## [EXPEDIA GROUP TECHNOLOGY — DATA](https://medium.com/expedia-group-tech/data/home)

# **Apache Spark Structured Streaming — First Streaming Example (1 of 6)**

## Using a scalable and fault-tolerant stream processing engine

[](https://medium.com/@bhadani.neeraj.08?source=post_page-----e8f3219748ef--------------------------------)

[IMG_257](https://medium.com/expedia-group-tech?source=post_page-----e8f3219748ef--------------------------------)

[Neeraj Bhadani](https://medium.com/@bhadani.neeraj.08?source=post_page-----e8f3219748ef--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https://medium.com/_/subscribe/user/f9ed76fbd13a&operation=register&redirect=https://medium.com/expedia-group-tech/apache-spark-structured-streaming-first-streaming-example-1-of-6-e8f3219748ef&user=Neeraj+Bhadani&userId=f9ed76fbd13a&source=post_page-f9ed76fbd13a----e8f3219748ef---------------------post_header-----------)

Published in

[Expedia Group Technology](https://medium.com/expedia-group-tech?source=post_page-----e8f3219748ef--------------------------------)

·

5 min read

·

Jan 28, 2021

225

3



Photo by Mike Lewis HeadSmart Media on Unsplash

In this blog series, we discuss Apache Spark™️ Structured Streaming. We cover components of Apache Spark Structured Streaming and play with examples to understand them.

You may also be interested in some of my other posts on Apache Spark.

* [Deep Dive into Apache Spark DateTime Functions](https://medium.com/expedia-group-tech/deep-dive-into-apache-spark-datetime-functions-b66de737950a)
* [Working with JSON in Apache Spark](https://medium.com/expedia-group-tech/working-with-json-in-apache-spark-1ecf553c2a8c)
* [Deep Dive into Apache Spark Window Functions](https://medium.com/expedia-group-tech/deep-dive-into-apache-spark-window-functions-7b4e39ad3c86)
* [Deep Dive into Apache Spark Array Functions](https://medium.com/expedia-group-tech/deep-dive-into-apache-spark-array-functions-720b8fbfa729)
* [Start Your Journey with Apache Spark](https://medium.com/expedia-group-tech/start-your-journey-with-apache-spark-part-1-3575b20ee088)

IMG_259

*Note: We use a Scala API in this blog series. For easy reference, Scala files are available on****[GitHub](https://github.com/NeerajBhadani/spark-streaming/tree/master/src/main/scala" \t "https://medium.com/expedia-group-tech/_blank)****.*

Let’s get started with our journey on Apache Spark Structured Streaming.

# **Overview**

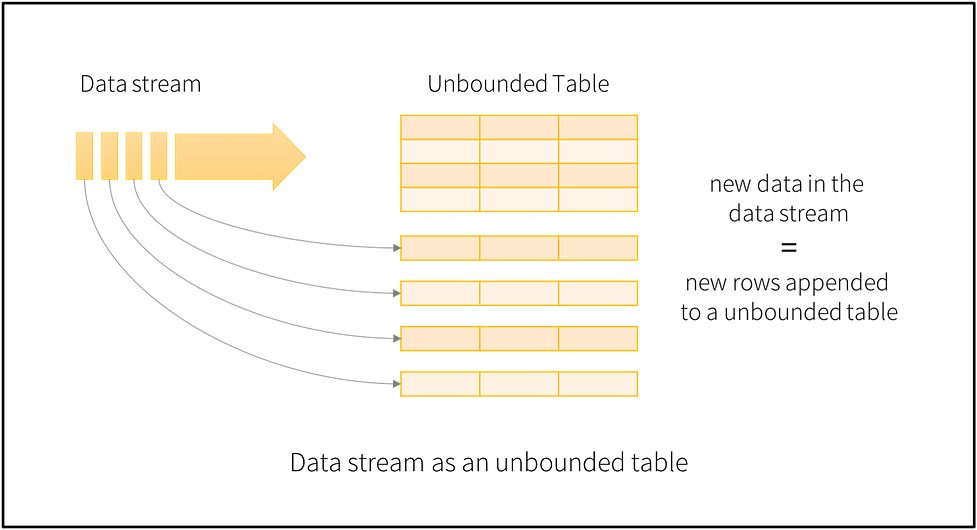
Working with streaming data is a little different from working with batch data. With streaming data, we will never have complete data for analysis, as data is continuously coming in. Apache Spark provides a streaming API to analyze streaming data in pretty much the same way we work with batch data. Apache Spark Structured Streaming is built on top of the Spark-SQL API to leverage its optimization. Spark Streaming is a processing engine to process data in real-time from sources and output data to external storage systems.



