

Session 23:
KAFKA INTRODUCTION
Assignment 1
- Prachi Mohite

Below are some Brief about Kafka

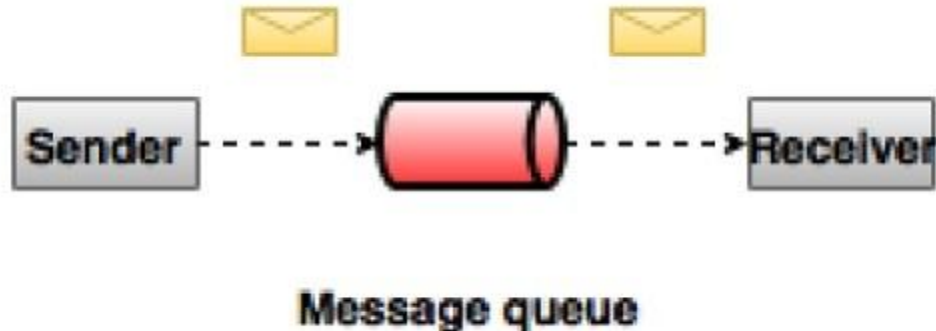
Kafka

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

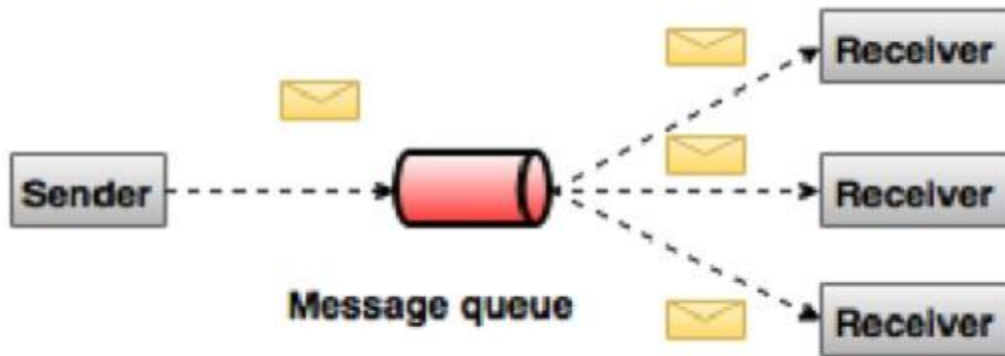
Point to Point

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

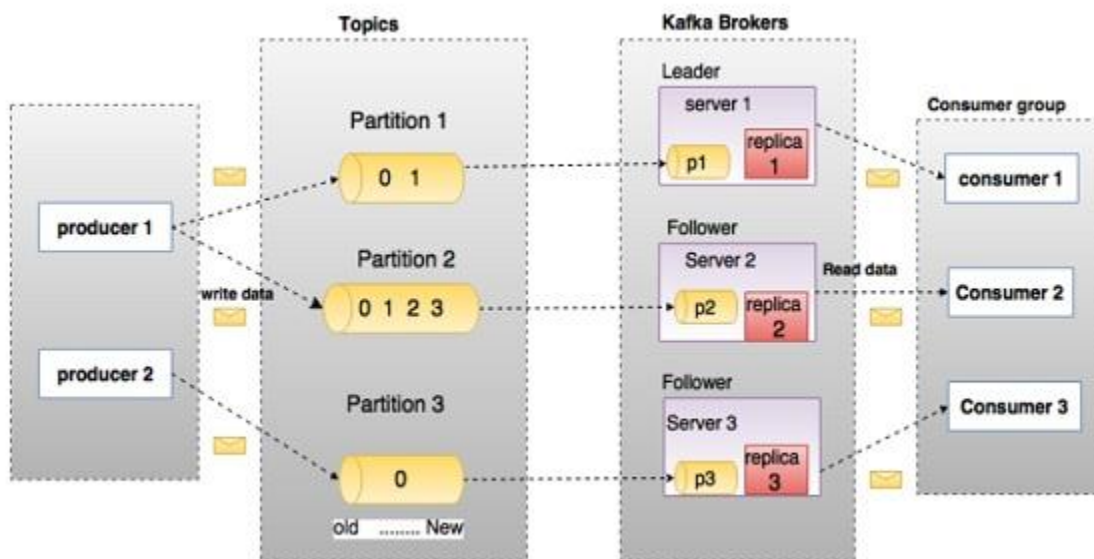
In the publish-subscribe system, messages are persisted in a topic. Unlike point-to-point system, consumers can subscribe to one or more topic and consume all the messages in that topic. In the Publish-Subscribe system, message producers are called publishers and message consumers are called subscribers. 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 minimum 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 published 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 publishes 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 subscribe 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 updates 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`

[illegible]

