Scala and the Kafka Streams DSL: the Beauty and the Beast?

Carsten Seibert

Zurich Scala Meetup, 24.9.2019

About me

- In the IT business since 1993
- Different areas of business
  - Semi conductor
  - Stock exchange
  - Banking / Fintech
- Addicted to Scala since 2013

  … and a long history with Java before
- Passionate about
  - Scala – of course ;)
  - Even driven microservice architectures
  - Agile and evolutionary development
  - Pizza and a glass of good wine!

The Beauty and the Beast …

Scala is the Beauty …

… but Scala can also be a bit of a diva

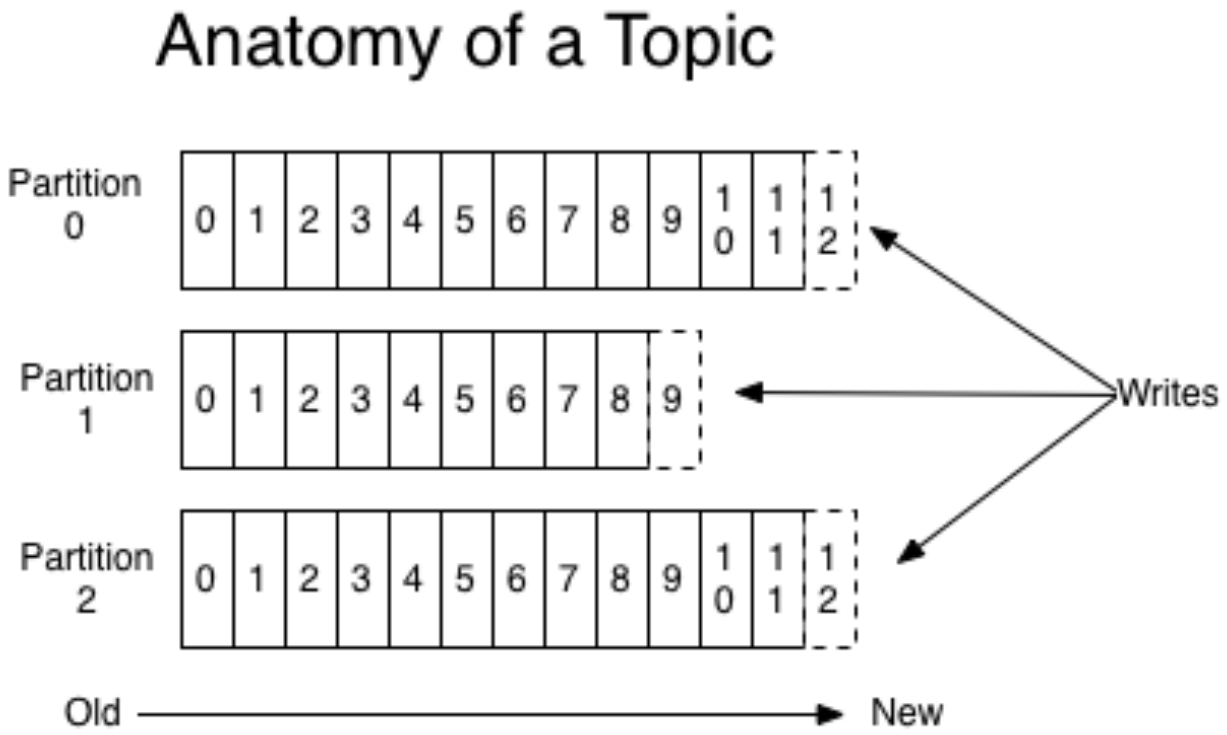
- Tuple22
- implicit resolution
- Macros
- Stack traces
- compile time

But all these flaws will be gone with Scala 3 and it will shine again ;)

Kafka – Very brief

- A distributed persistent message log
- With an eco system around it
  - KSQL
  - Schema Registry
  - Kafka Connect
  - …

