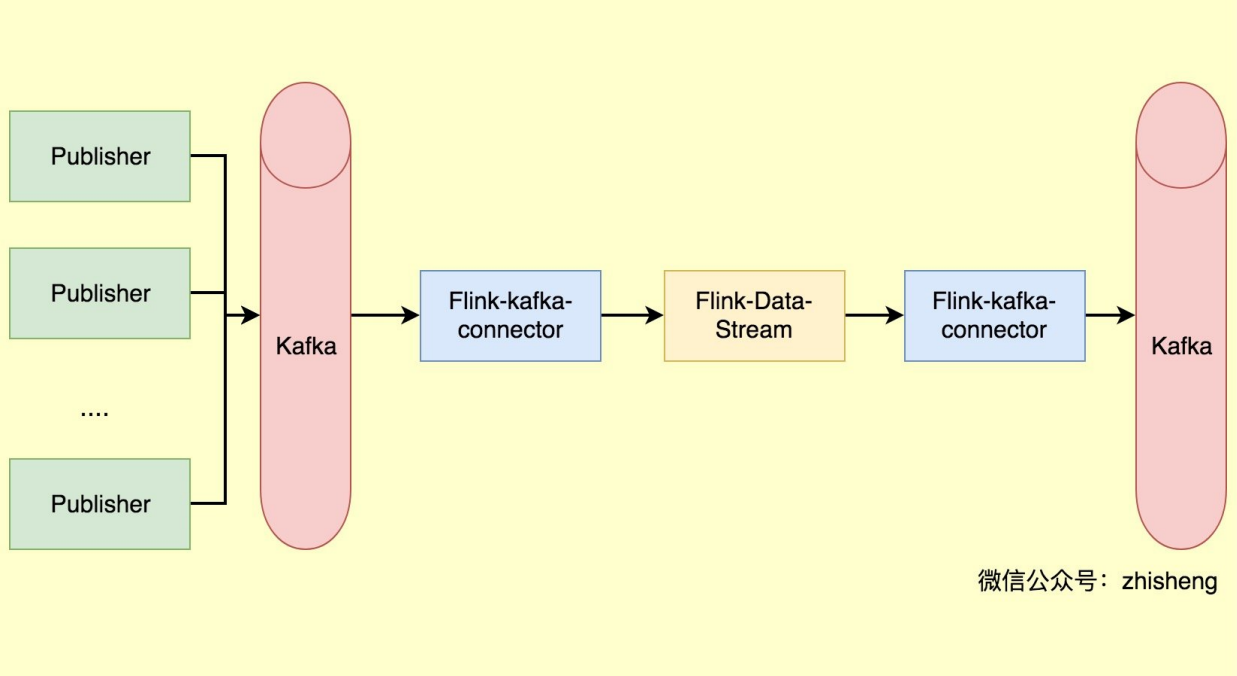
前言

在前面文章 Flink 常用的 Source 和 Sink Connectors 介绍 中我讲解了 Flink 中的 Data Source 和 Data Sink，然后介绍了 Flink 中自带的一些 Source 和 Sink 的 Connector。本篇文章我将讲解一下几乎用的最多的 Connector —— Kafka，带大家利用 Kafka Connector 读取 Kafka 数据，做一些计算操作后然后又通过 Kafka Connector 写入到 kafka 消息队列去。



