Package 'rkafka'

October 14, 2022

Type Package

Title Using Apache 'Kafka' Messaging Queue Through 'R'

Version 1.4

Date 2022-08-10

Author Shruti Gupta[aut,cre]

Maintainer Shruti Gupta <shrutigupta34@gmail.com>

Description Apache 'Kafka' is an open-source message broker project developed by the Apache Software Foundation which can be thought of as a distributed, partitioned, replicated commit log service.At a high level, producers send messages over the network to the 'Kafka' cluster which in turn serves them up to consumers.See <https://kafka.apache.org/> for more information.Functions included in this package enable:1.Creating 'Kafka' producer 2.Writing messages to a topic 3.Closing 'Kafka' producer 4.Creating 'Kafka' consumer 5.Reading messages from a topic 6.Closing 'Kafka' consumer. The jars required for this package are included in a separate package 'rkafkajars'.

Depends rJava,RUnit,rkafkajars

SystemRequirements Java JDK 1.7 or higher,Apache Kafka 2.8.0-0.8.1.1

License Apache License 2.0 | file LICENSE

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-10 15:40:02 UTC

R topics documented:

---

rkafka *Using Apache 'Kafka' Messaging Queue Through 'R'*

Description

It provides functionalities of creating a 'Kafka' producer, simple consumer,high level consumer and sending and receiving messages.

Details

Package:	rkafka
Type:	Package
Version:	1.3
Date:	2021-12-05
License:	Apache License 2.0

1)Start 'Zookeeper' server. 2)Start 'Kafka' server. 3)Start producer using 'rkafka.createProducer' function. 4)Send messages using 'rkafka.send' function. 5)Close producer using 'rkafka.closeProducer' function. 6)Start consumer using 'rkafka.createConsumer' function. 7)Read messages using 'rkafka.read' function. 8)Close consumer using 'rkafka.closeConsumer' function.

Author(s)

Shruti Gupta

Maintainer: Who to complain to shrutigupta34@gmail.com

References

To understand 'Kafka' https://kafka.apache.org/documentation.html

Examples

```
## Not run:
prod1=rkafka.createProducer("127.0.0.1:9092")
rkafka.send(prod1,"test","127.0.0.1:9092","Testing once")
rkafka.send(prod1,"test","127.0.0.1:9092","Testing twice")
rkafka.send(prod1,"test","127.0.0.1:9092","Testing thrice")
rkafka.closeProducer(prod1)
consumer1=rkafka.createConsumer("127.0.0.1:2181","test")
print(rkafka.read(consumer1))
print(rkafka.read(consumer1))
```

```
print(rkafka.read(consumer1))

## End(Not run)
```

`rkafka.closeConsumer` *Closing KAKFA consumer*

Description

This functions shuts down the KAFKA consumer

Usage

```
rkafka.closeConsumer(ConsumerObj)
```

Arguments

`ConsumerObj`	ConsumerObj:Consumer through which messages are to be read(Java Object) Required:Mandatory Type:Consumer

Value

Function doesn't return anything

Author(s)

Shruti Gupta

Examples

```
## Not run:
consumer1=rkafka.createHighConsumer("127.0.0.1:2181")
rkafka.closeHighConsumer(consumer1)

## End(Not run)
```

rkafka.closeProducer *KAFKA producer shutdown*

Description

This function closes the KAFKA producer

Usage

```
rkafka.closeProducer(producer)
```

Arguments

producer	Producer which is to be terminated Required:Mandatory Type:Producer

Value

Doesn't return anything

Author(s)

Shruti Gupta

Examples

```
## Not run:
producer1=rkafka.createProducer("127.0.0.1:9092")
rkafka.closeProducer(producer1)

## End(Not run)
```

rkafka.closeSimpleConsumer
Closing KAKFA Simple consumer

Description

This functions shuts down the KAFKA Simple consumer

Usage

```
rkafka.closeSimpleConsumer(SimpleConsumer)
```

Arguments

SimpleConsumer	SimpleConsumer:SimpleConsumer that has to be shut down Required:Mandatory Type:SimpleConsumer

Details

There are two types of KAFKA consumers:High-Level and Simple.This function shuts down the KAFKA Simple Consumer

Value

Function doesn't return anything

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

Examples

```
## Not run:
consumer1=rkafka.createSimpleConsumer("172.25.1.78","9092","10000","100000","test")
rkafka.receiveFromSimpleConsumer(consumer1,"test","0","0","test-group")
print(rkafka.readFromSimpleConsumer(consumer1))
rkafka.closeSimpleConsumer(consumer1)

## End(Not run)
```

---

rkafka.createConsumer *Creating KAFKA consumer*

Description

This function creates a KAFKA consumer

Usage

```
rkafka.createConsumer(zookeeperConnect, topicName,
groupId="test-consumer-group", zookeeperConnectionTimeoutMs="100000",
consumerTimeoutMs="10000", autoCommitEnable="NULL",
autoCommitInterval="NULL", autoOffsetReset="NULL")
```

