**Streaming Basics**

**SPARK:**

* It is open source **cluster-computing framework** designed for speed & ease of use. It's well known for in memory performance.
* It's highly accessible for offering view API for Scala, Java, Python, R and SQL. It has integrated libraries for ML, SQL, streaming, etc.
* **100x faster** **in memory** than Map reduce.
* **10x faster in disk** than Map reduce.
* Spark does not have its own distributed file system but can **use HDFS**.

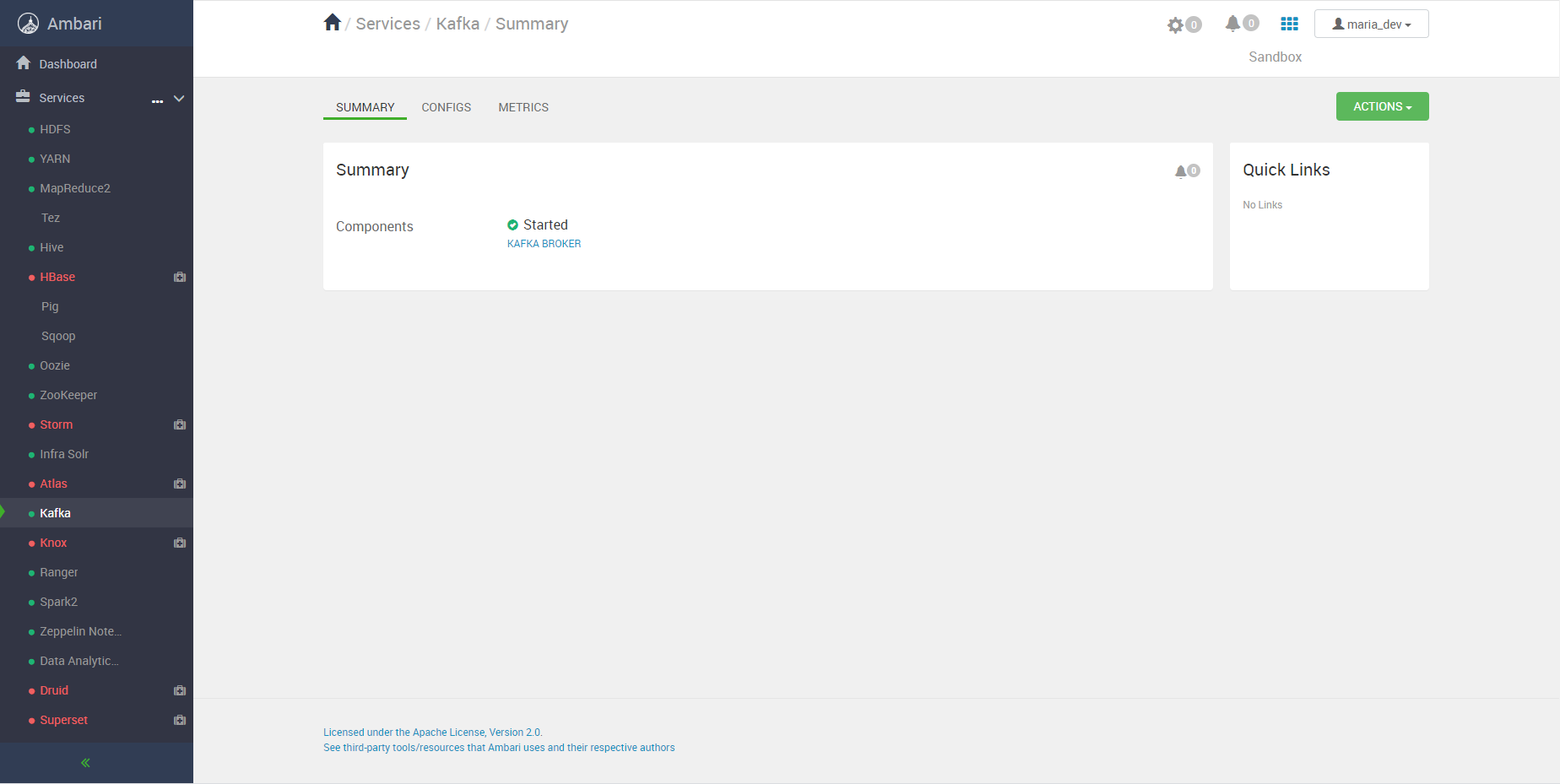
**Spark Streaming:**

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data.

**Kafka:**

It’s a **distributed stream platform**. A streaming platform has three key capabilities:

1. **Publish and subscribe to streams of records**, like a message queue or enterprise messaging system.
2. **Store streams of records in a fault-tolerant durable way**.
3. **Process streams of records** as they occur.

****

In the above image, Kafka in sandbox is running.

**Kafka has four core APIs:**

1. **Producer API**

The Producer API allows an application to **publish a stream of records to one or more Kafka topics**.

1. **Consumer API**

The Consumer API **allows an application to subscribe to one or more topics and process the stream of records produced to them**.