Kafka Source Connector 上手

准备工作

我们先来看下 Flink 从 Kafka topic 中获取数据的 demo，首先你需要安装好了 Flink 和 Kafka 。

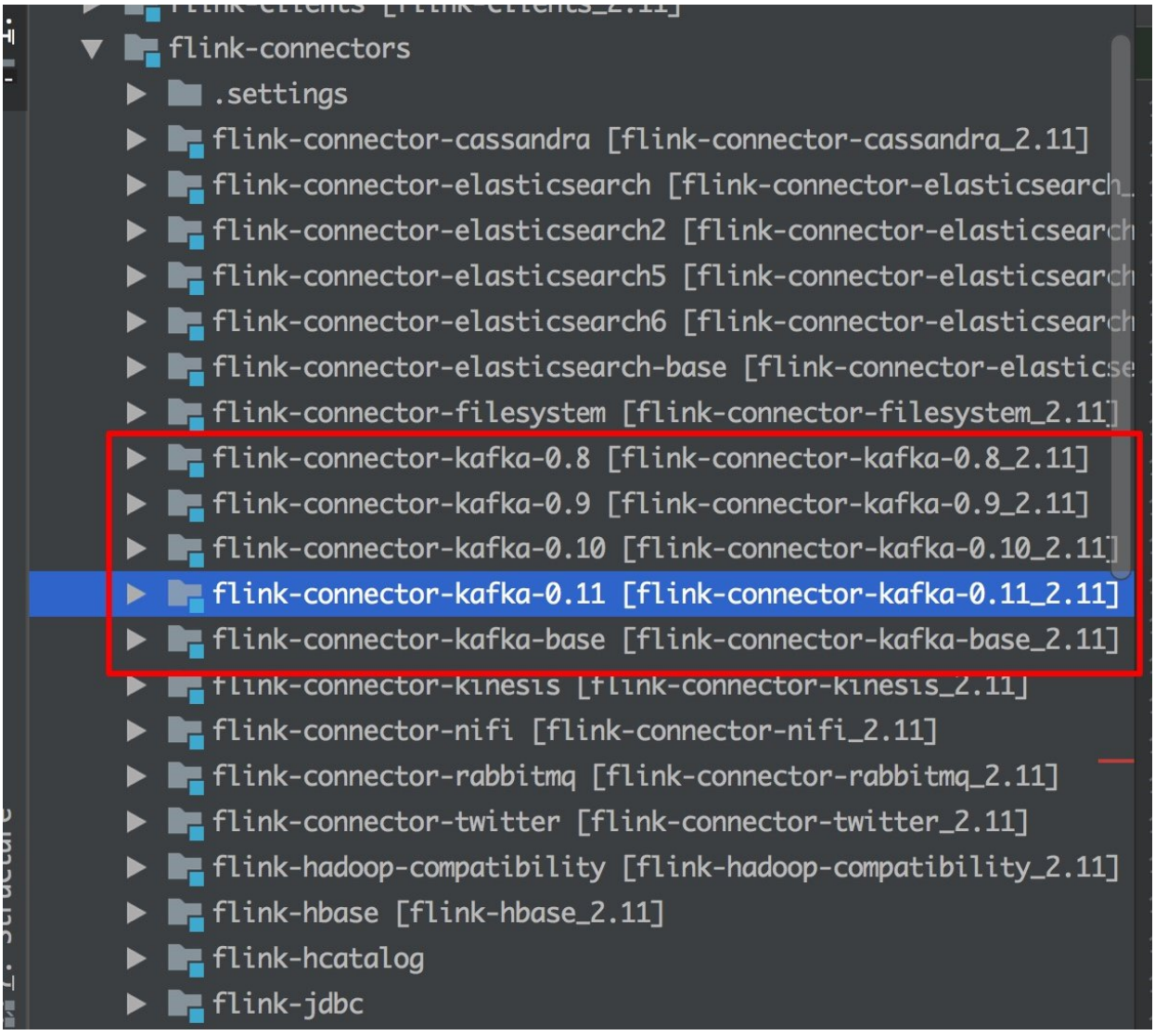
如果你已经安装好了 Flink 和 Kafka，那么接下来运行启动 Flink、Zookepeer、Kafka 就行了。

```
flink-console.sh          jobmanager.sh             mesos-taskmanager.sh      sql-client.sh             start-scala-shell.sh
r-quorum.sh  zookeeper.sh
 zhisheng@zhisheng /usr/local/Cellar/apache-flink/1.6.0/libexec/bin ./start-cluster.sh
Starting cluster.
Starting standalonesession daemon on host zhisheng.
Starting taskexecutor daemon on host zhisheng.
 zhisheng@zhisheng /usr/local/Cellar/apache-flink/1.6.0/libexec/bin ls
config.sh                 jobmanager.sh             pyflink.sh                start-zookeeper-quorum.sh zookeeper.sh
flink                     mesos-appmaster-job.sh    sql-client.sh             stop-cluster.sh
flink-console.sh          mesos-appmaster.sh        standalone-job.sh         stop-zookeeper-quorum.sh
flink-daemon.sh           mesos-taskmanager.sh      start-cluster.sh          taskmanager.sh
historyserver.sh          pyflink-stream.sh         start-scala-shell.sh      yarn-session.sh
 zhisheng@zhisheng /usr/local/Cellar/apache-flink/1.6.0/libexec/bin
 zhisheng@zhisheng /usr/local/Cellar/apache-flink/1.6.0/libexec/bin
```

```
                                                given to allow fail-over.
 ✘ zhisheng@zhisheng  /usr/local/kafka_2.11-1.1.0  ls
LICENSE   NOTICE    bin     config    libs     logs     run.sh    site-docs
 zhisheng@zhisheng  /usr/local/kafka_2.11-1.1.0
 zhisheng@zhisheng  /usr/local/kafka_2.11-1.1.0  cat run.sh
#!/bin/bash
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
bin/kafka-server-start.sh  config/server.properties
# bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic metrics
 zhisheng@zhisheng  /usr/local/kafka_2.11-1.1.0
```

