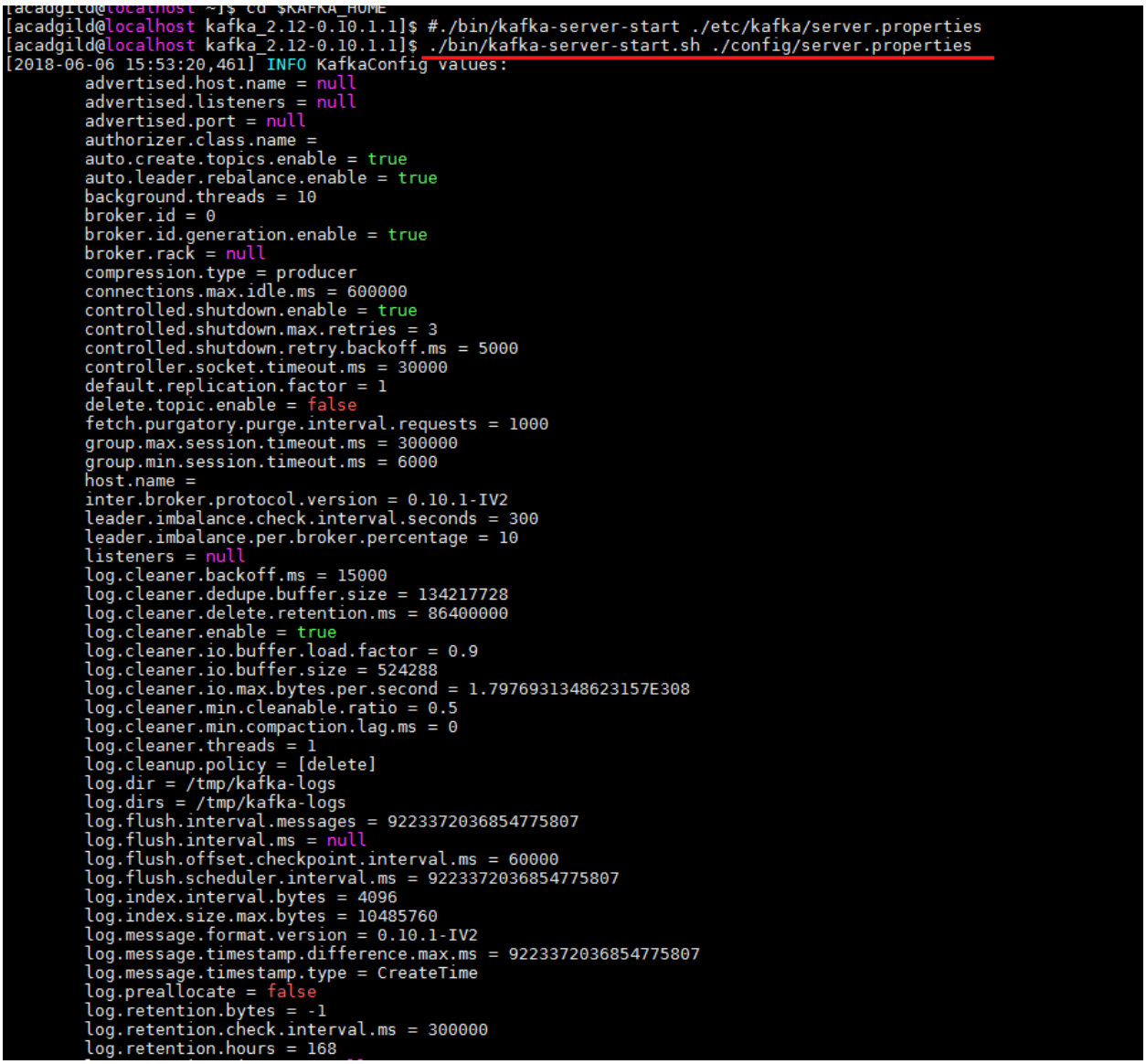
- cd $KAFKA_HOME ($KAFKA_HOME = /home/acadgild/install/kafka/kafka_2.12-0.10.1.1/)
- ./bin/zookeeper-server-start.sh ./config/zookeeper.properties



2. Start the Kafka Broker

Commands

- cd $KAFKA_HOME
- ./bin/kafka-server-start.sh ./config/server.properties



Then start new command prompt window

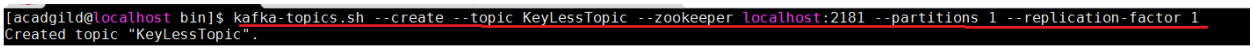
Task 1:
Create a kafka topic named KeyLessTopic.

Solution Approach –

Use the below command to create the topic

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1
--partitions 1 --topic topic-name
```

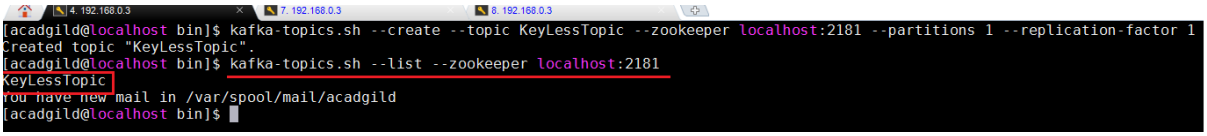
Code Execution as below



Verify if topic is created with the list command as below