1. **Streams API**

The Streams API **allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.**

1. **Connector API**

The Connector API **allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems**.

**Topic in Kafka:**

A topic is a **category name to which records are published**. Topics in **Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it**.



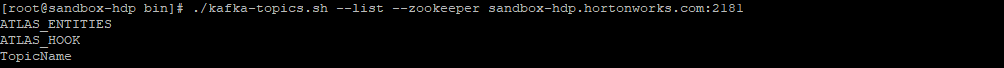
In the above image we have following details:

Partitions: 1

Replication factor: 1

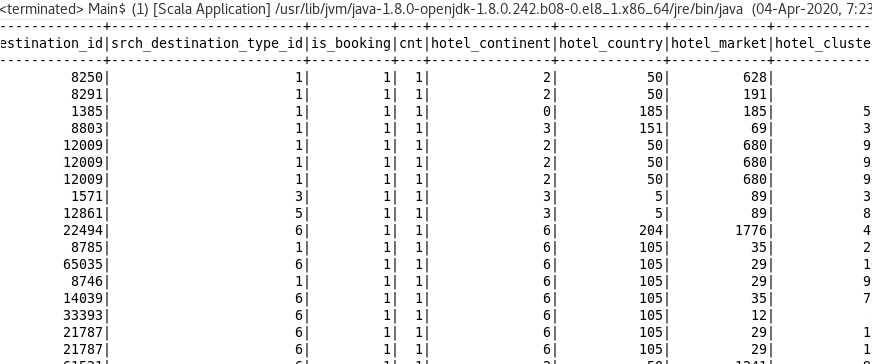
Topic name: TopicName

**Topic Listing in Kafka:**

****

In the above image, we can list the Topic names currently exists.

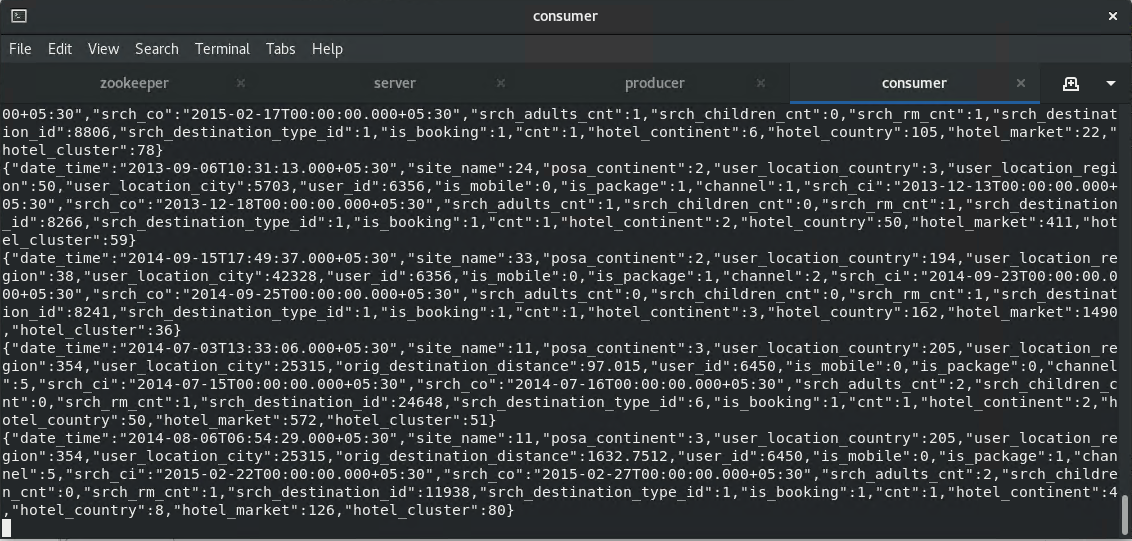
**Listing CSV Data in Eclipse:**



In the above image, I have printed data on the console.

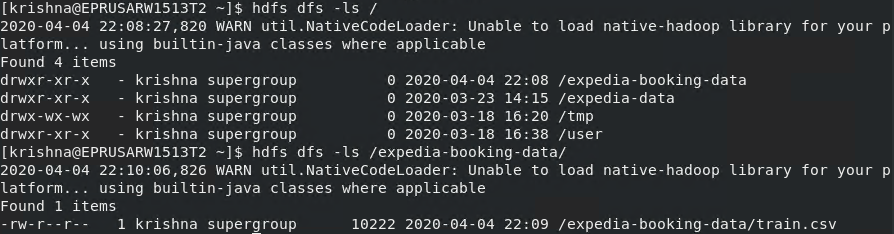
**Consumer in Kafka:**

Each **record published to a topic is delivered to one consumer instance within each subscribing consumer group**. Consumer instances can be in separate processes or on separate machines.



In the above image, you can see consumer is getting o/p which send by the eclipse scala-spark program.