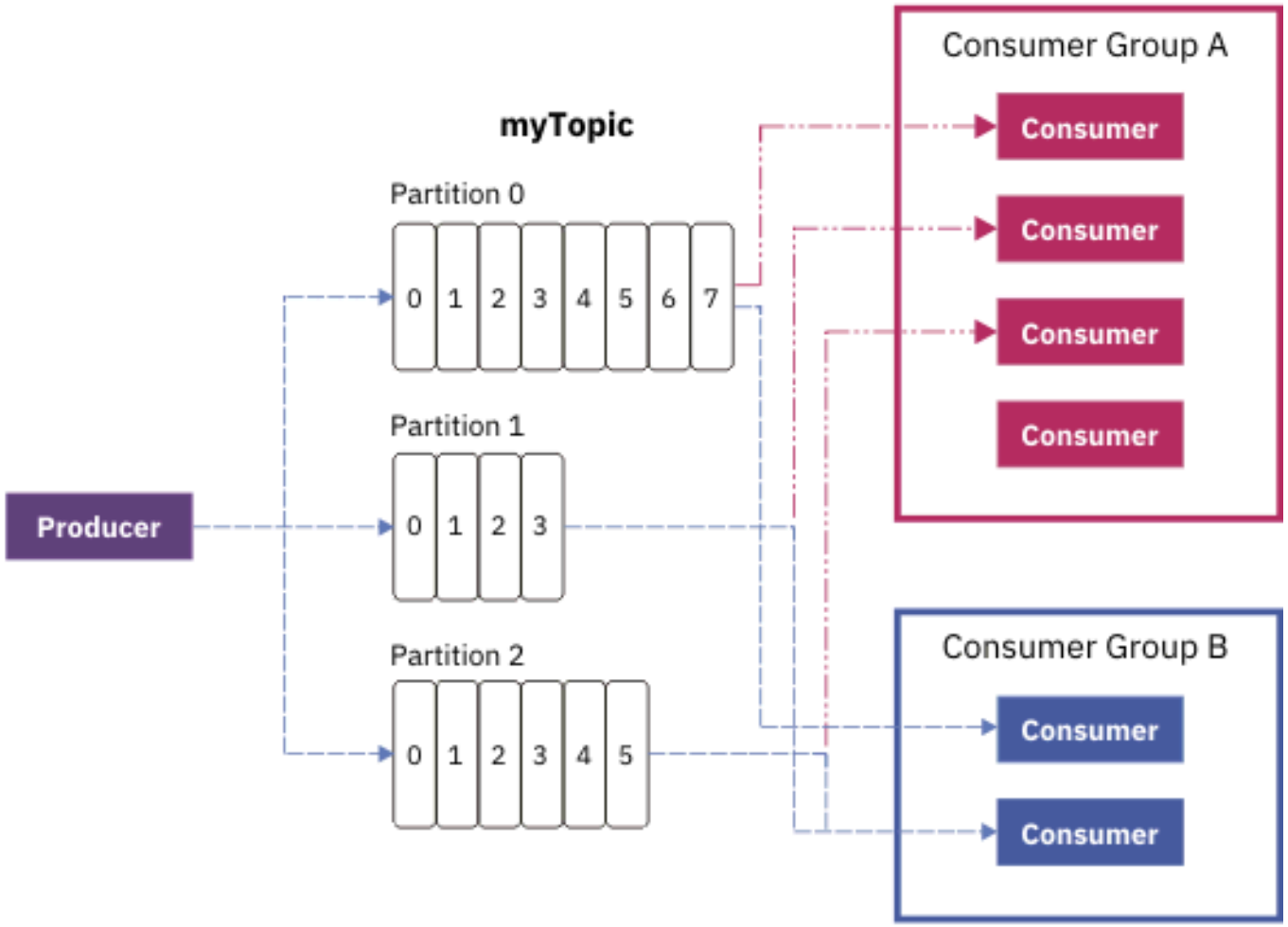
Topics and Partitions - Producing



https://kafka.apache.org/documentation.html

- Topics are created with a number of partitions
- A producer publishes to a topic
- Kafka determines the partition (by default)
- Order guaranteed per partition, NOT per topic!

