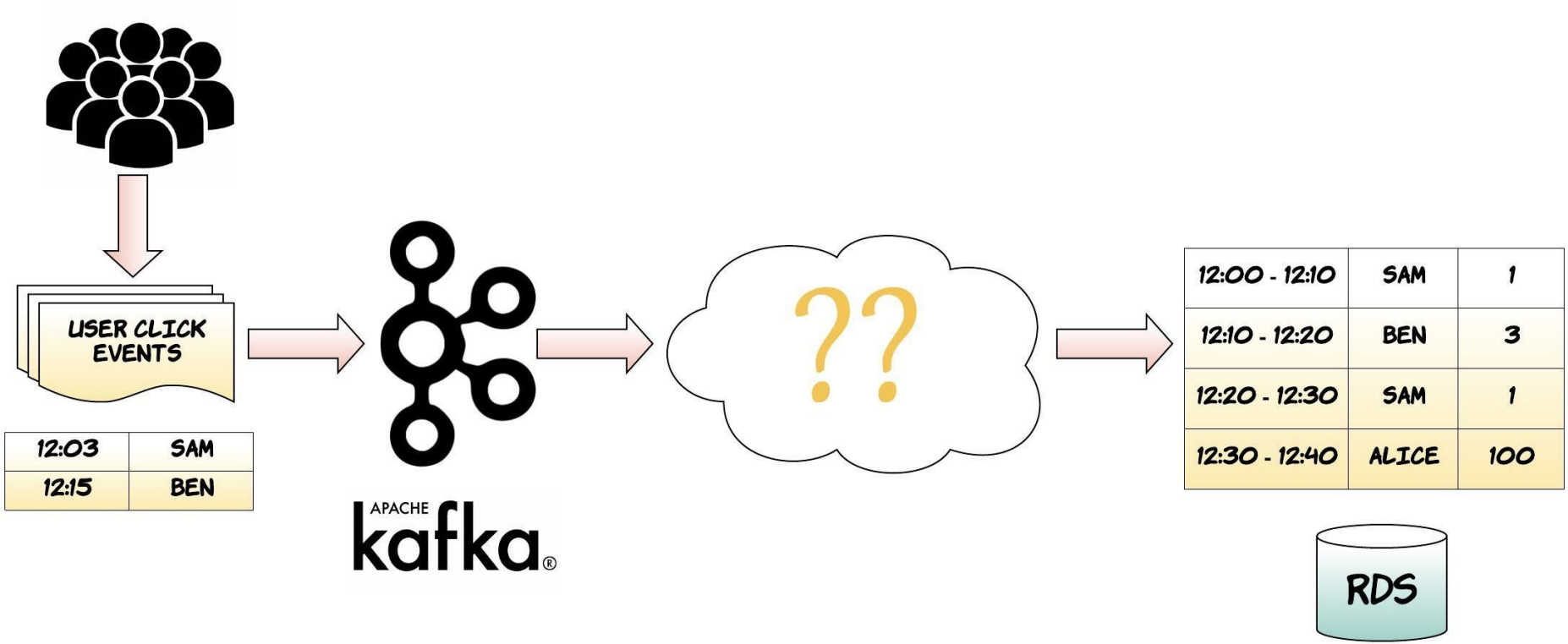
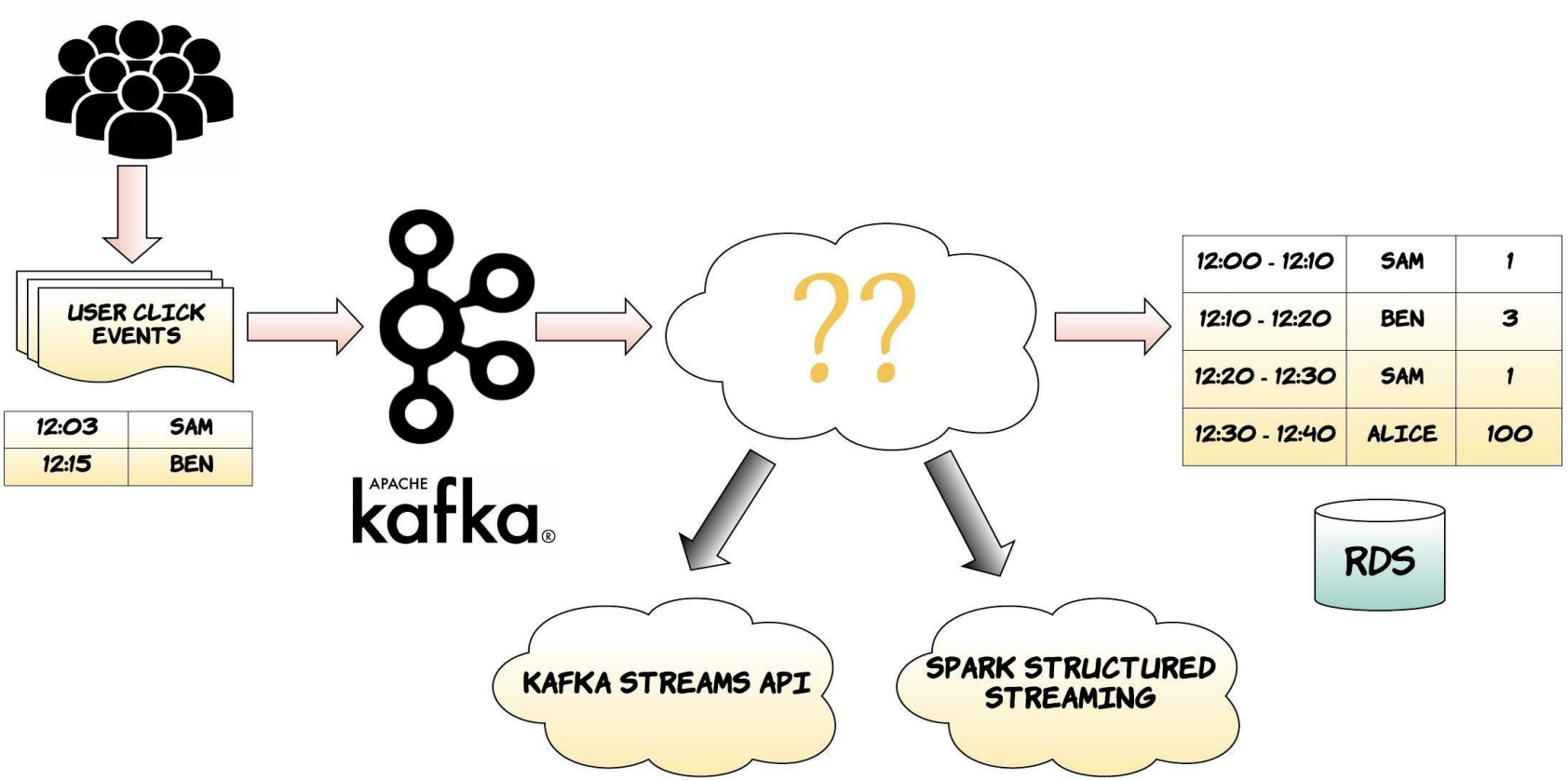
Navigating the stream