好了，都启动了！

Maven 依赖



Flink 里面支持 Kafka 0.8、0.9、0.10、0.11 以及更高版本，我们采用哪个版本的 Maven 依赖需要根据我们自己安装的 Kafka 版本。因为之前我们安装的 Kafka 是 0.11 版本，所以这里我们选择的 Kafka Connector 为 `flink-connector-kafka-0.11_2.11` 。

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

另外你还要添加的依赖有：

```xml
<!--flink java-->
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-java</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_${scala.binary.version}</artifactId>
    <version>${flink.version}</version>
    <scope>provided</scope>
</dependency>

<!--log-->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.7</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
    <scope>runtime</scope>
</dependency>

<!--alibaba fastjson-->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.51</version>
</dependency>
```

测试发送数据到 kafka topic

实体类，Metric.java

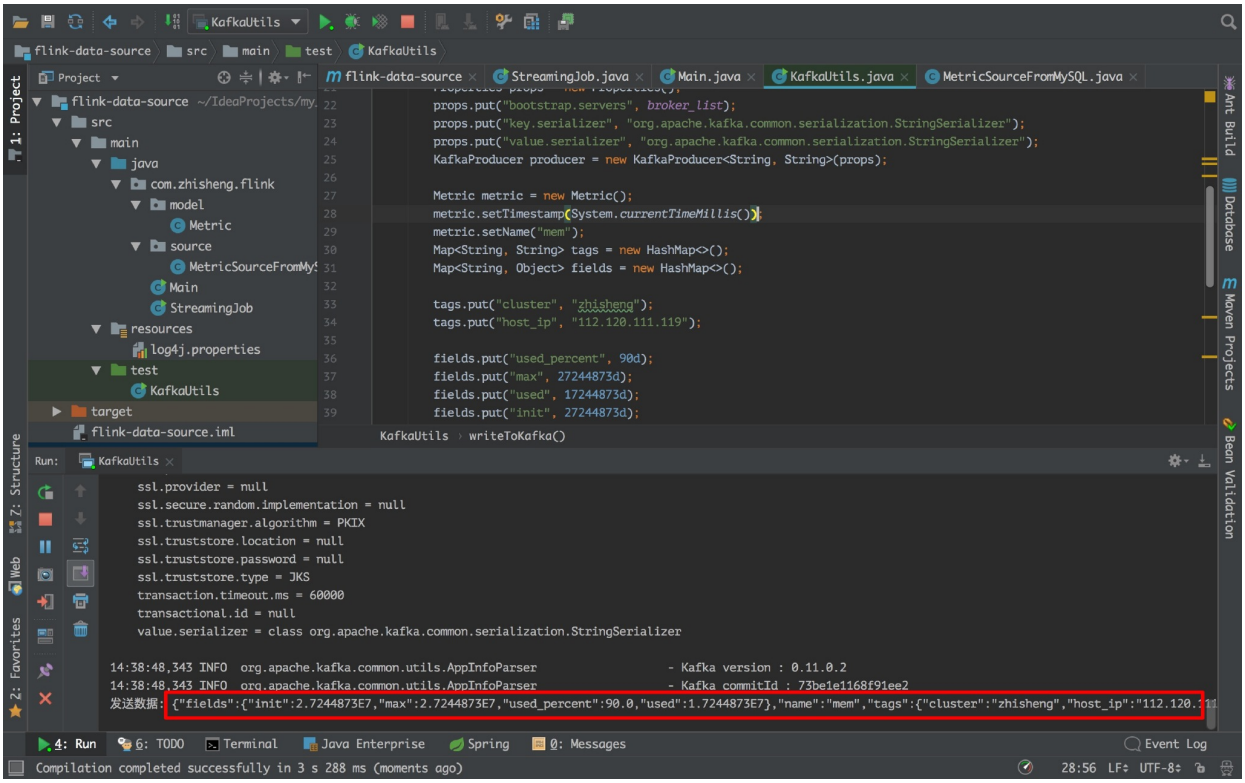
```java
package com.zhisheng.flink.model;

import java.util.Map;

/**
 * blog: http://www.54tianzhisheng.cn/
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Metric {
    public String name;
    public long timestamp;
    public Map<String, Object> fields;
    public Map<String, String> tags;
}
```