[Reference](https://spark.apache.org/docs/latest/streaming-programming-guide.html" \l "overview" \t "https://medium.com/expedia-group-tech/_blank)

Spark Streaming has 3 major components: input sources, streaming engine, and sink. Input sources generate data like Kafka, Flume, HDFS/S3, etc. Spark Streaming engine processes incoming data from various input sources. Sinks store processed data from Spark Streaming engine like HDFS, relational databases, or NoSQL datastores.

Let’s conceptualise Spark Streaming data as an unbounded table where new data will always be appended at the end of the table.



[Reference](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html" \l "basic-concepts" \t "https://medium.com/expedia-group-tech/_blank)

Spark will process data in micro-batches which can be defined by triggers. For example, let's say we define a trigger as 1 second, this means Spark will create micro-batches every second and process them accordingly. We will discuss triggers in a separate blog.

# **Output modes**

After processing the streaming data, Spark needs to store it somewhere on persistent storage. Spark uses various output modes to store the streaming data.

* ****Append Mode****: In this mode, Spark will output only newly processed rows since the last trigger.
* ****Update Mode****: In this mode, Spark will output only updated rows since the last trigger. If we are not using aggregation on streaming data (meaning previous records can’t be updated) then it will behave similarly to append mode.
* ****Complete Mode****: In this mode, Spark will output all the rows it has processed so far.

# **Our first streaming example using**rate**source**

Now let’s get our hands dirty with our first Spark Streaming example using ratesource and consolesink. Rate source will auto-generate data which we will then print onto a console.

## **Import libraries**

Let’s first import the required libraries.

## **Create Spark session**

Now let’s create the sparkSession and set the logging level to Error to avoid the Warning and INFO logs.

## **Create streaming DataFrame**

Let’s create our first Spark Streaming DataFrame using rate source. Here we have specified the format as rate and specified rowsPerSecond = 1 to generate 1 row for each micro-batch and load the data into initDF streaming DataFrame. Also, we check if the initDF is a streaming DataFrame or not.

Output should read as follows:

Streaming DataFrame : true

# **Basic transformation**

Perform a basic transformation on initDF to generate another column result by just adding 1 to column value :

We created a derived column result from an existing column value in a very similar way to creating one in a batch DataFrame. You can find various operations on batch DataFrame in [this](https://medium.com/expedia-group-tech/start-your-journey-with-apache-spark-part-2-682891efda4b) blog.

## **Output to console**

Let’s try to print the contents of a streaming DataFrame to console. Here, we use append output mode to output only newly generated data and format as console to print the result on the console.

This output should resemble the following:

-------------------------------------------  
Batch: 1  
-------------------------------------------  
+-----------------------+-----+------+  
|timestamp |value|result|  
+-----------------------+-----+------+  
|2020-12-28 11:40:37.867|0 |1 |  
+-----------------------+-----+------+-------------------------------------------  
Batch: 2  
-------------------------------------------  
+-----------------------+-----+------+  
|timestamp |value|result|  
+-----------------------+-----+------+  
|2020-12-28 11:40:38.867|1 |2 |  
+-----------------------+-----+------+

Here we printed the result of 2 micro-batches which contain columns timestamp, value and the generated result column. We have only one record per second (check timestamps above) as specified in initDF and value was generated automatically. Also, in each batch we only get newly generated data since we have used the append output mode.

We have successfully written our first spark streaming application!

IMG_262

The next post in this series covers [input sources](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-input-sources-2-of-6-6a72f798838c) of different types.

I hope you enjoyed this overview of Spark Streaming! For easy reference, you can find the complete code on [GitHub](https://github.com/NeerajBhadani/spark-streaming/blob/master/src/main/scala/streamRateSource.scala" \t "https://medium.com/expedia-group-tech/_blank).

Here are other blogs on Apache Spark Structured Streaming series.

* [Apache Spark Structured Streaming — Input Sources](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-input-sources-2-of-6-6a72f798838c)
* [Apache Spark Structured Streaming — Output Sinks](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-output-sinks-3-of-6-ed3247545fbc)
* [Apache Spark Structured Streaming — Checkpoints and Triggers](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-checkpoints-and-triggers-4-of-6-b6f15d5cfd8d)
* [Apache Spark Structured Streaming — Operations](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-operations-5-of-6-40d907866fa7)
* [Apache Spark Structured Streaming — Watermarking](https://medium.com/expedia-group-tech/apache-spark-structured-streaming-watermarking-6-of-6-1187542d779f)

# **Reference**

* [https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html" \t "https://medium.com/expedia-group-tech/_blank)