- Pranavi Chandramohan
Campaign Monitor
pranavic@campaignmonitor.com

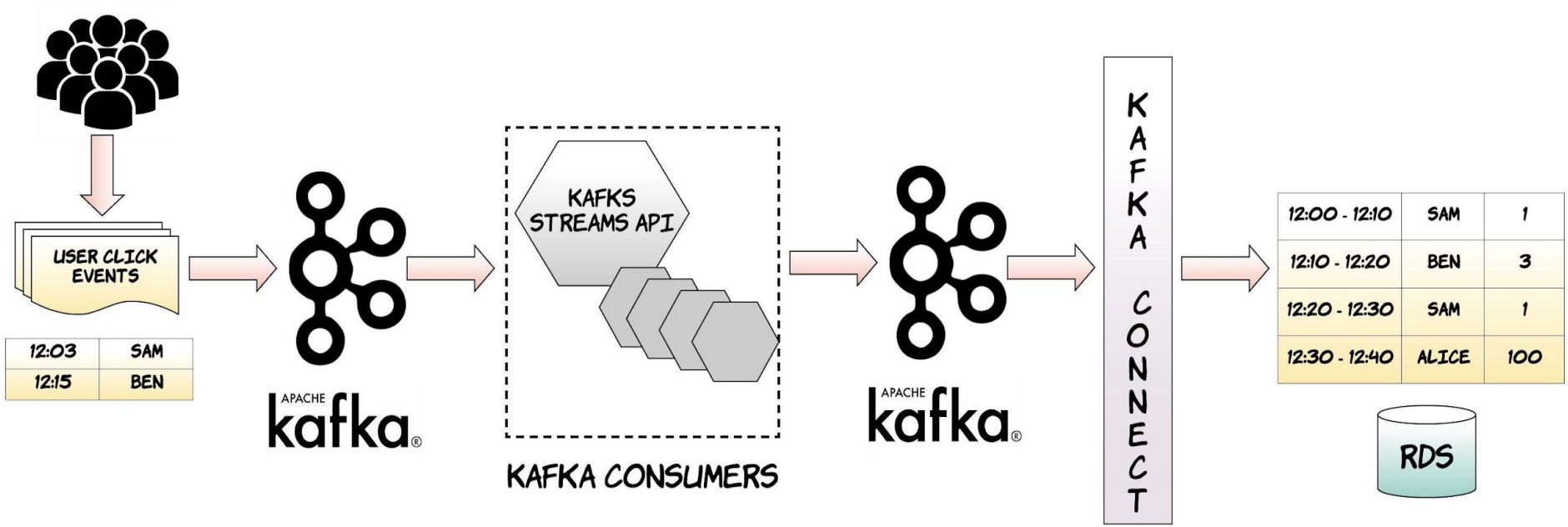
Real Time ETL



Real Time ETL



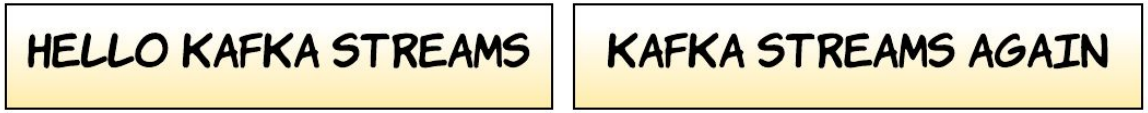
Kafka Streams



Word count in Kafka streams

```
Properties streamsConfiguration = new Properties();
streamsConfiguration.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "host:port");
…
…

KStream<String, String> textLines = builder.stream("topic1");

KTable<String, Long> wordCounts = textLines
      .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
      .groupBy((key, value) -> value)
      .count()
      .toStream()
      .to("output-topic", Produced.with(Serdes.String(), Serdes.Long()));

KafkaStreams streams = new KafkaStreams(textLines.build(),streamsConfiguration);
streams.start();
```

Word count in Kafka streams




```
KStream<String, String> textLines = builder.stream("topic1");
```

Word count in Kafka streams

```
KTable<String, Long> wordCounts = textLines
      .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
      .groupBy((key, value) -> value)
      .count()
      .toStream()
      .to("output-topic", Produced.with(Serdes.String(), Serdes.Long()));
```

HELLO	1
KAFKA	2
STREAMS	2
AGAIN	1

Word count in Kafka streams

```
KTable<String, Long> wordCounts = textLines
        .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
        .groupBy((key, value) -> value)
        .count()
        .toStream()
        .to("output-topic", Produced.with(Serdes.String(), Serdes.Long()));
```