往 kafka 中写数据工具类：KafkaUtils.java

```
import com.alibaba.fastjson.JSON;
import com.zhisheng.flink.model.Metric;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.HashMap;
import java.util.Map;
import java.util.Properties;

/**
 * 往kafka中写数据
 * 可以使用这个main函数进行测试一下
 * blog: http://www.54tianzhisheng.cn/
 */
public class KafkaUtils {
    public static final String broker_list = "localhost:9092";
    public static final String topic = "metric";  // kafka topic, Flink 程序中需要和这个统一

    public static void writeToKafka() throws InterruptedException {
        Properties props = new Properties();
        props.put("bootstrap.servers", broker_list);
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer"); //key 序列化
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer"); //value 序列化
        KafkaProducer producer = new KafkaProducer<String, String>(props);

        Metric metric = new Metric();
        metric.setTimestamp(System.currentTimeMillis());
        metric.setName("mem");
        Map<String, String> tags = new HashMap<>();
        Map<String, Object> fields = new HashMap<>();

        tags.put("cluster", "zhisheng");
        tags.put("host_ip", "101.147.022.106");

        fields.put("used_percent", 90d);
        fields.put("max", 27244873d);
        fields.put("used", 17244873d);
        fields.put("init", 27244873d);

        metric.setTags(tags);
        metric.setFields(fields);

        ProducerRecord record = new ProducerRecord<String, String>(topic, null, null, JSON.toJSONString(metric));
        producer.send(record);
        System.out.println("发送数据： " + JSON.toJSONString(metric));

        producer.flush();
    }

    public static void main(String[] args) throws InterruptedException {
        while (true) {
            Thread.sleep(300);
```

```
                writeToKafka();
            }
        }
}
```