Topics and Partitions - Consuming



https://cloud.ibm.com/docs/services/EventStreams?topic=eventstreams-apache_kafka

- Individual consumers belong to a consumer group (by consumerGroupId)
- Kafka assigns partitions to members of a consumer group
- Kafka rebalances this assignment if the members of a consumer group change

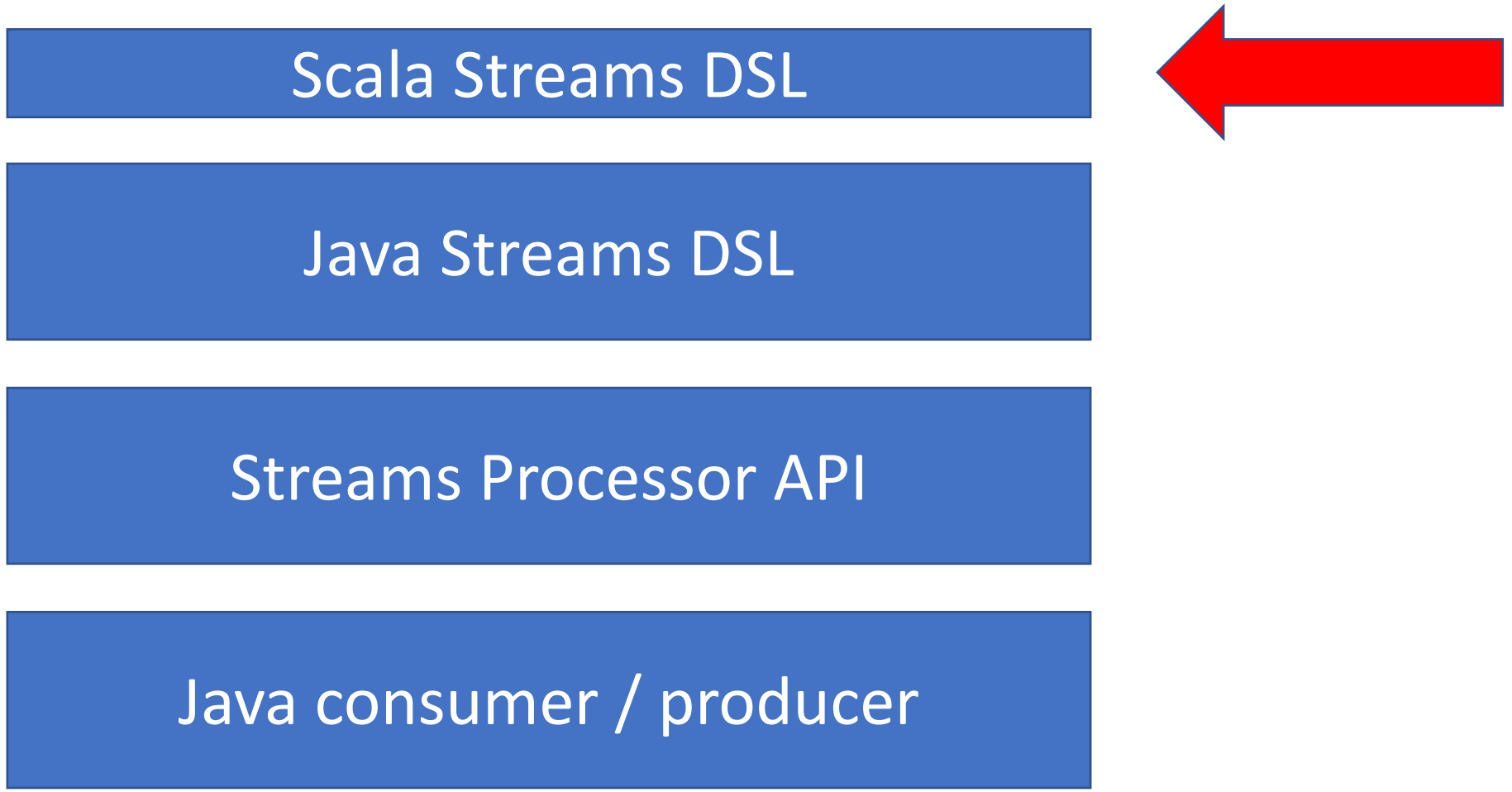
What is stream processing?