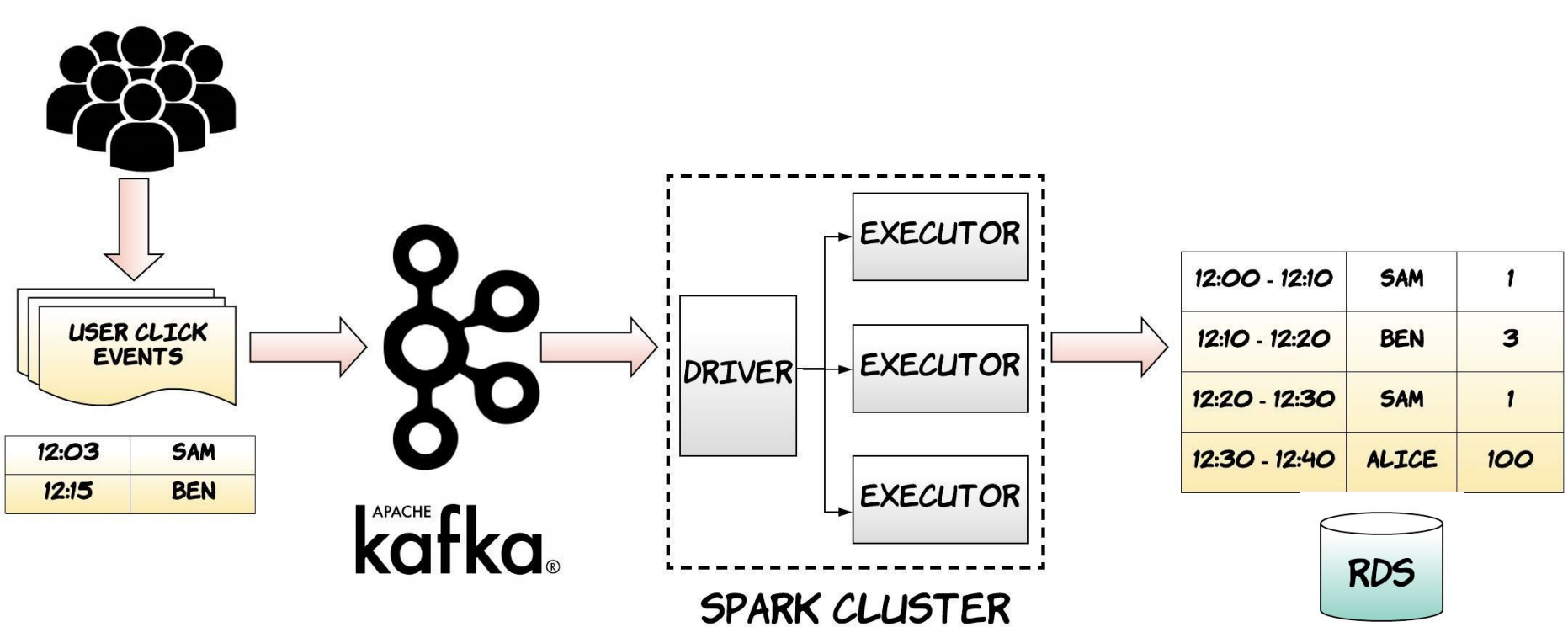
HELLO	1

KAFKA	2

STREAMS	2

AGAIN	1

Spark Struct Streaming

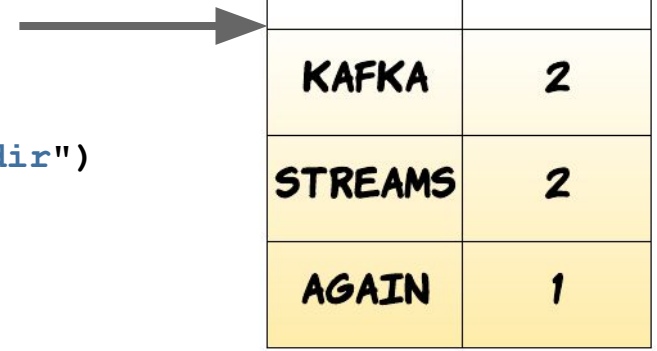


Word count in Struct Streaming

```
val textLines = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")
  …
  …

val words = textLines.flatMap(_.split("\\W+"))
val wordCounts = words.groupBy("value").count()
val query = wordCounts.writeStream
  .outputMode("update")
  .option("checkpointLocation", "path/to/HDFS/dir")
  .trigger("5 seconds")
  .format("kafka") / foreach(JDBCWriter)
  ...
  .start()

query.awaitTermination()
```

Word count in Struct Streaming

```
val textLines = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")
  …
  …

val words = textLines.flatMap(_.split("\\W+"))
val wordCounts = words.groupBy("value").count()
val query = wordCounts.writeStream
  .outputMode("update")
  .option("checkpointLocation", "path/to/HDFS/dir")
  .trigger("5 seconds")
  .format("kafka") / foreach(JDBCWriter)
  ...
  .start()

query.awaitTermination()
```



Word count in Struct Streaming

```
val textLines = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")
  …
  …

val words = textLines.flatMap(_.split("\\W+"))
val wordCounts = words.groupBy("value").count()
val query = wordCounts.writeStream
  .outputMode("update")
  .option("checkpointLocation", "path/to/HDFS/dir")
  .trigger("5 seconds")
  .format("kafka") / foreach(JDBCWriter)
  ...
  .start()

query.awaitTermination()
```

