Code

FlinkKafkaProducer.java

```
package wikiedits;

import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.common.functions.ReduceFunction;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.streaming.api.datastream.DataStream;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import
org.apache.flink.streaming.api.windowing.assigners.TumblingEventTimeWindow
s;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaProducer011;
import
org.apache.flink.streaming.connectors.kafka.partitioner.FlinkKafkaPartitio
ner;
import org.apache.flink.streaming.connectors.wikiedits.WikipediaEditEvent;
import
org.apache.flink.streaming.connectors.wikiedits.WikipediaEditsSource;
import org.apache.flink.util.Preconditions;

import java.util.Optional;

import static
org.apache.flink.streaming.connectors.kafka.FlinkKafkaProducerBase.getProp
ertiesFromBrokerList;

public class FlinkKafkaProducer {

    public static void main(String[] args) throws Exception {
        // set up the streaming execution environment
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        DataStream<WikipediaEditEvent> edits = env.addSource(new
WikipediaEditsSource());

        DataStream<String> stream =
edits.map(WikipediaEditEvent::toString);
        FlinkKafkaProducer011<String> myProducer = new
FlinkKafkaProducer011<String>(
                "wiki-edits", // target topic
                new SimpleStringSchema(),  // serialization schema
                getPropertiesFromBrokerList("localhost:9092"), // broker
list
```

```
                Optional.of(new CustomPartitioner<>()));  // round robin
partitioner
        stream.addSink(myProducer);

        DataStream<Tuple2<String, Integer>> result = edits
                // project the event user and the diff
                .map(new MapFunction<WikipediaEditEvent, Tuple2<String,
Integer>>() {
                    @Override
                    public Tuple2<String, Integer> map(WikipediaEditEvent
event) {
                        return new Tuple2<>(
                                event.getUser(), event.getByteDiff());
                    }
                })
                // group by user
                .keyBy(0)
                // aggregate changes per user
                .reduce(new ReduceFunction<Tuple2<String, Integer>>() {
                    @Override
                    public Tuple2<String, Integer> reduce(Tuple2<String,
Integer> e1, Tuple2<String, Integer> e2) {
                        return new Tuple2<>(e1.f0, e1.f1 + e2.f1);
                    }
                })
                // filter out negative byte changes
                .filter(new FilterFunction<Tuple2<String, Integer>>() {
                    @Override
                    public boolean filter(Tuple2<String, Integer> e)
throws Exception {
                        return e.f1 >= 0;
                    }
                });

        // execute program
        env.execute("Flink Streaming Java API Skeleton");
    }

    static class CustomPartitioner<T> extends FlinkKafkaPartitioner<T> {
        private int next = 0;

        @Override
        public int partition(T record, byte[] key, byte[] value, String
targetTopic, int[] partitions) {
            Preconditions.checkArgument(partitions != null &&
partitions.length > 0, "Partitions of the target topic is empty.");
            this.next = (this.next + 1) % partitions.length;
            return partitions[this.next];
        }
    }
}
```

FlinkKafkaConsumer.java

```
package wikiedits;

import java.util.Properties;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.flink.api.common.functions.FilterFunction;
import org.apache.flink.api.common.functions.MapFunction;
import org.apache.flink.api.common.functions.ReduceFunction;
import
org.apache.flink.api.common.serialization.AbstractDeserializationSchema;
import org.apache.flink.api.java.tuple.Tuple2;
import org.apache.flink.streaming.api.datastream.DataStream;
import
org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer011;
import org.apache.flink.streaming.connectors.wikiedits.WikipediaEditEvent;

public class FlinkKafkaConsumer {

    public static void main(String[] args) throws Exception {
        // set up the streaming execution environment
        final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

        Properties kafkaProps = new Properties();
        kafkaProps.setProperty("zookeeper.connect", "localhost:2181");
        kafkaProps.setProperty("bootstrap.servers", "localhost:9092");
        kafkaProps.setProperty("group.id", "test-consumer-group");
        // always read the Kafka topic from the start
        kafkaProps.setProperty("auto.offset.reset", "earliest");
        DataStream<WikipediaEditEvent> edits = env
                .addSource(new FlinkKafkaConsumer011<>("wiki-edits",
                        new CustomDeserializationSchema(), kafkaProps));

        DataStream<Tuple2<String, Integer>> result = edits
                // project the event user and the diff
                .map(new MapFunction<WikipediaEditEvent, Tuple2<String,
Integer>>() {
                    @Override
                    public Tuple2<String, Integer> map(WikipediaEditEvent
event) {
                        return new Tuple2<>(
                                event.getUser(), event.getByteDiff());
                    }
                })
                // group by user
                .keyBy(0)
                // tumbling event-time windows
                .timeWindow(Time.seconds(10))
                // aggregate changes per user
                .reduce(new ReduceFunction<Tuple2<String, Integer>>() {
                    @Override
                    public Tuple2<String, Integer> reduce(Tuple2<String,
Integer> e1, Tuple2<String, Integer> e2) {
                        return new Tuple2<>(e1.f0, e1.f1 + e2.f1);
                    }
                })
                // filter out negative byte changes
                .filter(new FilterFunction<Tuple2<String, Integer>>() {
                    @Override
                    public boolean filter(Tuple2<String, Integer> e)
throws Exception {
                        return e.f1 >= 0;
                    }
                });
        result.print();

        // execute program
        env.execute("Flink Streaming Java API Skeleton");
    }

    static class CustomDeserializationSchema extends
AbstractDeserializationSchema<WikipediaEditEvent> {
        @Override

        public WikipediaEditEvent deserialize(byte[] message) {
            // format:
            // "WikipediaEditEvent{timestamp=" + this.timestamp + ",
channel='" + this.channel + '\'' + ", title='" + this.title + '\'' + ",
diffUrl='" + this.diffUrl + '\'' + ", user='" + this.user + '\'' + ",
byteDiff=" + this.byteDiff + ", summary='" + this.summary + '\'' + ",
flags=" + this.flags + '}';
            // example:
            // WikipediaEditEvent{timestamp=1552870599316,
channel='#en.wikipedia', title='Talk:Shooting of Trayvon Martin',
diffUrl='https://en.wikipedi=888128575', user='Psalm84', byteDiff=-2,
summary='/* Proposed split */ minor word changes', flags=33}

            String s = new String(message);
            Pattern p = Pattern.compile("WikipediaEditEvent\\{timestamp=
(.*), channel='(.*)', title='(.*)', diffUrl='(.*)', user='(.*)', byteDiff=
(.*), summary='(.*)', flags=(.*)}");
            Matcher m = p.matcher(s);
            if (m.find()) {
                int flags = Integer.valueOf(m.group(8));
                boolean isMinor = (flags & 1) != 0;
                boolean isNew = (flags & 2) != 0;
                boolean isUnpatrolled = (flags & 4) != 0;
                boolean isBotEdit = (flags & 8) != 0;
                boolean isSpecial = (flags & 16) != 0;
                boolean isTalk = (flags & 32) != 0;
                return new WikipediaEditEvent(Long.valueOf(m.group(1)),
m.group(2), m.group(3), m.group(4), m.group(5),
Integer.valueOf(m.group(6)), m.group(7), isMinor, isNew, isUnpatrolled,
isBotEdit, isSpecial, isTalk);
            } else {
                throw new RuntimeException("error in parsing!");
            }
        }
    }
}
```