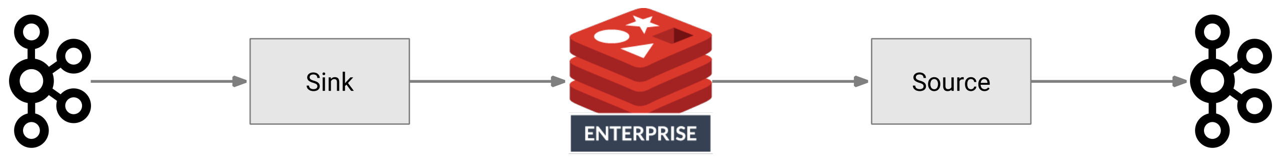
Redis Kafka Connector

Version 7.4

Table of Contents

Introduction

The Redis Kafka Connector is used to import and export data between Apache Kafka and Redis.



This guide provides documentation and usage information across the following topics:

- Install
- Connect to Redis
- Sink Connector
- Source Connector
- Docker Example
- Resources

Install

Select one of the methods below to install {name}.

Download

Download the latest release archive from here.

IMPORTANT

Confluent Platform Version

Download the connector archive that matches your version of Confluent Platform:

- Confluent Platform 5.0+: `redis-redis-enterprise-kafka-5.7.4.zip`
- Confluent Platform 6.0+: `redis-redis-enterprise-kafka-6.7.4.zip`

Confluent Hub

1. Install the Confluent Hub Client
2. Install the Redis Kafka Connector using the Confluent Hub Client

Manually

Follow the instructions in Manually Installing Community Connectors

Connect to Redis

This section provides information on configuring the Redis Kafka Source or Sink Connector.

Configuration

Specify the Redis URI in the `redis.uri` property, for example:

```
redis.uri=redis://redis-12000.redis.com:12000
```

Details on the Redis URI syntax can be found in the Lettuce project wiki.

TLS connection URIs start with `rediss://`. To disable certificate verification for TLS connections use the following property:

```
redis.insecure=true
```

Username and password can be specified in the URI or separately with the following properties:

```
redis.username=user1
redis.password=pass
```

Sink Connector Guide

The Redis Kafka Sink Connector consumes records from a Kafka topic and writes the data to Redis.

Features

The Redis Kafka Sink Connector includes the following features:

- At least once delivery
- Multiple tasks
- Redis Data Structures
- Supported Data Formats

At least once delivery

The Redis Kafka Sink Connector guarantees that records from the Kafka topic are delivered at least once.

Multiples tasks

The Redis Kafka Sink Connector supports running one or more tasks. You can specify the number of tasks with the `tasks.max` configuration property.

Redis Data Structures

The Redis Kafka Sink Connector supports the following Redis data-structure types as targets:

- Collections: stream, list, set, sorted set, time series

  Collection keys are generated using the `redis.key` configuration property which may contain `${topic}` (default) as a placeholder for the originating topic name.

  For example with `redis.key = ${topic}` and topic `orders` the Redis key is `set:orders`.

- Hash, string, JSON

  For other data-structures the key is in the form `<keyspace>:<record_key>` where `keyspace` is generated using the `redis.key` configuration property like above and `record_key` is the sink record key.

  For example with `redis.key = ${topic}`, topic `orders`, and sink record key `123` the Redis key is `orders:123`.

Hash

Use the following properties to write Kafka records as Redis hashes:

```
redis.type=HASH
```

```
key.converter=<string or bytes> ①
value.converter=<Avro or JSON> ②
```

① String or bytes

② Avro or JSON. If value is null the key is deleted.

String

Use the following properties to write Kafka records as Redis strings:

```
redis.type=STRING
key.converter=<string or bytes> ①
value.converter=<string or bytes> ②
```

① String or bytes

② String or bytes. If value is null the key is deleted.

JSON

Use the following properties to write Kafka records as RedisJSON documents:

```
redis.type=JSON
key.converter=<string, bytes, or Avro> ①
value.converter=<string, bytes, or Avro> ②
```

① String, bytes, or Avro

② String, bytes, or Avro. If value is null the key is deleted.