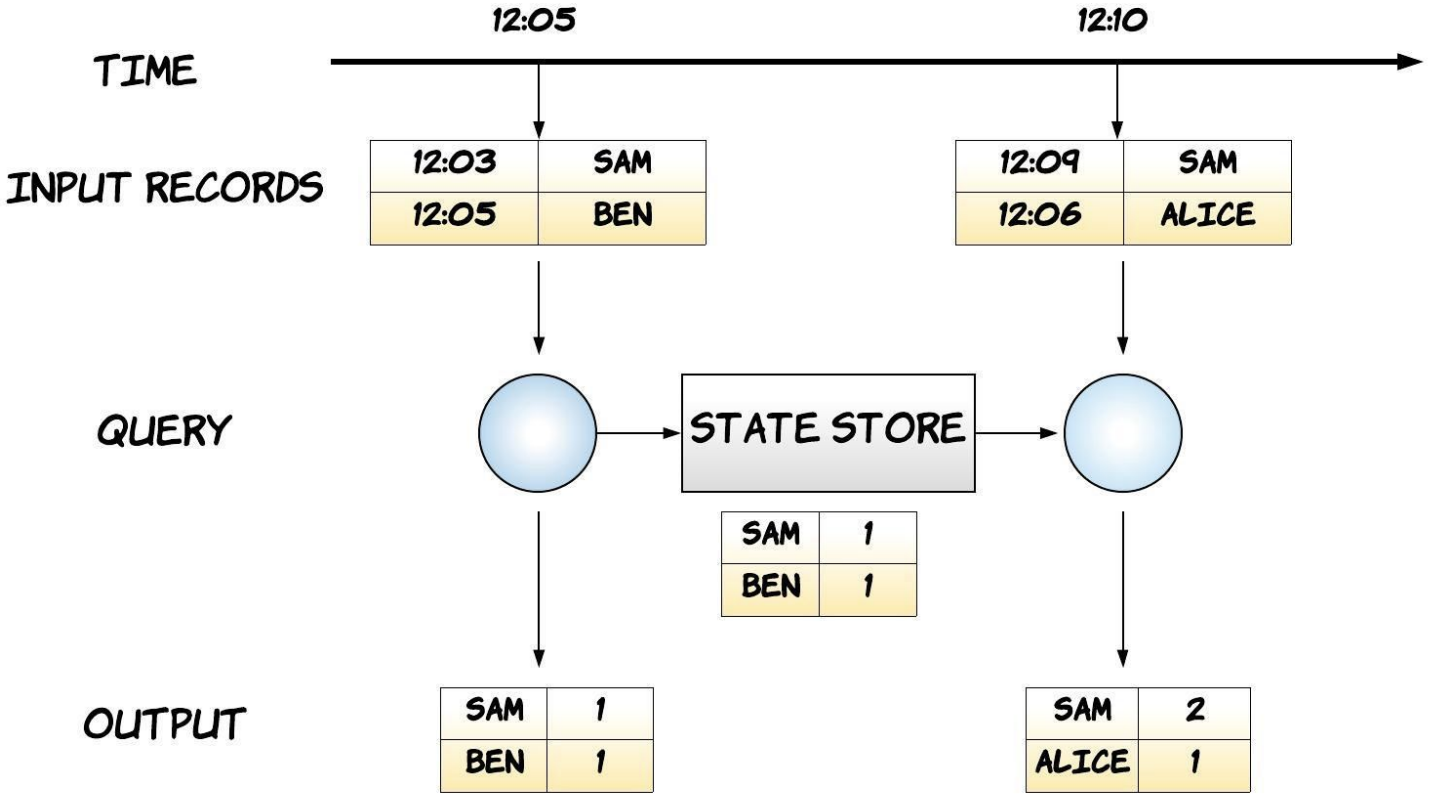
Word count in Struct Streaming

```
val textLines = spark
  .readStream
  .format("kafka")
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")
  …
  …

val words = textLines.flatMap(_.split("\\W+"))
val wordCounts = words.groupBy("value").count()
val query = wordCounts.writeStream
  .outputMode("update")
  .option("checkpointLocation", "path/to/HDFS/dir")
  .trigger("5 seconds")
  .format("kafka") / foreach(JDBCWriter)
  ...
  .start()

query.awaitTermination()
```