Arguments

zookeeperConnect	Zookeeper connection string comma separated host:port pairs, each corresponding to a zk server. e.g."127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002" Required:Mandatory Type:String default:NONE
topicName	Name of the topic from which to read messages Required:Mandatory Type:String

`groupId` consumer group id Required:Mandatory Type:String default:test-consumer-group

`zookeeperConnectionTimeoutMs`
: timeout in ms for connecting to zookeeper Required:Mandatory Type:String default:100000

`consumerTimeoutMs`
: Throw a timeout exception to the consumer if no message is available for consumption after the specified interval Required:Mandatory Type:String default:10000

`autoCommitEnable`
: If true, periodically commit to ZooKeeper the offset of messages already fetched by the consumer. This committed offset will be used when the process fails as the position from which the new consumer will begin. Required:Optional Type:String default:true

`autoCommitInterval`
: The frequency in ms that the consumer offsets are committed to zookeeper. Required:Optional Type:String default:60*1000

`autoOffsetReset`
: smallest : automatically reset the offset to the smallest offset largest : automatically reset the offset to the largest offset anything else: throw exception to the consumer Required:Optional Type:String default:largest

Details

There are two types of KAFKA consumers: High-level and Simple. This functions creates a high level consumer

Value

Returns a consumer

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

To know how to use a high level consumer refer this: https://cwiki.apache.org/confluence/display/KAFKA/Consumer+Group+Example

Examples

```
## Not run:
consumer1=rkafka.createConsumer("127.0.0.1:2181","test123")
consumer2=rkafka.createConsumer("127.0.0.1:2181","test123","test-consumer-group","50000","1000")

## End(Not run)
```

rkafka.createProducer *Creating KAFKA producer*

Description

This function is used to create a KAFKA producer

Usage

```
rkafka.createProducer(metadataBrokerList, producerType="sync",
compressionCodec="none", serializerClass="kafka.serializer.StringEncoder",
partitionerClass="NULL", compressedTopics="NULL",
queueBufferingMaxTime="NULL", queueBufferingMaxMessages="NULL",
queueEnqueueTimeoutTime="NULL", batchNumMessages="NULL")
```

Arguments

`metadataBrokerList`
: List of brokers used for bootstrapping knowledge about the rest of the cluster format: host1:port1,host2:port2... Required:Mandatory Type:String default:localhost:9092

`producerType`
: specifies whether the messages are sent asynchronously (async) or synchronously (sync) Required:Mandatory Type:String default:sync

`compressionCodec`
: specify the compression codec for all data generated: none , gzip, snappy. Required:Mandatory Type:String default:none

`serializerClass`
: specifies the class for serialization Required:Mandatory Type:String default:kafka.serializer.StringEncode

`partitionerClass`
: name of the partitioner class for partitioning events Required:Optional Type:String default:NULL(default partition spreads data randomly)

`compressedTopics`
: allow topic level compression Required:Optional Type:String default:NULL

`queueBufferingMaxTime`
: maximum time, in milliseconds, for buffering data on the producer queue Required:Optional(for Async Producer only) Type:String default:NULL

`queueBufferingMaxMessages`
: the maximum size of the blocking queue for buffering on the producer Required:Optional(for Async Producer only) Type:String default:NULL

`queueEnqueueTimeoutTime`
: 0: events will be enqueued immediately or dropped if the queue is full -ve: enqueue will block indefinitely if the queue is full +ve: enqueue will block up to this many milliseconds if the queue is full Required:Optional(for Async Producer only) Type:String default:NULL

`batchNumMessages`
: the number of messages batched at the producer Required:Optional(for Async Producer only) Type:String default:NULL

Value

Returns Producer

Author(s)

Shruti Gupta

Examples

```
## Not run:
producer1=rkafka.createProducer("127.0.0.1:9092")
producer2=rkafka.createProducer("127.0.0.1:9092","sync","none","kafka.serializer.StringEncoder")

## End(Not run)
```

---

rkafka.createSimpleConsumer

Creating simple KAFKA consumer

Description

This function creates the Simple Consumer

Usage

```
rkafka.createSimpleConsumer(kafkaServerURL,
kafkaServerPort, connectionTimeOut,
kafkaProducerBufferSize, clientId)
```

Arguments

kafkaServerURL	
kafkaServerPort	Port number of the KAFKA server Required:Mandatory Type:String
connectionTimeOut	Connection Timeout in ms Required:Mandatory Type:String
kafkaProducerBufferSize	Buffer size Required:Mandatory Type:String
clientId	ID of the client Required:Mandatory Type:String

Details

There are two types of KAFKA consumers:High-Level and Simple. This function creates the Simple Consumer. Use caution on deciding to use the Simple Consumer as it doesn't persist offset.