```
bin/kafka-topics.sh --list --zookeeper localhost:2181
```

List of topics created



Task 1.2 Inside KeyLessTopic insert following data:

{"name":"John", "exp":16}
{"name":"Finn", "exp":20}
{"name":"Cylin", "exp":18}
{"name":"Mark", "exp":2}
{"name":"Akshay", "exp":14}

To Insert the data we must start the producer

Start Producer to Send Messages

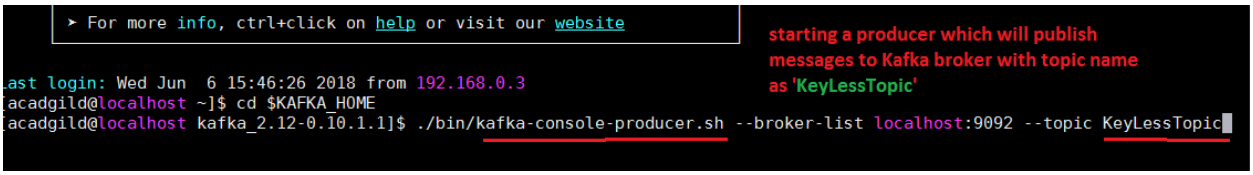
```
bin/kafka-console-producer.sh --broker-list localhost:9092 --topic topic-name
```

Broker-list – The list of brokers that we want to send the messages to. In this case we only have one broker. The Config/server.properties file contains broker port id, since we know our broker is listening on port 9092, so you can specify it directly.

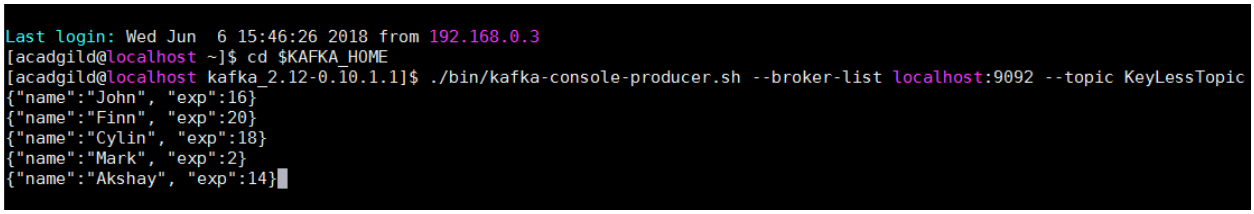
Topic name – Here is an example for the topic name.

Every line entered on the console is treated as a value with **NULL as key** as this topic is key less

Code execution



Inserting the data



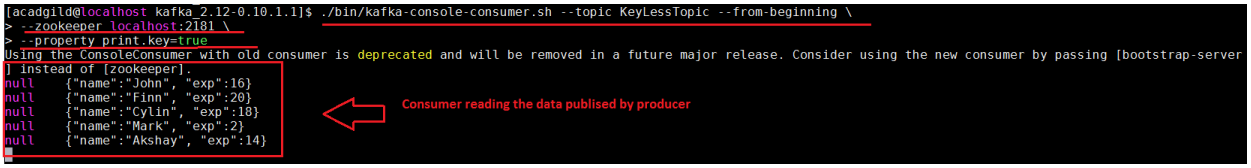
Task 2:
Create a console consumer that reads KeyLessTopic from beginning

To read we have to **Start Consumer to Receive Messages**

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

```
bin/kafka-console-consumer.sh --zookeeper localhost:2181 —topic topic-name
--from-beginning
```

Code Execution with output



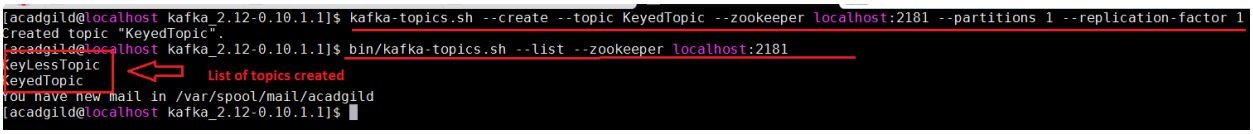
Task 3:
Create a kafka topic named KeyedTopic. Inside KeyedTopic insert following data:
The part before comma(,) should be treated as key and after comma(,) should be treated as value

Data:

{"name":"John"},{"exp":16}
{"name":"Finn"},{"exp":20}
{"name":"Cylin"},{"exp":18}
{"name":"Mark"},{"exp":2}
{"name":"Akshay"},{"exp":14}

Create a topic which accepts key-value pair of data

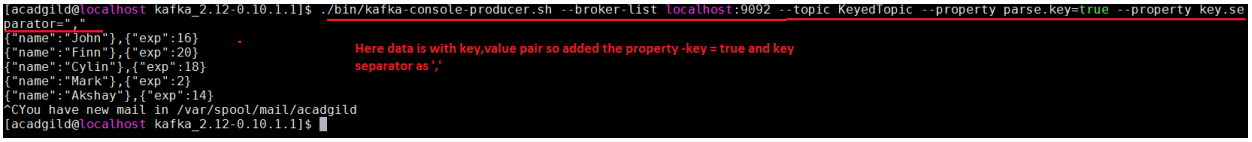
Code Execution and Output



Create a producer to insert the data with key value pair . To achive the same we have to metion two properties while creating producer

- Parse-key as true
- Key separator as ‘,’ (comma) in this case.

Code Execution and Output



Task 4:
Create a console consumer that reads KeyedTopic from beginning
The key and value should be separated by '-'

The topic has data inserted in key value format by the producer created in above task (3). However the producer gave the key separator as ‘,’. However we can overwrite the key separator value in console consumer and mandate to print key-value pair. To achive the same we must write console consumer with the two properties

- Print.key = true
- Key-separator = ‘-‘

Command is

./bin/kafka-console-consumer.sh --topic KeyedTopic --property print.key=true --property key.separator="-" --from-beginning --zookeeper localhost:2181 --property print.key=true

Code Execution and Output