rkafka

---

**rkafka is a package created to expose functionalities provided by Apache Kafka in the R layer.**

---

**Version 1.1**

Wednesday 28th June, 2017

# rkafka

Shruti Gupta

Wednesday 28th June, 2017

## Contents

**Abstract**

Apache Kafka is an open-source message broker project developed by the Apache Software Foundation written in Scala. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds.

This package contains functions that allows a R developer to use the messaging functionalities provided by Apache Kafka.

# 1 Introduction

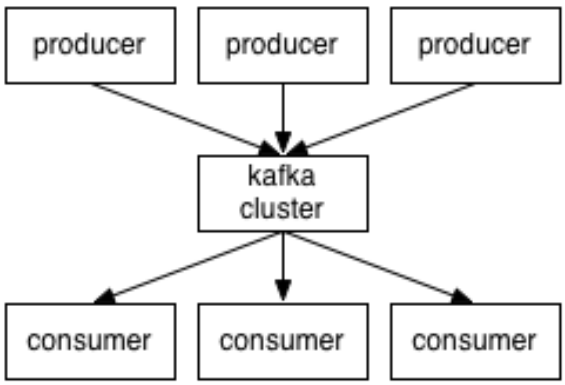
rkafka is a package created to expose functionalities provided by Apache Kafka in the R layer.

Apache Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design.

## 1.1 Messaging Terminology

- **Topics:** Kafka maintains feeds of messages in categories called topics.
- **Producers:** Processes that publish messages to a Kafka topic are producers.
- **Consumers:** Processes that subscribe to topics and process the feed of published messages are consumers.

Kafka is run as a cluster comprised of one or more servers each of which is called a broker. So, at a high level, producers send messages over the network to the Kafka cluster which in turn serves them up to consumers like this:




For more on Apache Kafka, refer `http://kafka.apache.org/documentation.html#introduction`

# 2 Administration

To make use of this package, following steps need to be followed:

- Apache Kafka(Version 2.8.0-0.8.1.1) to be downloaded
- Zookeeper server needs to be running
- Kafka server needs to be running

# 3 Getting Started

## 3.1 Make sure the zookeeper server and Kafka server are running

## 3.2 Creating a Producer

To create a producer, Kafka needs to know the list of brokers. If Kafka is setup on the local system, Kafka producer can be created by the following line of code:

***producer1=rkafka.createProducer("127.0.0.1:9092")***

This creates a producer with all the default properties. Any of the properties of the producer can be altered by changing the corresponding arguments.

## 3.3 Sending messages

After creating a producer, it can be used to write messages to topics. If a topic doesn't exist already, it's created. For producer1 to write a message to a topic named "test", the following line of code will be used:

***rkafka.send(producer1,"test","127.0.0.1:9092","Testing")***

## 3.4 Creating a consumer

To read messages from a topic, a consumer needs to be created. One consumer can be associated with only one topic. To read from multiple topics, multiple consumers need to be created.

To create a consumer, the IP of the Zookeeper server to which the consumer has to connect to and the topic from which the consumer has to be read needs to be specified. If Zookeeper is setup on the local system, use the following code to create a consumer:

***consumer1=rkafka.createConsumer("127.0.0.1:2181","test")***

## 3.5 Reading messages

The messages from a topic can be read in two ways:-one at a time and in a batch

If your application requires messages to be received one at a time, use the ***rkafka.read()*** function.

However, if your application requires messages to received in a batch, use the ***rakfka.readPoll()*** function.

Both the functions, wait for a period of 'x' ms(specified by ConsumerTimeoutMsProperty while creating the consumer), for new messages. After 'x' ms have passed, they'll return a blank string.

## 3.6 Closing the producer and consumer

It's a good coding practice to close all the open connections in your application.

Besides it's essential to close the producer connection after message sending is done, and to close the consumer connection after your application is done receiving messages.

Failure to do so can result in the producer and consumer behaving erratically. To close the producer and consumer created in the previous examples, use the following lines of code:

***rkafka.closeProducer(producer1)***

***rkakfa.closeConsumer(consumer1)***

## 3.7 Simple Consumer

Kafka has a lower-level consumer api for reading message chunks directly from servers. Under most circumstances this should not be needed as the developer of the application needs to handle offsets. It's usage is as follows:

### 3.7.1 Creating Simple Consumer

***consumer2=rkafka.createSimpleConsumer("172.25.1.78","9092","10000","100000","test")***

### 3.7.2 Reading messages

Reading messages through Simple Consumer requires the rkafka.receivefromSimpleConsumer function to be called first. The topic to read from, offset, partition and client ID can be passed as arguments to this function.

The following line of code reads from the topic "test" from 0th partition and 0th offset:

***rkafka.receiveFromSimpleConsumer(consumer2,"test","0","0","test-group")***

After calling this function, to read the messages received by the SimpleConsumer, use the rkafka.readFromSimpleConsumer() function.

This function returns one message at a time.

Executing the following line of code will return the first message read by the SimpleConsumer

***print(rkafka.readFromSimpleConsumer(consumer2))***

### 3.7.3 Closing Simple Consumer

To close the simple consumer, execute the following line of code

***rkafka.closeSimpleConsumer(consumer2)***

# 4 Example of Package Usage

The following piece of code is based on the following assumptions: Kafka server is running on "127.0.0.1:9092" Zookeeper server is running on "127.0.0.1:2181"

After ensuring Kafka server and Zookeeper server are running, create a producer.

***>producer1=rkafka.createProducer("127.0.0.1:9092")***

Send messages to the topic "test".

***>rkafka.send(producer1,"test","127.0.0.1:9092","Message 1")***

***>rkafka.send(producer1,"test","127.0.0.1:9092","Message 2")***

Create consumer to read from the topic "test".

***>consumer1=rkafka.createConsumer("127.0.0.1:2181","test")***

Read first message from the topic "test".

***>print(rkafka.read(consumer1))***

Message 1

***>print(rkafka.read(consumer1))***

Message 2

***>print(rkafka.read(consumer1))***

""

Send more messages to the topic "test".

***>rkafka.send(producer1,"test","127.0.0.1:9092","Message 3")***

***>rkafka.send(producer1,"test","127.0.0.1:9092","Message 4")***

***>rkafka.send(producer1,"test","127.0.0.1:9092","Message 5")***

***>print(rkafka.read(consumer1))***

Message 3

***>print(rkafka.readPoll(consumer1))***

Message 4 , Message 5

Closing the producer.

***>rkafka.closeProducer(producer1)***

Closing the consumer.

***>rkafka.closeConsumer(consumer1)***

## References

[1] Apache Kafka documentation : `http://kafka.apache.org/documentation.html`