Stream

Use the following properties to store Kafka records as Redis stream messages:

```
redis.type=STREAM
redis.key=<stream key> ①
value.converter=<Avro or JSON> ②
```

① Stream key

② Avro or JSON

List

Use the following properties to add Kafka record keys to a Redis list:

```
redis.type=LIST
redis.key=<key name> ①
key.converter=<string or bytes> ②
```

```
redis.push.direction=<LEFT or RIGHT> ③
```

① List key

② String or bytes: Kafka record keys to push to the list

③ `LEFT`: LPUSH (default), `RIGHT`: RPUSH

The Kafka record value can be any format. If a value is null then the member is removed from the list (instead of pushed to the list).

Set

Use the following properties to add Kafka record keys to a Redis set:

```
redis.type=SET
redis.key=<key name> ①
key.converter=<string or bytes> ②
```

① Set key

② String or bytes: Kafka record keys to add to the set

The Kafka record value can be any format. If a value is null then the member is removed from the set (instead of added to the set).

Sorted Set

Use the following properties to add Kafka record keys to a Redis sorted set:

```
redis.type=ZSET
redis.key=<key name> ①
key.converter=<string or bytes> ②
```

① Sorted set key

② String or bytes: Kafka record keys to add to the set

The Kafka record value should be `float64` and is used for the score. If the score is null then the member is removed from the sorted set (instead of added to the sorted set).

Time Series

Use the following properties to write Kafka records as RedisTimeSeries samples:

```
redis.type=TIMESERIES
redis.key=<key name> ①
```

① Timeseries key

The Kafka record key must be an integer (e.g. `int64`) as it is used for the sample time in milliseconds.

The Kafka record value must be a number (e.g. `float64`) as it is used as the sample value.

Data Formats

The Redis Kafka Sink Connector supports different data formats for record keys and values depending on the target Redis data structure.

Kafka Record Keys

The Redis Kafka Sink Connector expects Kafka record keys in a specific format depending on the configured target Redis data structure:

Target	Record Key	Assigned To
Stream	Any	None
Hash	String	Key
String	String or bytes	Key
List	String or bytes	Member
Set	String or bytes	Member
Sorted Set	String or bytes	Member
JSON	String or bytes	Key
TimeSeries	Integer	Sample time in milliseconds

StringConverter

If record keys are already serialized as strings use the StringConverter:

```
key.converter=org.apache.kafka.connect.storage.StringConverter
```

ByteArrayConverter

Use the byte array converter to use the binary serialized form of the Kafka record keys:

```
key.converter=org.apache.kafka.connect.converters.ByteArrayConverter
```

Kafka Record Values

Multiple data formats are supported for Kafka record values depending on the configured target Redis data structure. Each data structure expects a specific format. If your data in Kafka is not in the format expected for a given data structure, consider using Single Message Transformations to convert to a byte array, string, Struct, or map before it is written to Redis.

Target	Record Value	Assigned To
Stream	Avro or JSON	Message body
Hash	Avro or JSON	Fields
String	String or bytes	Value
List	Any	Removal if null
Set	Any	Removal if null
Sorted Set	Number	Score or removal if null
JSON	String or bytes	Value
TimeSeries	Number	Sample value

StringConverter

If record values are already serialized as strings, use the StringConverter to store values in Redis as strings:

```
value.converter=org.apache.kafka.connect.storage.StringConverter
```

ByteArrayConverter

Use the byte array converter to store the binary serialized form (for example, JSON, Avro, Strings, etc.) of the Kafka record values in Redis as byte arrays:

```
value.converter=org.apache.kafka.connect.converters.ByteArrayConverter
```

Avro

```
value.converter=io.confluent.connect.avro.AvroConverter
value.converter.schema.registry.url=http://localhost:8081
```

JSON

```
value.converter=org.apache.kafka.connect.json.JsonConverter
value.converter.schemas.enable=<true|false> ①
```

① Set to `true` if the JSON record structure has an attached schema

Source Connector Guide

The Redis Kafka Source Connector reads from a Redis stream and publishes messages to a Kafka topic.

Features

The Redis Kafka Source Connector includes the following features:

- At least once delivery
- Multiple tasks
- Stream Reader

Delivery Guarantees

The Redis Kafka Source Connector can be configured to ack stream messages either automatically (at-most-once delivery) or explicitly (at-least-once delivery). The default is at-least-once delivery.