"Stream processing is the processing of *data in motion*, or in other words, computing on data directly as it is produced or received."

"The majority of data are born as continuous streams: sensor events, user activity on a website, financial trades, and so on – all these data are created as a series of events over time."

https://www.ververica.com/what-is-stream-processing

The Kafka client stack



Stream processing with Kafka

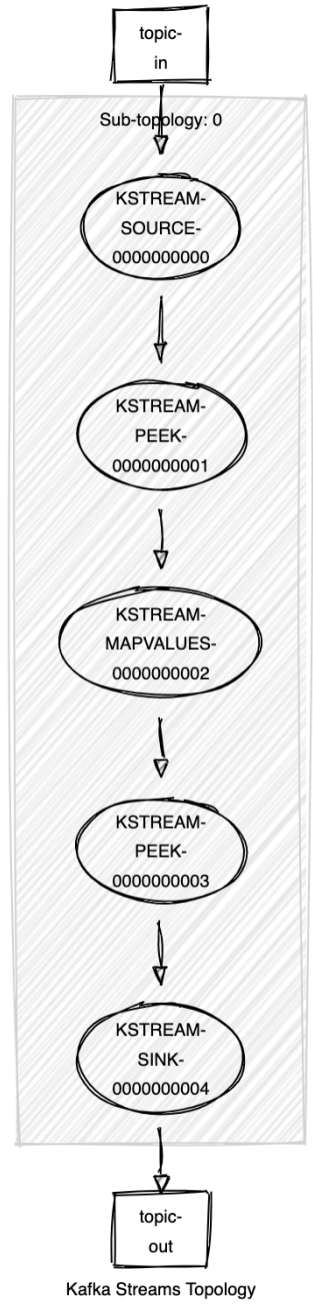
- in topic -> net of operations (topology) -> out topic
- "atomic" operation
- KStream vs KTable
  - KStream: events as published to the topic (changelog of a table)
  - KTable: only the latest value for a key is provided (snapshot of a stream)

  => Like some of major databases do!
- Stateless vs stateful operations

https://docs.confluent.io/current/streams/concepts.html

The first DSL – A very simple flow

```scala
package ch.seibertec.demo.scalakafka

import ...

object SimpleStream {

  val logger = LoggerFactory.getLogger(SimpleStream.getClass)

  val TopicIn = "topic-in"
  val TopicOut = "topic-out"

  import org.apache.kafka.streams.scala.ImplicitConversions._
  import org.apache.kafka.streams.scala.Serdes.String

  def buildWith(builder: StreamsBuilder): Unit =
    builder
      .stream[String, String](TopicIn)
      .peek( (k,v) => logger.info(s"From input stream $k -> $v"))
      .mapValues((k, v) => v + v)
      .peek( (k,v) => logger.info(s"To output stream $k -> $v"))
      .to(TopicOut)
}
```