运行：



如果出现如上图标记的，即代表能够不断往 kafka 发送数据的。

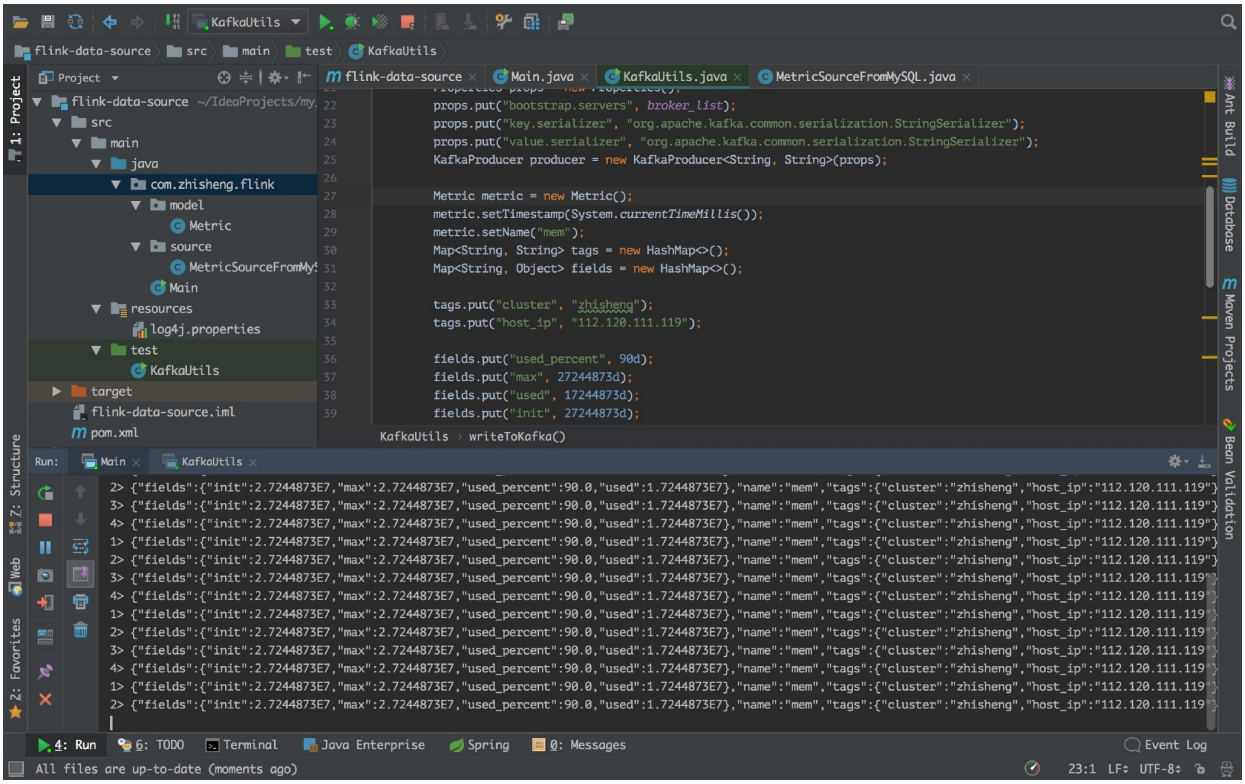
Flink 程序

Main.java

```
package com.zhisheng.flink;

import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStreamSource;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer011;

import java.util.Properties;

/**
 * blog: http://www.54tianzhisheng.cn/
 */
public class Main {
    public static void main(String[] args) throws Exception {
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

```

```
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("zookeeper.connect", "localhost:2181");
        props.put("group.id", "metric-group");
        props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");  //key 反序列化
        props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        props.put("auto.offset.reset", "latest"); //value 反序列化

        DataStreamSource<String> dataStreamSource = env.addSource(new
FlinkKafkaConsumer011<>(
                "metric",  //kafka topic
                new SimpleStringSchema(),  // String 序列化
                props)).setParallelism(1);

        dataStreamSource.print(); //把从 kafka 读取到的数据打印在控制台

        env.execute("Flink add data source");
    }
}
```

运行起来：



看到没程序，Flink 程序控制台能够源源不断地打印数据呢。

代码分析

使用 FlinkKafkaConsumer011 时传入了三个参数。

- Kafka topic：这个代表了 Flink 要消费的是 Kafka 哪个 Topic，如果你要同时消费多个 Topic 的话，那么你可以传入一个 Topic List 进去，另外也支持正则表达式匹配 Topic
- 序列化：上面代码我们使用的是 SimpleStringSchema
- 配置属性：将 Kafka 等的一些配置传入

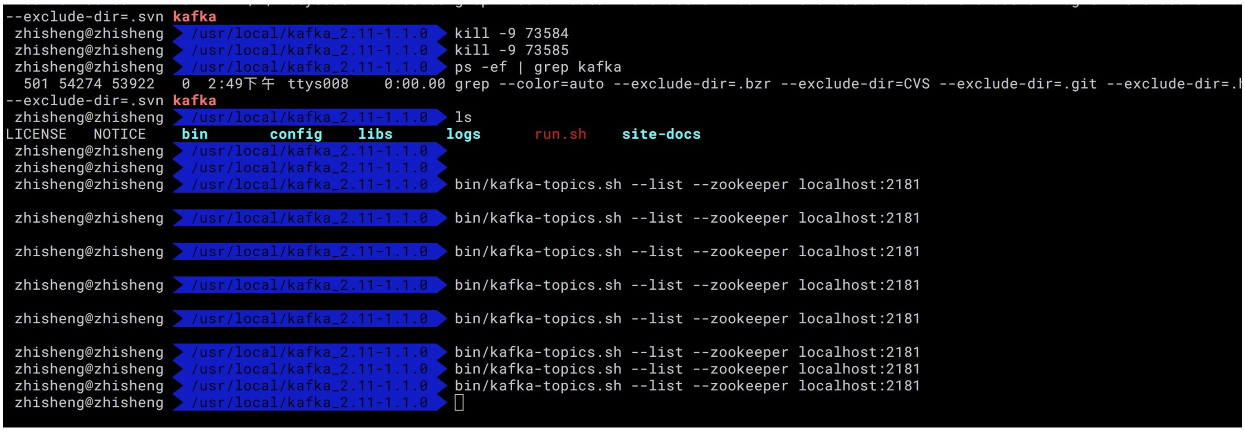
这里我们接着演示把其他 Kafka 集群中 topic 数据原样写入到自己本地起的 Kafka 中去。