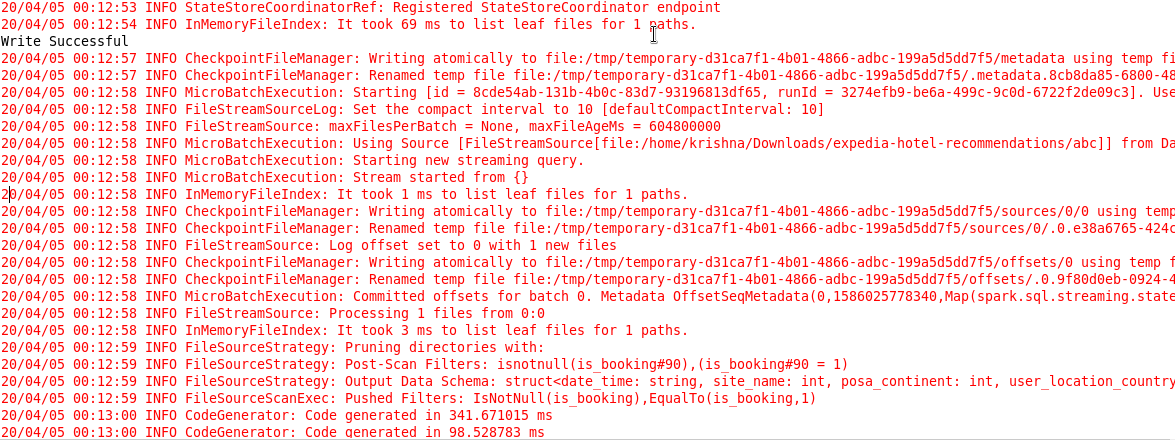
**Finally Data is save back to the HDFS:**



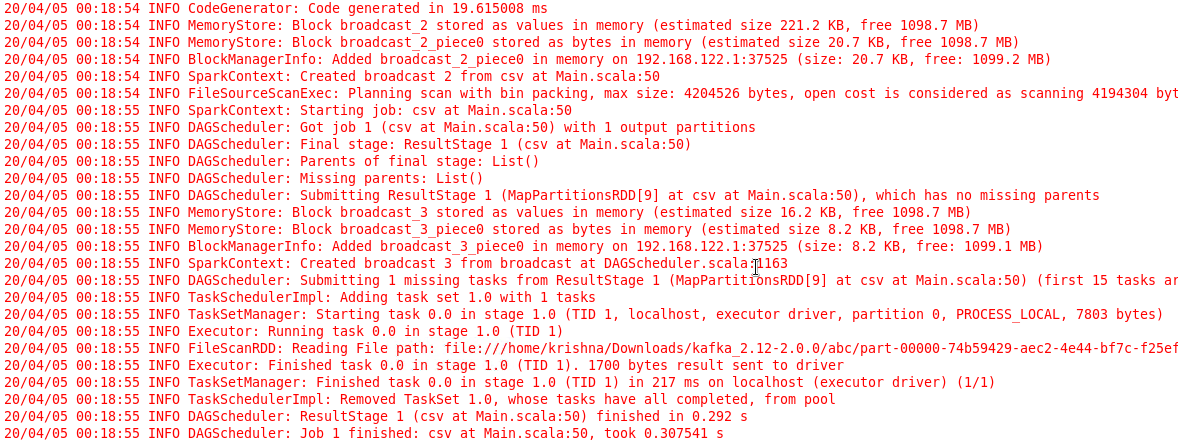
In the above image, we can see that data is saved in HDFS with the file name train.csv.

**Logs to Show the performance:**

1. **Writing file data to the kafka topic**



1. **Writing Data to the hdfs.**



**Comparing with Batch Processing:**

1. Hadoop is an open source framework which uses a Map Reduce algorithm whereas Spark is lightning fast cluster computing technology, which extends the Map Reduce model to efficiently use with more type of computations.
2. Hadoop’s Map Reduce model reads and writes from a disk, thus slow down the processing speed whereas Spark reduces the number of read/write cycles to disk and store intermediate data in-memory, hence faster-processing speed.
3. Hadoop requires developers to hand code each and every operation whereas Spark is easy to program with RDD – Resilient Distributed Dataset.
4. Hadoop Map Reduce model provides a batch engine, hence dependent on different engines for other requirements whereas Spark performs batch, interactive, Machine Learning and Streaming all in the same cluster.
5. Hadoop is designed to handle batch processing efficiently whereas Spark is designed to handle real-time data efficiently.
6. Hadoop is a high latency computing framework, which does not have an interactive mode whereas Spark is a low latency computing and can process data interactively.
7. With Hadoop Map Reduce, a developer can only process data in batch mode only whereas Spark can process real-time data through Spark Streaming.
8. Hadoop is designed to handle faults and failures, it is naturally resilient toward faults, hence a highly fault-tolerant system whereas, with Spark, RDD allows recovery of partitions on failed nodes.
9. Hadoop needs an external job scheduler for example – Oozie to schedule complex flows whereas Spark has in-memory computation, so it has its own flow scheduler.
10. Hadoop is a cheaper option available while comparing it in terms of cost whereas Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost.