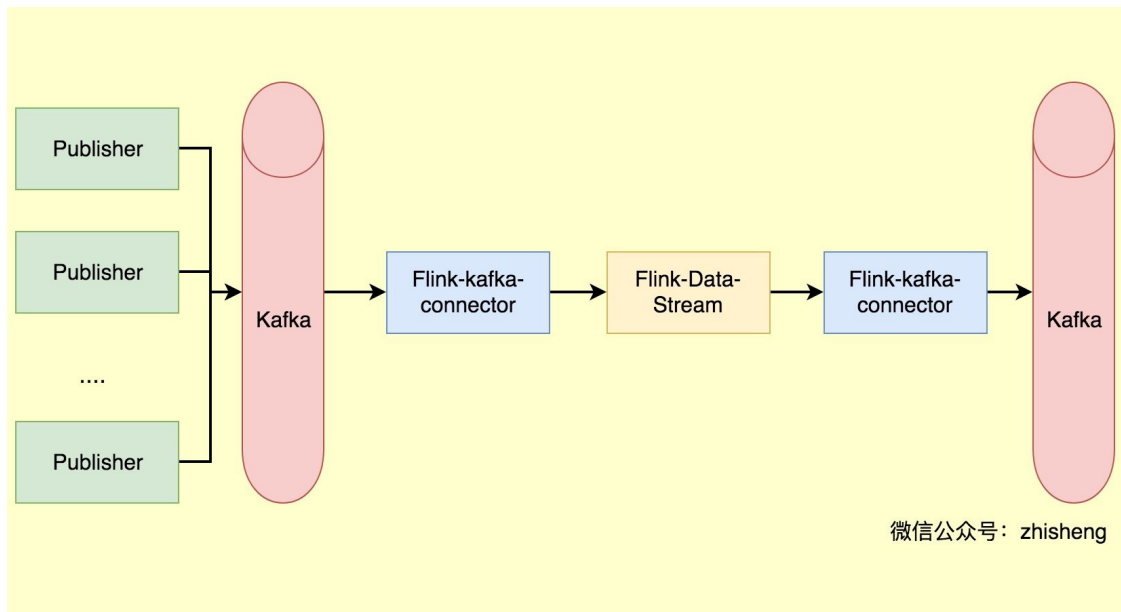

toc: true title: 《从0到1学习Flink》—— Flink 写入数据到 Kafka date: 2019-01-06
tags:

- Flink
 - 大数据
 - 流式计算
 - Kafka
-



前言

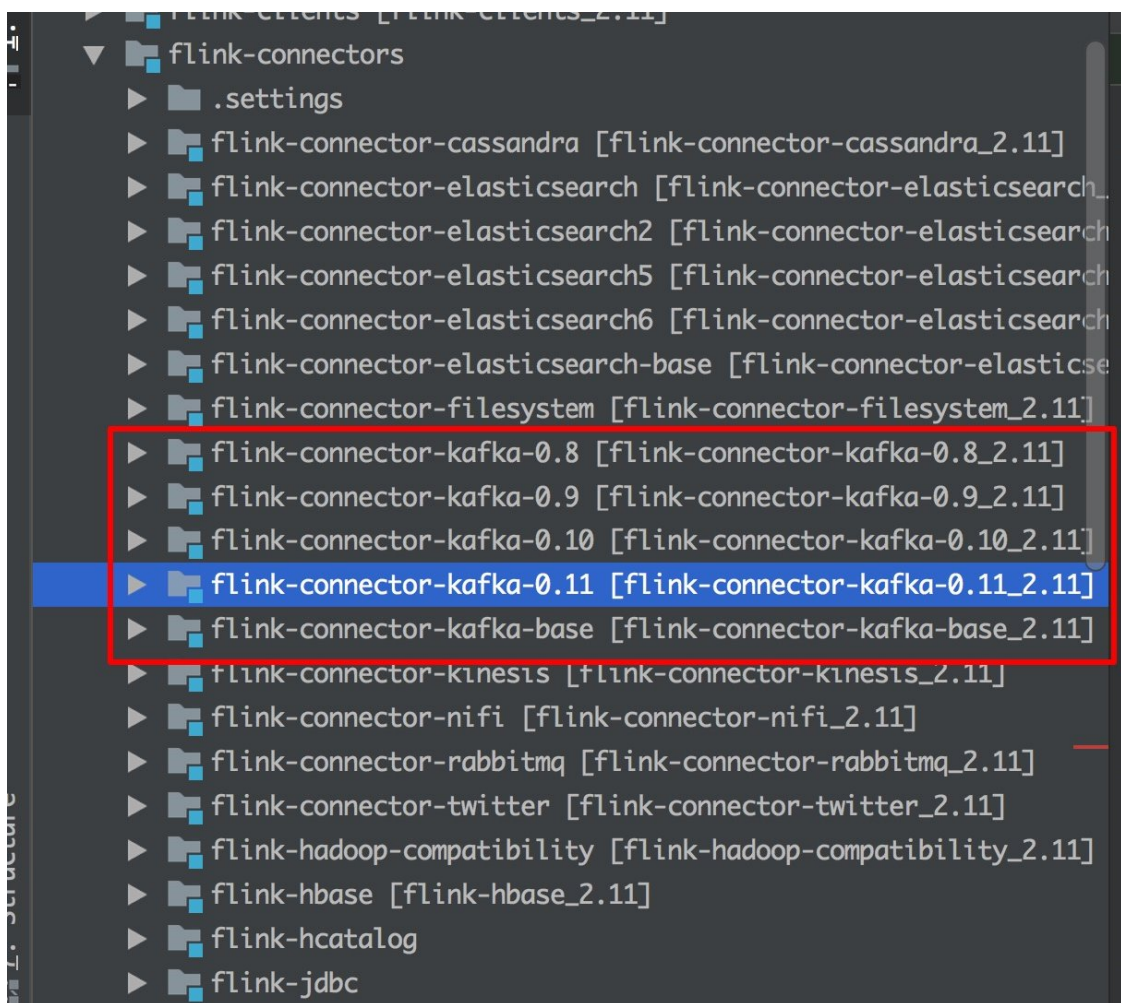
之前文章 [《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch](#) 写了如何将 Kafka 中的数据存储到 ElasticSearch 中，里面其实就已经用到了 Flink 自带的 Kafka source connector (FlinkKafkaConsumer)。存入到 ES 只是其中一种情况，那么如果我们有多个地方需要这份通过 Flink 转换后的数据，是不是又要我们继续写个 sink 的插件呢？确实，所以 Flink 里面就默认支持了不少 sink，比如也支持 Kafka sink connector (FlinkKafkaProducer)，那么这篇文章我们就讲讲如何将数据写入到 Kafka。



准备

添加依赖

Flink 里面支持 Kafka 0.8、0.9、0.10、0.11，以后有时间可以分析下源码的实现。



这里我们需要安装下 Kafka，请对应添加对应的 Flink Kafka connector 依赖的版本，这里我们使用的是 0.11 版本：

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kafka-0.11_2.11</artifactId>
  <version>${flink.version}</version>
</dependency>
```

Kafka 安装

这里就不写这块内容了，可以参考我以前的文章 [Kafka 安装及快速入门](#)。

这里我们演示把其他 Kafka 集群中 topic 数据原样写入到自己本地起的 Kafka 中去。

配置文件

```
kafka.brokers=xxx:9092,xxx:9092,xxx:9092
kafka.group.id=metrics-group-test
kafka.zookeeper.connect=xxx:2181
metrics.topic=xxx
stream.parallelism=5
kafka.sink.brokers=localhost:9092
kafka.sink.topic=metric-test
stream.checkpoint.interval=1000
stream.checkpoint.enable=false
stream.sink.parallelism=5
```

目前我们先看下本地 Kafka 是否有这个 metric-test topic 呢？需要执行下这个命令：

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

[illegible]

可以看到本地的 Kafka 是没有任何 topic 的，如果等下我们的程序运行起来后，再次执行这个命令出现 metric-test topic，那么证明我的程序确实起作用了，已经将其他集群的 Kafka 数据写入到本地 Kafka 了。

程序代码

Main.java

```

public class Main {
    public static void main(String[] args) throws Exception{
        final ParameterTool parameterTool =
            ExecutionEnvUtil.createParameterTool(args);
        StreamExecutionEnvironment env =
            ExecutionEnvUtil.prepare(parameterTool);
        DataStreamSource<Metrics> data =
            KafkaConfigUtil.buildSource(env);

        data.addSink(new FlinkKafkaProducer011<Metrics>(
            parameterTool.get("kafka.sink.brokers"),
            parameterTool.get("kafka.sink.topic"),
            new MetricSchema()
        )).name("flink-connectors-kafka")

        .setParallelism(parameterTool.getInt("stream.sink.parallelism"));

        env.execute("flink learning connectors kafka");
    }
}

```

运行结果

启动程序，查看运行结果，不段执行上面命令，查看是否有新的 topic 出来：