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
```



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。

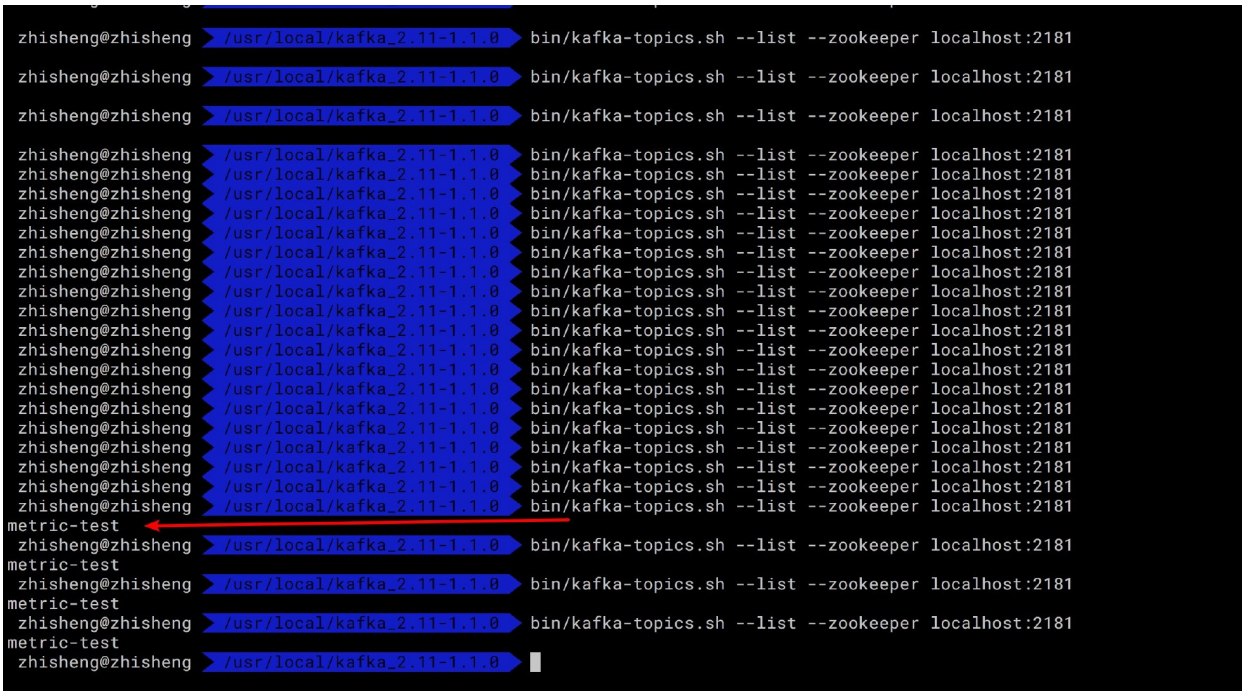
程序代码

Main.java