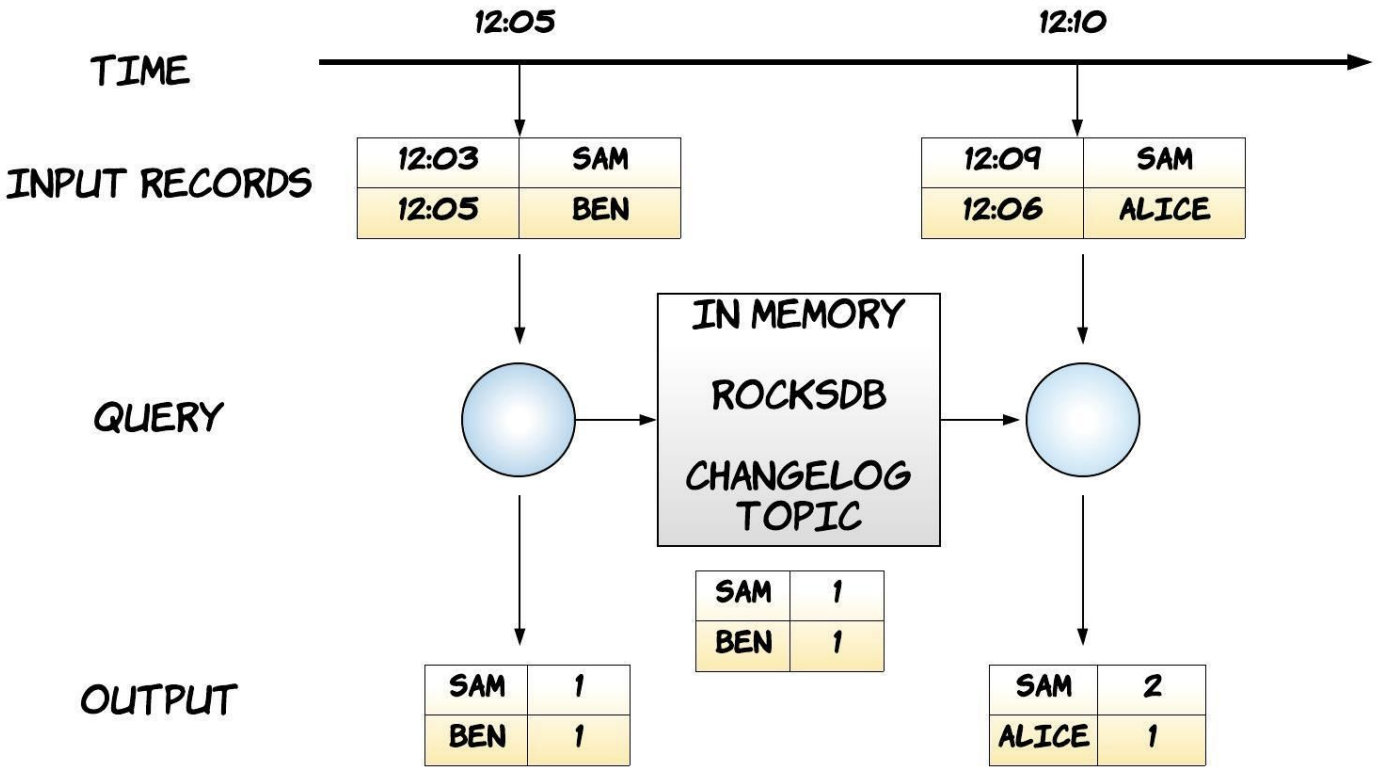
Challenges

- Seamless stateful processing like aggregations.
- Event time vs Processing time
  - Windowing and out of order data
- Distributed and fault tolerant processing.
- Exactly once processing.

