# 1.Create Execution Environment

(1) **StreamExecutionEnvironment.getExecutionEnvironment()**

The above code includes the following two functions:

**StreamExecutionEnvironment.createLocalEnvironment().**

**StreamExecutionEnvironment.createRemoteEnvironment().**

(2) Execution Mode

There are two kinds of execution modes: batch mode and stream mode. Use **ExecutionEnvironment.getExecutionEnvironment()** to start batch mode or use **StreamExecutionEnvironment.getExecutionEnvironment()** to start stream mode.

The two modes can be unified in new Flink: when running the program, type: **bin/flink run -Dexecution.runtime=BATCH** to start batch mode, if not, stream mode will be started.

Besides, it is also possible to change mode in our codes. After creating execution environment, just type: **env.setRuntimeMode(RuntimeExecutionMode.BATCH).**

# 2.Source Operators

Different operators can be used to read the input data stream. In Java program “event” and “sourcetest”, four basic methods and Kafka are showed and tested, the first three of them belong to batch mode, while the last two are stream mode. For the first four kinds of data input, I just need to run the program, while for Kafka, more settings should be taken into consideration.

Flink often works with Kafka, which is widely used for obtaining and sending data. In this program, I show how to start Kafka with Flink.

First, a new source is needed to be created, so just type: **env.addSource(new FlinkKafkaConsumer<String>("clicks", new SimpleStringSchema(), properties)).** In this sentence, there are three parameters, the first is the name, the second is a kind of type (I don’t understand), and the last one is the properties we need to set. Then some properties should be set. Generally, the following sentences are necessary and enough for Kafka and Flink:

**Properties properties = new Properties()**

**properties.setProperty("bootstrap.servers","node1:9092");**

**properties.setProperty("group.id","consumer-group"); properties.setProperty("key.deserializer","org.apache.kafka.common.serialization.StringDeserializer"); properties.setProperty("value.deserializer","org.apache.kafka.common.serialization.StringDeserializer");**

**properties.setProperty("auto.offset.reset","latest");**