```

zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng > /usr/local/kafka_2.11-1.1.0

```

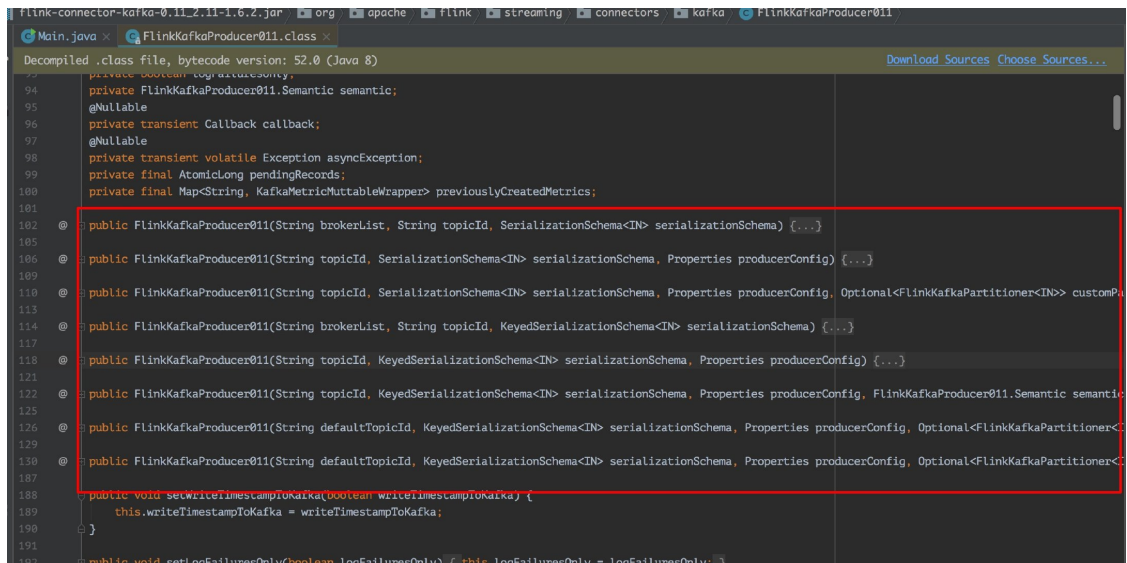
执行命令可以查看该 topic 的信息：


```
bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic metric-test
```

```
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --list --zookeeper localhost:2181
metric-test
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic metric-test
Topic: metric-test PartitionCount:1 ReplicationFactor:1 Configs:
Topic: metric-test Partition: 0 Leader: 0 Replicas: 0 Isr: 0
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic metric-test
Topic: metric-test PartitionCount:1 ReplicationFactor:1 Configs:
Topic: metric-test Partition: 0 Leader: 0 Replicas: 0 Isr: 0
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0 bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic metric-test
Topic: metric-test PartitionCount:1 ReplicationFactor:1 Configs:
Topic: metric-test Partition: 0 Leader: 0 Replicas: 0 Isr: 0
zhisheng@zhisheng /usr/local/kafka_2.11-1.1.0
```

分析

上面代码我们使用 Flink Kafka Producer 只传了三个参数：brokerList、topicId、serializationSchema（序列化）



The screenshot shows the decompiled Java code for the `FlinkKafkaProducer011` class. Several constructor methods are visible, with the first three highlighted by a red box:

- `public FlinkKafkaProducer011(String brokerList, String topicId, SerializationSchema<IN> serializationSchema) {...}`
- `public FlinkKafkaProducer011(String topicId, SerializationSchema<IN> serializationSchema, Properties producerConfig) {...}`
- `public FlinkKafkaProducer011(String topicId, SerializationSchema<IN> serializationSchema, Properties producerConfig, Optional<FlinkKafkaPartitioner<IN>> customPartitioner) {...}`

Other constructors and methods like `setWriteTimestampToKafka` and `setLogFailuresOnly` are also visible below the highlighted section.

其实也可以传入多个参数进去，现在有的参数用的是默认参数，因为这个内容比较多，后面可以抽出一篇文章单独来讲。

总结

本篇文章写了 Flink 读取其他 Kafka 集群的数据，然后写入到本地的 Kafka 上。我在 Flink 这层没做什么数据转换，只是原样的将数据转发了下，如果你们有什么其他的需求，是可以在 Flink 这层将数据进行各种转换操作，比如这篇文章中的一些转换：《从0到1学习Flink》—— [Flink Data transformation\(转换\)](#)，然后将转换后的数据发到 Kafka 上去。

本文原创地址是：<http://www.54tianzhisheng.cn/2019/01/06/Flink-Kafka-sink/>，未经允许禁止转载。

关注我

微信公众号：zhisheng

另外我自己整理了些 Flink 的学习资料，目前已经全部放到微信公众号了。你可以加我的微信：zhisheng_tian，然后回复关键字：Flink 即可无条件获取到。



相关文章

- 1、《从0到1学习Flink》—— Apache Flink 介绍
- 2、《从0到1学习Flink》—— Mac 上搭建 Flink 1.6.0 环境并构建运行简单程序入门
- 3、《从0到1学习Flink》—— Flink 配置文件详解
- 4、《从0到1学习Flink》—— Data Source 介绍
- 5、《从0到1学习Flink》—— 如何自定义 Data Source ?
- 6、《从0到1学习Flink》—— Data Sink 介绍
- 7、《从0到1学习Flink》—— 如何自定义 Data Sink ?
- 8、《从0到1学习Flink》—— Flink Data transformation(转换)
- 9、《从0到1学习Flink》—— 介绍Flink中的Stream Windows
- 10、《从0到1学习Flink》—— Flink 中的几种 Time 详解
- 11、《从0到1学习Flink》—— Flink 写入数据到 ElasticSearch
- 12、《从0到1学习Flink》—— Flink 项目如何运行?
- 13、《从0到1学习Flink》—— Flink 写入数据到 Kafka