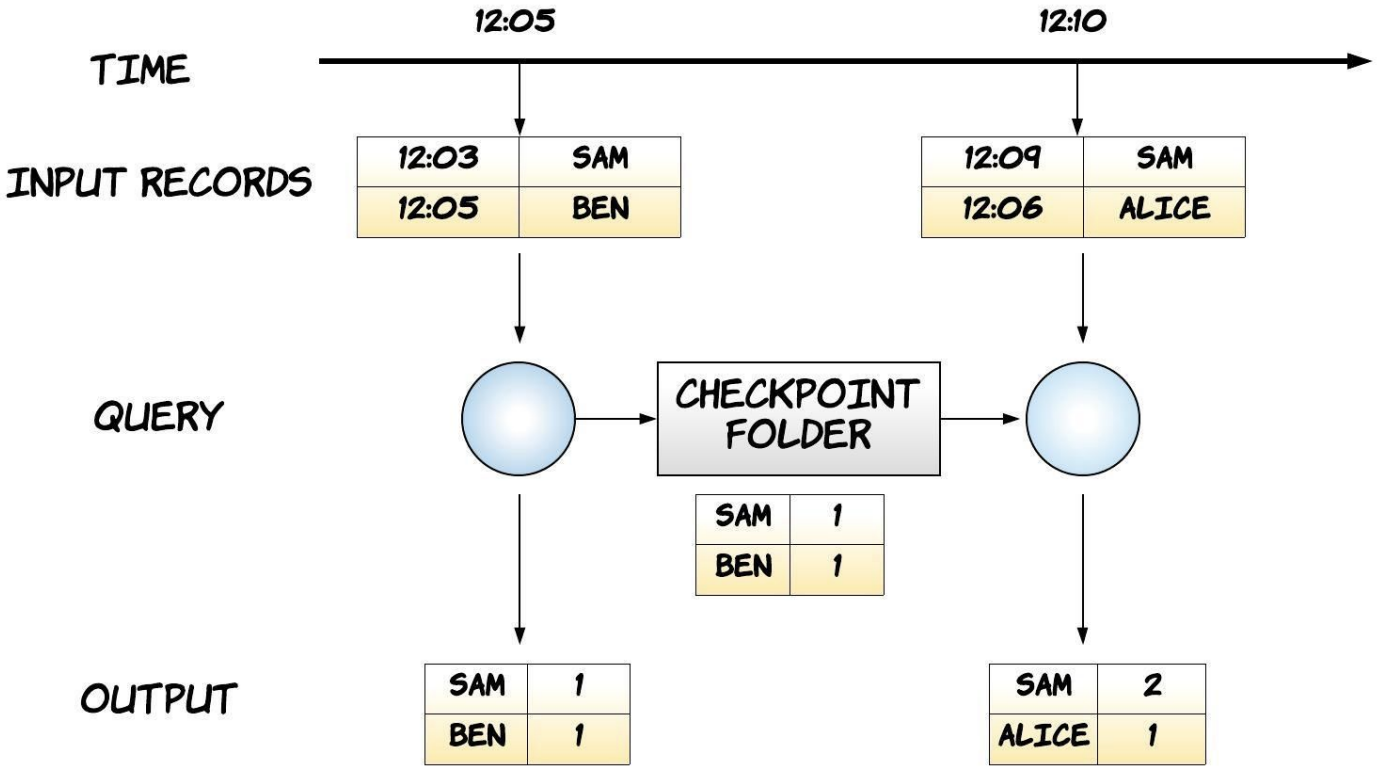
Stateful transformation



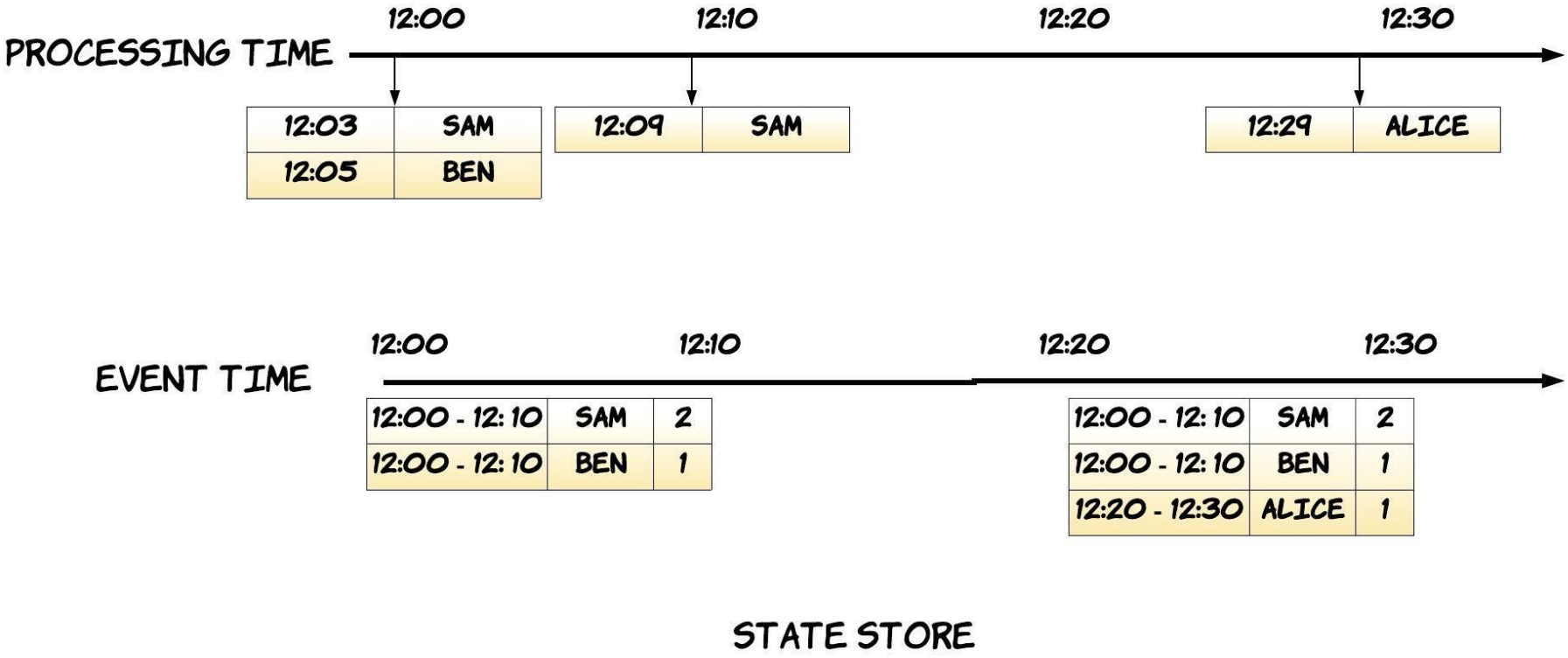
Kstreams state store



Struct streaming state store



Window operations



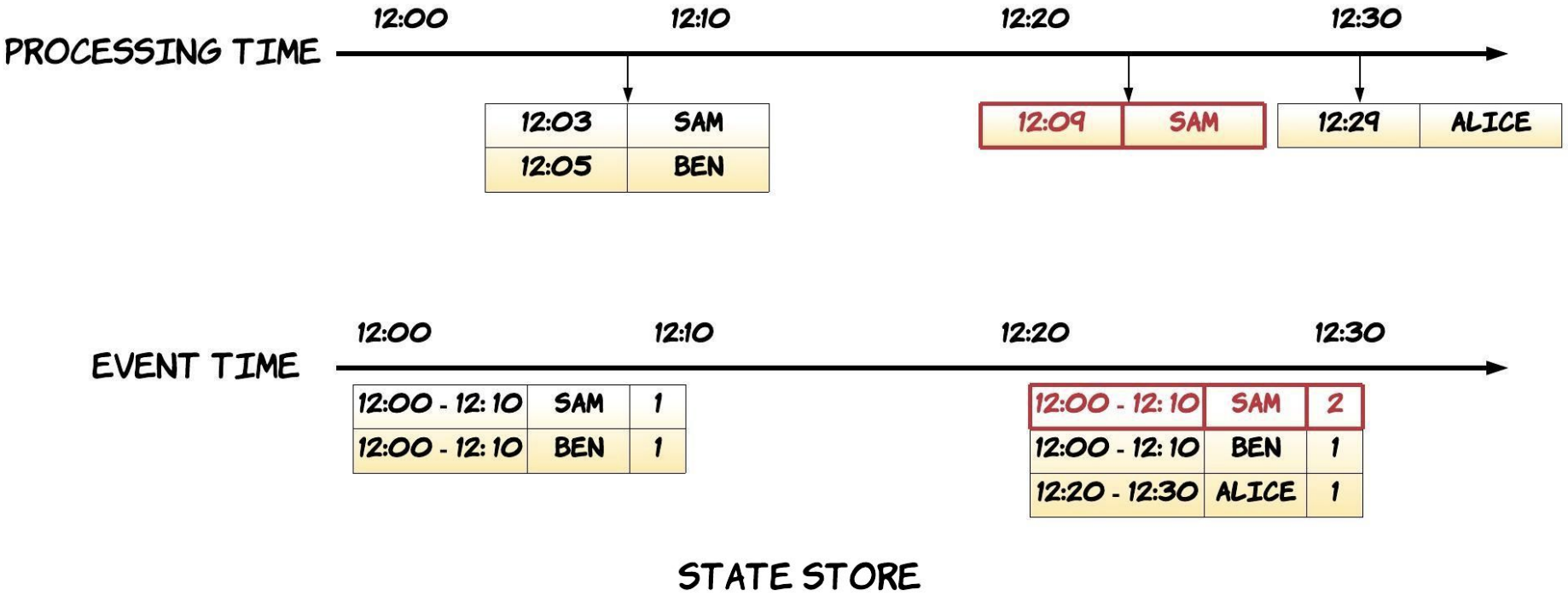
KStreams window

```
KTable<Windowed<String>, Long> windowedClickCounts =
      clicks
      .groupByKey(Serialized.with(Serdes.String(),genericAvroSerde))
      .windowedBy(TimeWindows.of(TimeUnit.MINUTES.toMillis(10)))
      .count();
```

Struct Streams window

```
clicks
    .groupBy(
      "userId",
      window("clickTime", "10 minutes")
    .count()
```

Late data


PROCESSING TIME
12:00
12:10
12:20
12:30
12:03 SAM
12:05 BEN
12:09 SAM
12:29 ALICE
EVENT TIME
12:00
12:10
12:20
12:30
12:00 - 12: 10 SAM 1
12:00 - 12: 10 BEN 1
12:00 - 12: 10 SAM 2
12:00 - 12: 10 BEN 1
12:20 - 12:30 ALICE 1
STATE STORE