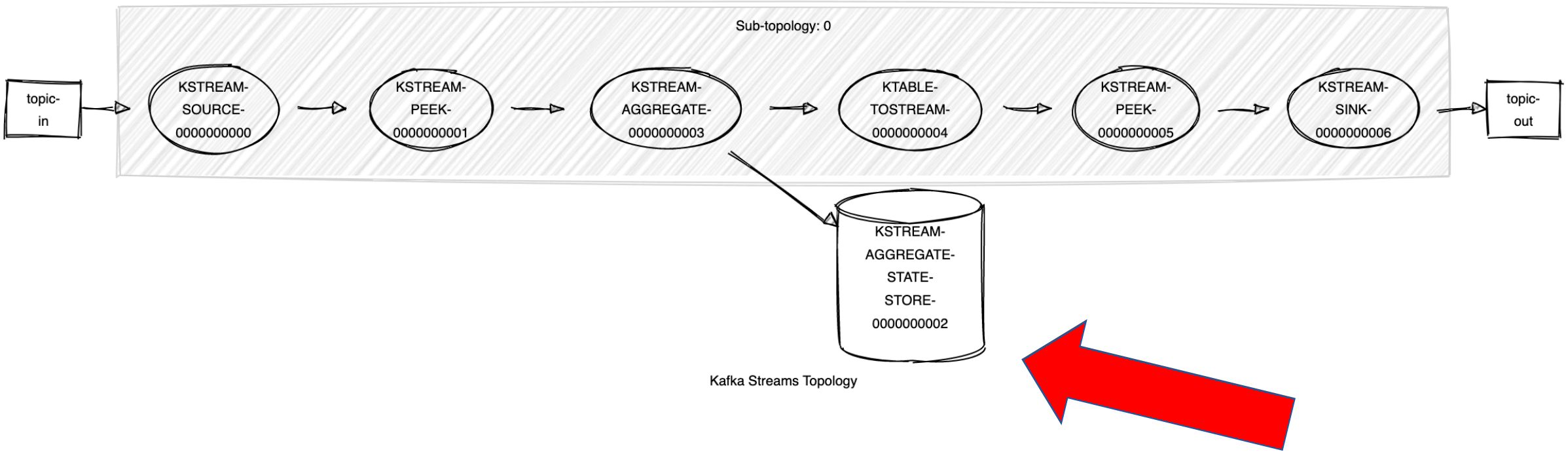


Kafka Streams Topology

DSL - Aggregation

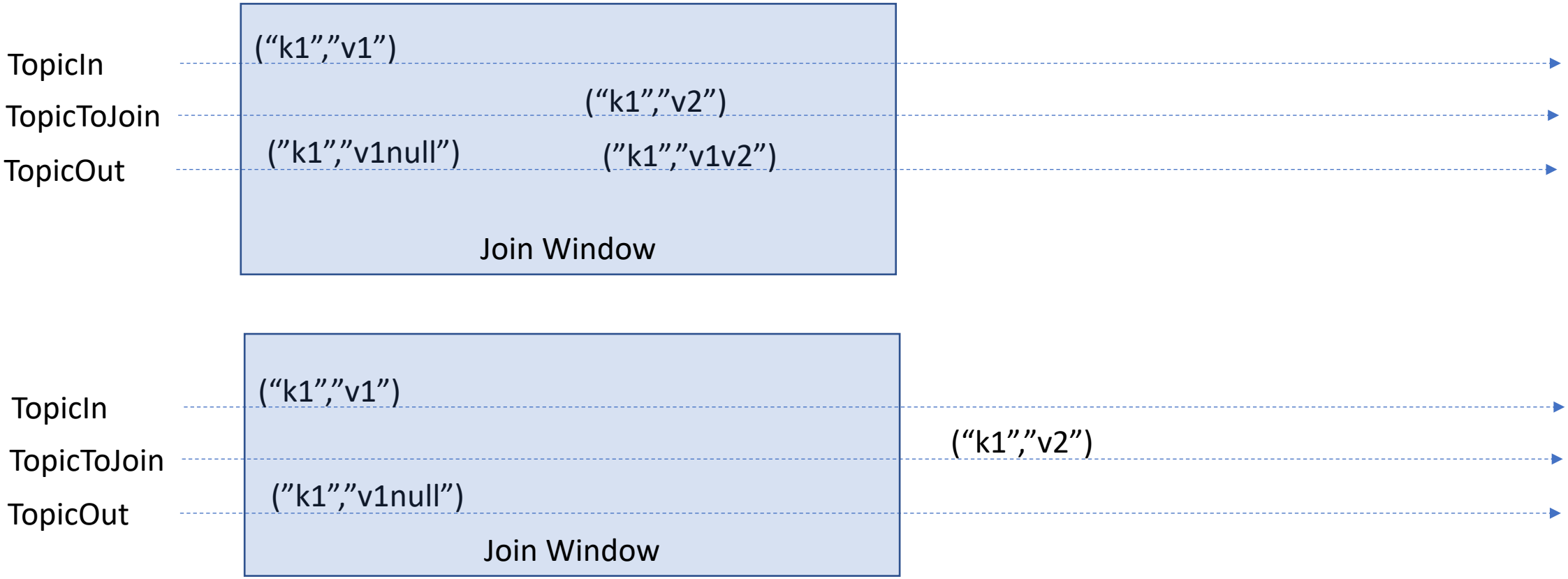
```
12
13      import org.apache.kafka.streams.scala.ImplicitConversions._
14      import org.apache.kafka.streams.scala.Serdes.String
15
16      def buildWith(builder: StreamsBuilder): Unit =
17        builder
18          .stream[String, String](TopicIn)
19          .peek((k, v) => logger.info(s"From input stream $k -> $v"))
20          .groupByKey
21          .aggregate( initializer = "") { (k, v, agg) =>
22            agg + v
23          }
24          .toStream
25          .peek((k, v) => logger.info(s"To output stream $k -> $v"))
26          .to(TopicOut)
```