Value

Doesn't return anything

Note

Warning: Ensure to run the rkafka.receiveFromSimpleConsumer() function before executing the rkafka.runFromSimpleConsumer() function

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

Examples

```
## Not run:
consumer1=rkafka.createSimpleConsumer("172.25.1.78","9092","10000","100000","test")

## End(Not run)
```

---

rkafka.read *KAFKA consumer reading messages(single)*

Description

This function reads messages received by a KAFKA consumer. It fetches one message at a time

Usage

```
rkafka.read(ConsumerObj)
```

Arguments

ConsumerObj	Consumer through which messages are to be read Required:Mandatory Type:Consumer

Details

This function returns one message at a time from the topic to which the consumer is associated. If no new message is found with 'x' time(set by ConsumerTimeoutMs property), then it returns ""

Value

String

Note

Warning: Ensure to close the consumer after reading messages. Won't work correctly next time otherwise

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

To know how to use a high level consumer refer this: https://cwiki.apache.org/confluence/display/KAFKA/Consumer+Group+Example

Examples

```
## Not run:
consumer1=rkafka.createConsumer("127.0.0.1:2181","test123")
print(rkafka.read(consumer1)

## End(Not run)
```

---

rkafka.readFromSimpleConsumer

KAFKA Simple Consumer Reading

Description

This function returns one message at a time which are read by a KAFKA Simple Consumer

Usage

```
rkafka.readFromSimpleConsumer(SimpleConsumerObj)
```

Arguments

`SimpleConsumerObj`
: Consumer through which messages were received Required:Mandatory Type:Consumer

Details

There are two types of KAFKA consumers:High-Level and Simple. This function receives messages using the Simple Consumer. Use caution on deciding to use the Simple Consumer as it doesn't persist offset.The function rkafka.receiveFromSimpleConsumer needs to be executed before running this function

Value

String

Note

Warning:The function rkafka.receiveFromSimpleConsumer needs to be executed before running this function

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

Examples

```
## Not run:
consumer1=rkafka.createSimpleConsumer("172.25.1.78","9092","10000","100000","test")
rkafka.receiveFromSimpleConsumer(consumer1,"test","0","0","test-group")
print(rkafka.readFromSimpleConsumer(consumer1))

## End(Not run)
```

---

rkafka.readPoll *KAFKA consumer reading messages(batch)*

Description

This function reads messages received by a KAFKA consumer. It returns a batch of messages

Usage

```
rkafka.readPoll(ConsumerObj)
```

Arguments

ConsumerObj	Consumer through which messages are to be read Required:Mandatory Type:Consumer

Details

This function returns messages as a batch from the topic to which the consumer is associated. If no new message is found with 'x' time(set by ConsumerTimeoutMs property), then it returns ""

Value

Array of Strings

Note

Warning: Ensure to close the consumer after reading messages. Won't work correctly next time otherwise

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

To know how to use a high level consumer refer this: https://cwiki.apache.org/confluence/display/KAFKA/Consumer+Group+Example

Examples

```
## Not run:

consumer1=rkafka.createConsumer("127.0.0.1:2181","test123")
print(rkafka.readPoll(consumer1)

## End(Not run)
```

---

rkafka.receiveFromSimpleConsumer

KAKFA Simple Consumer receiving messages

Description

This function allows the KAKFA Simple Consumer to receive messages from a particular topic. However, this doesn't display the messages. To read the messages, use the rkafka.readFromSimpleConsumer function.

Usage

```
rkafka.receiveFromSimpleConsumer(SimpleConsumerObj,
topicName, partition, Offset, msgReadSize)
```

Arguments

SimpleConsumerObj	Simple Consumer object through which messages are to be read Required:Mandatory Type:SimpleConsumer
topicName	Name of the topic from where to read messages Required:Mandatory Type:String
partition	Partition Number Required:Mandatory Type:String
Offset	Offset Number Required:Mandatory Type:String
msgReadSize	Size of the message to be read Required:Mandatory Type:String

Details

There are two types of KAFKA consumers:High-Level and Simple. This function receives messages using the Simple Consumer. Use caution on deciding to use the Simple Consumer as it doesn't persist offset.This function needs to be run before executing the rkafka.readFromSimpleConsumer function

Value

Nothing

Note

Warning: Ensure to close the consumer after reading messages. Won't work correctly next time otherwise

Author(s)

Shruti Gupta

References

To know when to use simple consumer and when to use High-level Consumer, refer the url below: https://cwiki.apache.org/confluence/display/KAFKA/0.8.0+SimpleConsumer+Example

Examples

```
## Not run:
consumer1=rkafka.createSimpleConsumer("172.25.1.78","9092","10000","100000","test")
rkafka.receiveFromSimpleConsumer(consumer1,"test","0","0","test-group")

## End(Not run)
```

---

rkafka.send *KAFKA producer sending message*

Description

This function sends message to a particular name through a producer

Usage

```
rkafka.send(producer, topicName, ip, message)
```

Arguments

producer	Producer through which messages are to be sent Required:Mandatory Type:String
topicName	Topic to which messages are to be sent. If topicName doesn't exist, new topic is created Required:Mandatory Type:String
ip	ip on which producer is running Required:Mandatory Type:String
message	message to be sent Required:Mandatory Type:String

Value

Doesn't return a value

Author(s)

Shruti Gupta

Examples

```
## Not run:
producer1=rkafka.createProducer("127.0.0.1:9092")
rkafka.send(producer1,"test","127.0.0.1:9092","Testing")

## End(Not run)
```

Index