State retention

- Kstreams
  - Log compaction is enabled on the changelog topics.

- Struct Streaming
  - Watermarks

```
pageViews
    .withWatermark("pageViewTime", "30 minutes")
    .groupBy(
      "userId",
      window("pageViewTime", "10 minutes")
    .count()
```

Exactly once processing

- Kstreams
  - “processing.guarantee” to “exactly_once”

- Spark Struct Streaming
  - Replayable Input source.
  - Fault tolerant state store.
  - Idempotent output sinks.

Extras

	Kafka streams	Struct Streaming
Deployment	Any java consumer	Spark cluster
Latency	Event at a time processing	Trigger cycle with micro batches
Query language	Kafka streams DSL	Sql, Dataframe and Dataset API
Language support	Java	Java, Python, R, Scala
Input sources	Kafka	File, Socket, Kafka ...
Output sinks	Kafka	File, kafka, console, memory ...

Gotchas

- KStreams
  - Additional topics being created.
  - Re partition topics - Kafka Streams inserts a repartitioning step if a key-based operation like aggregation or join is preceded by a key changing operation like selectKey(), map, or flatMap().
  - Memory management.
  - Sizing and scaling.

- Struct Streaming
  - Backpressure - kafka config ***"maxOffsetsPerTrigger"***
  - Changes in a Streaming Query will need reprocessing from beginning.
    - Checkpoint <-> query

Summary!

Question time!

References

- https://databricks.com/blog/2017/01/19/real-time-streaming-etl-structured-streaming-apache-spark-2-1.html
- https://databricks.com/blog/2017/04/26/processing-data-in-apache-kafka-with-structured-streaming-in-apache-spark-2-2.html
- https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
- https://docs.confluent.io/current/streams/introduction.html