At-Least-Once

In this mode, each stream message is acknowledged after it has been written to the corresponding topic.

```
redis.stream.delivery=at-least-once
```

At-Most-Once

In this mode, stream messages are acknowledged as soon as they are read.

```
redis.stream.delivery=at-most-once
```

Multiple Tasks

Use configuration property `tasks.max` to have the change stream handled by multiple tasks. The connector splits the work based on the number of configured key patterns. When the number of tasks is greater than the number of patterns, the number of patterns will be used instead.

For example `foo:*` translates to pubsub channel `__keyspace@0__:foo:*` and will capture changes to keys `foo:1`, `foo:2`, etc. Use comma-separated values for multiple patterns (`foo:*,bar:*`)

Stream Reader

The Redis Kafka Source Connector reads messages from a stream and publishes to a Kafka topic. Reading is done through a consumer group so that multiple instances of the connector configured via the `tasks.max` can consume messages in a round-robin fashion.

Stream Message Schema

Key Schema

Keys are of type String and contain the stream message id.

Value Schema

The value schema defines the following fields:

Name	Schema	Description
id	STRING	Stream message ID
stream	STRING	Stream key
body	Map of STRING	Stream message body

Configuration

```
redis.stream.name=<name> ①
redis.stream.offset=<offset> ②
redis.stream.block=<millis> ③
redis.stream.consumer.group=<group> ④
redis.stream.consumer.name=<name> ⑤
topic=<name> ⑥
```

① Name of the stream to read from.

② Message ID to start reading from (default: `0-0`).

③ Maximum XREAD wait duration in milliseconds (default: `100`).

④ Name of the stream consumer group (default: `kafka-consumer-group`).

⑤ Name of the stream consumer (default: `consumer-${task}`). May contain `${task}` as a placeholder for the task id. For example, `foo${task}` and task `123` ⇒ consumer `foo123`.

⑥ Destination topic (default: `${stream}`). May contain `${stream}` as a placeholder for the originating stream name. For example, `redis_${stream}` and stream `orders` ⇒ topic `redis_orders`.

Quick Start with Docker

This guide provides a hands-on look at the functionality of the Redis Kafka Source and Sink Connectors:

- The **redis-sink** connector reads data from a Kafka topic and writes it to a Redis stream
- The **redis-source** connector reads data from a Redis stream and writes it to a Kafka topic

Requirements

Docker

Run the example

Clone the redis-kafka-connect repository and execute `run.sh` in `docker` directory:

```
git clone https://github.com/redis-field-engineering/redis-kafka-connect.git
./run.sh
```

This will:

- Run `docker-compose up`
- Wait for Redis, Kafka, and Kafka Connect to be ready
- Register the Confluent Datagen Connector
- Register the Redis Kafka Sink Connector
- Register the Redis Kafka Source Connector
- Publish some events to Kafka via the Datagen connector
- Write the events to Redis
- Send messages to a Redis stream
- Write the Redis stream messages back into Kafka

Once running, examine the topics in the Kafka control center: http://localhost:9021/

- The `pageviews` topic should contain the 10 simple documents added, each similar to:

```
{
  "viewtime": {
    "$numberLong": "81"
  },
  "pageid": "Page_1",
  "userid": "User_8"
}
```

- The `pageviews` stream should contain the 10 change events.

Examine the stream in Redis:

```
docker-compose exec redis /usr/local/bin/redis-cli
xread COUNT 10 STREAMS pageviews 0
```

Messages added to the `mystream` stream will show up in the `mystream` topic

Resources

Kafka

What is Apache Kafka?

https://youtu.be/06iRM1Ghr1k

Should You Put Several Event Types in the Same Kafka Topic?

https://www.confluent.io/blog/put-several-event-types-kafka-topic/

Kafka Quickstart

https://kafka.apache.org/quickstart

Console Producer and Consumer Basics

https://kafka-tutorials.confluent.io/kafka-console-consumer-producer-basics/kafka.html

Kafka Connect

Introduction to Kafka Connectors

https://www.baeldung.com/kafka-connectors-guide

Kafka Connect Documentation

https://docs.confluent.io/platform/current/connect/index.html

Redis

Redis

https://redis.io/topics/introduction

Redis Streams

https://redis.io/topics/streams-intro

Redis Enterprise Advantages

https://redis.com/redis-enterprise/advantages/