2. Start the Kafka Broker

Commands

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

```
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ # ./bin/kafka-server-start ./etc/kafka/server.properties
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-server-start.sh ./config/server.properties
[2018-06-06 15:53:20,461] INFO KafkaConfig values:
    advertised.host.name = null
    advertised.listeners = null
    advertised.port = null
    authorizer.class.name =
    auto.create.topics.enable = true
    auto.leader.rebalance.enable = true
    background.threads = 10
    broker.id = 0
    broker.id.generation.enable = true
    broker.rack = null
    compression.type = producer
    connections.max.idle.ms = 600000
    controlled.shutdown.enable = true
    controlled.shutdown.max.retries = 3
    controlled.shutdown.retry.backoff.ms = 5000
    controller.socket.timeout.ms = 30000
    default.replication.factor = 1
    delete.topic.enable = false
    fetch.purgatory.purge.interval.requests = 1000
    group.max.session.timeout.ms = 300000
    group.min.session.timeout.ms = 6000
    host.name =
    inter.broker.protocol.version = 0.10.1-IV2
    leader.imbalance.check.interval.seconds = 300
    leader.imbalance.per.broker.percentage = 10
    listeners = null
    log.cleaner.backoff.ms = 15000
    log.cleaner.dedupe.buffer.size = 134217728
    log.cleaner.delete.retention.ms = 86400000
    log.cleaner.enable = true
    log.cleaner.io.buffer.load.factor = 0.9
    log.cleaner.io.buffer.size = 524288
    log.cleaner.io.max.bytes.per.second = 1.7976931348623157E308
    log.cleaner.min.cleanable.ratio = 0.5
    log.cleaner.min.compaction.lag.ms = 0
    log.cleaner.threads = 1
    log.cleanup.policy = [delete]
    log.dir = /tmp/kafka-logs
    log.dirs = /tmp/kafka-logs
    log.flush.interval.messages = 9223372036854775807
    log.flush.interval.ms = null
    log.flush.offset.checkpoint.interval.ms = 60000
    log.flush.scheduler.interval.ms = 9223372036854775807
    log.index.interval.bytes = 4096
    log.index.size.max.bytes = 10485760
    log.message.format.version = 0.10.1-IV2
    log.message.timestamp.difference.max.ms = 9223372036854775807
    log.message.timestamp.type = CreateTime
    log.preallocate = false
    log.retention.bytes = -1
    log.retention.check.interval.ms = 300000
    log.retention.hours = 168
```

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

```
[acagdild@localhost bin]$ kafka-topics.sh --create --topic KeyLessTopic --zookeeper localhost:2181 --partitions 1 --replication-factor 1
Created topic "KeyLessTopic".
```

Verify if topic is created with the list command as below

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

List of topics created

```
[acagdild@localhost bin]$ kafka-topics.sh --create --topic KeyLessTopic --zookeeper localhost:2181 --partitions 1 --replication-factor 1
Created topic "KeyLessTopic".
[acagdild@localhost bin]$ kafka-topics.sh --list --zookeeper localhost:2181
KeyLessTopic
you have new mail in /var/spool/mail/acagdild
[acagdild@localhost bin]$
```

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

```
> For more info, ctrl+click on help or visit our website

Last login: Wed Jun  6 15:46:26 2018 from 192.168.0.3
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic KeyLessTopic
```

starting a producer which will publish
messages to Kafka broker with topic name
as 'KeyLessTopic'

Inserting the data

```
Last login: Wed Jun  6 15:46:26 2018 from 192.168.0.3
[acadgild@localhost ~]$ cd $KAFKA_HOME
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic KeyLessTopic
{"name":"John", "exp":16}
{"name":"Finn", "exp":20}
{"name":"Cylin", "exp":18}
{"name":"Mark", "exp":2}
{"name":"Akshay", "exp":14}
```

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

```
[acadgild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic KeyLessTopic --from-beginning \
> --zookeeper localhost:2181 \
> --property print.key=true
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server]
instead of [zookeeper].
null {"name":"John", "exp":16}
null {"name":"Finn", "exp":20}
null {"name":"Cylin", "exp":18}
null {"name":"Mark", "exp":2}
null {"name":"Akshay", "exp":14}
```

Consumer reading the data published by producer

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

```
[acadmild@localhost kafka_2.12-0.10.1.1]$ kafka-topics.sh --create --topic KeyedTopic --zookeeper localhost:2181 --partitions 1 --replication-factor 1  
Created topic "KeyedTopic".  
[acadmild@localhost kafka_2.12-0.10.1.1]$ bin/kafka-topics.sh --list --zookeeper localhost:2181  
keylessTopic  
KeyedTopic  
you have new mail in /var/spool/mail/acadmild  
[acadmild@localhost kafka_2.12-0.10.1.1]$
```

← List of topics created

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

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

Code Execution and Output

```
[acadmild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic KeyedTopic --property parse.key=true --property key.separator=",  
{\"name\":\"John\"},{\"exp\":16}  
{\"name\":\"Finn\"},{\"exp\":20}  
{\"name\":\"Cylin\"},{\"exp\":18}  
{\"name\":\"Mark\"},{\"exp\":2}  
{\"name\":\"Akshay\"},{\"exp\":14}  
^CYou have new mail in /var/spool/mail/acadmild  
[acadmild@localhost kafka_2.12-0.10.1.1]$
```

Here data is with key,value pair so added the property -key = true and key separator as ','

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 achieve 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

```
[acagild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-producer.sh --broker-list localhost:9092 --topic KeyedTopic --property parse.key=true --property key.separator=","
{"name":"John"}, {"exp":16}
{"name":"Finn"}, {"exp":20}
{"name":"Cylin"}, {"exp":18}
{"name":"Mark"}, {"exp":2}
{"name":"Akshay"}, {"exp":14}
^CYou have new mail in /var/spool/mail/acagild
[acagild@localhost kafka_2.12-0.10.1.1]$ ./bin/kafka-console-consumer.sh --topic KeyedTopic --property print.key=true --property key.separator="-" --from-beginning
> --zookeeper localhost:2181 \
> --property print.key=true
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
{"name":"John"}-{"exp":16}
{"name":"Finn"}-{"exp":20}
{"name":"Cylin"}-{"exp":18}
{"name":"Mark"}-{"exp":2}
{"name":"Akshay"}-{"exp":14}
```



Here is output with the '-' separated, to get this we have to specify the two properties print.key=true and key.separator='-'