```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");
    }
}
```

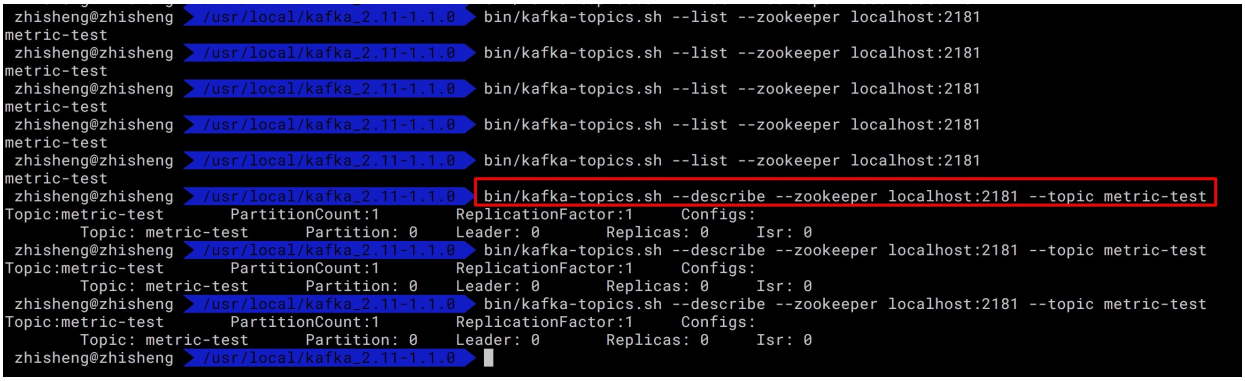
运行结果

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：



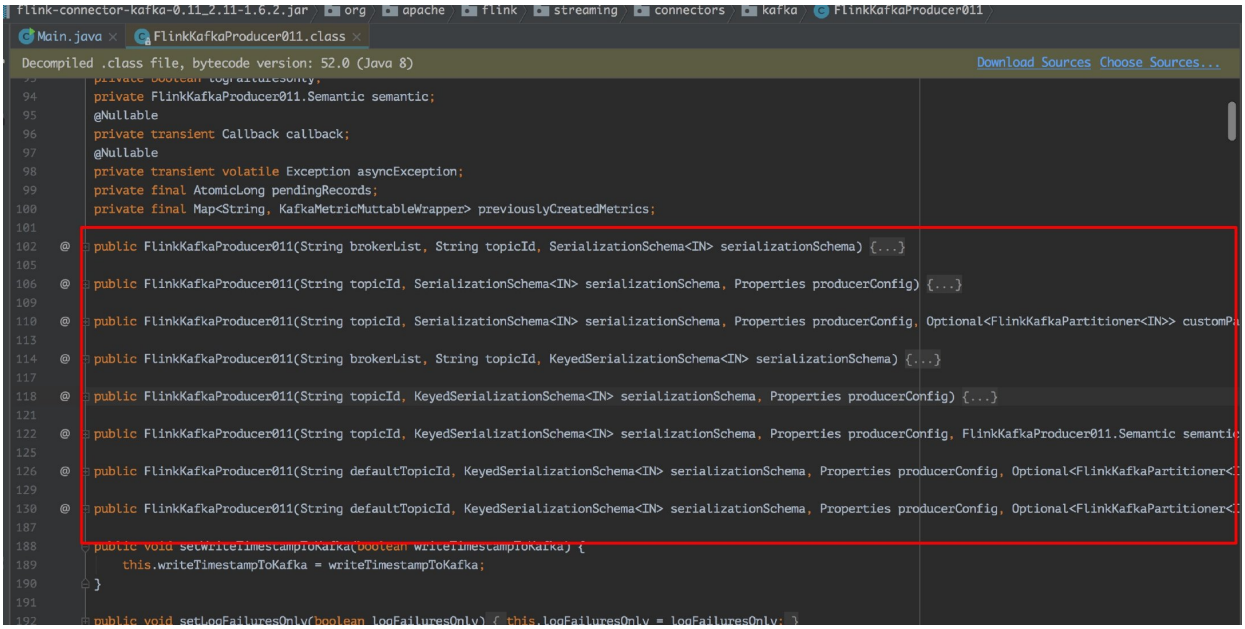
执行命令可以查看该 topic 的信息：

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



Flink Kafka Producer 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）



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。如果你们有其他的需求，是可以在 Flink 这层将数据进行各种转换操作的，比如这篇文章中的一些转换：1Flink 数据转换必须熟悉的算子（Operator），然后将转换后的数据发到 Kafka 上去。

Github 代码仓库

https://github.com/zhisheng17/flink-learning/tree/master/flink-learning-connectors/flink-learning-connectors-kafka