DSL – Aggregation – cont’d



DSL – leftJoin (simplified)

Joiner: value 1 + value 2



DSL - Join

```
19
20      private val joinWindowDuration = Duration.ofMinutes( minutes = 2)
21
22      def buildWith(builder: StreamsBuilder): Unit = {
23
24        val streamToJoin = builder.stream[String,String](TopicToJoin)
25
26        builder
27          .stream[String, String](TopicIn)
28          .peek( (k,v) => logger.info(s"From input stream $k -> $v"))
29          .leftJoin(streamToJoin)( { (v1,v2) => s"$v1:$v2" }, JoinWindows.of(joinWindowDuration))
30          .peek( (k,v) => logger.info(s"To output stream $k -> $v"))
31          .to(TopicOut)
32      }
```

- Further reading: https://docs.confluent.io/current/streams/developer-guide/dsl-api.html#stateful-transformations

DSL – Join – cont’d

Kafka Streams Topology

Other DSL operations

- filter / filterNot
- branch
- foreach
- …

Time for some real code!!

(My) Conclusions

- Beauty
  - Fascinating programming model with seamless scalability
  - Nice functional abstraction
  - Excellent test support on several levels
- Beast
  - DSL operation sometime limited
    - Usage of “null”, branch function
  - The nice abstraction sometimes hides the inherent complexity

A beast with a good heart!

Questions?

Thanks!

mailto:seibert@seibertec.ch

Still there?

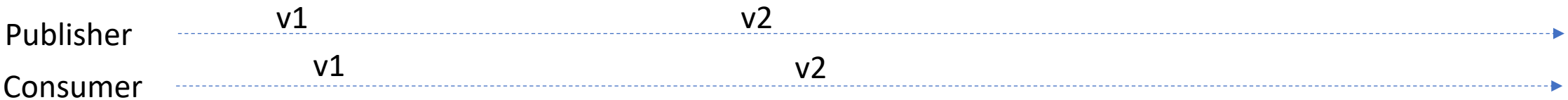
Serialization revisited: Avro

```
@namespace("ch.seibertec.demo.scalakafka")
protocol MySpecificarotocol {

  record MySpecificData {
    timestamp_ms dueTimeInMillis;
    string name;
    long aLongValue = 0;
    union { null, string}  anOptionalString = null;
  }
}
```

```
dueTimeInMillis: Instant, name: String, aLongValue: Long = 0L, anOptionalString: Option[String] = None

                                MySpecificData()
```

Serialization revisited: forward / backward compatibility

Newer producer, older consumer



Newer consumer, older producer => forward compatitibility & replay of older messages