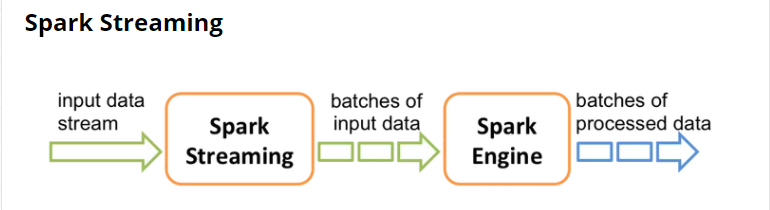
**Assignment22.1**

**Problem Statement**

**Spark Streaming using TCP Socket**



* Spark Streaming continuously receives live input data streams and divides the data into multiple batches.
* These new batches are created at regular time intervals, called batch intervals. The application developer can set batch intervals according to their requirement.
* Any data that arrives during an interval gets added to the batch.
* At the end of a batch interval,

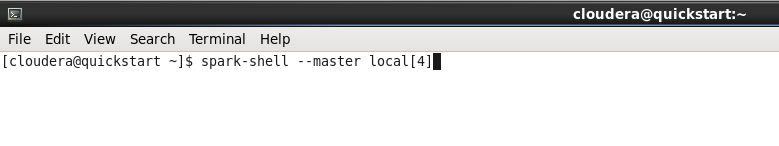
In this post, we will discuss the Spark streaming concepts by performing its demonstration with TCP socket.

Spark Streaming is an extension of core Spark API, which allows processing of live data streaming. In layman’s terms, Spark Streaming provides a way to consume a continuous data stream, and some of its features are listed below.

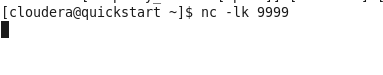
* Enables scalable, high throughput, and fault-tolerant data processing.
* Supports many input sources like TCP sockets, Kafka, Flume, HDFS/S3, etc.
* Uses a micro-batch architecture.
* Spark engine processes these batches.

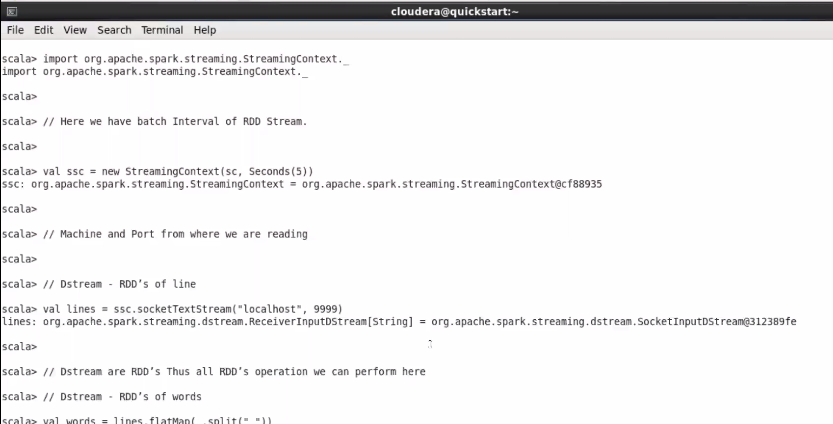
Spark Streaming is built on an abstraction called Discretized Stream or DStream. It represents the sequence of data arriving with time. Internally, each DStream is represented as a sequence of RDDs. A DStream is created from StreamingContext.

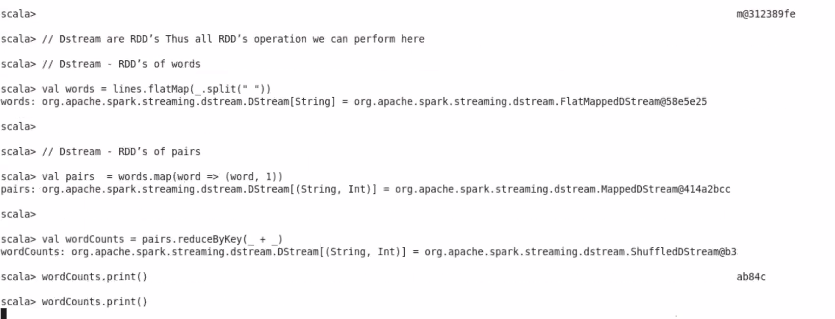
We can only have one StreamingContext per JVM.

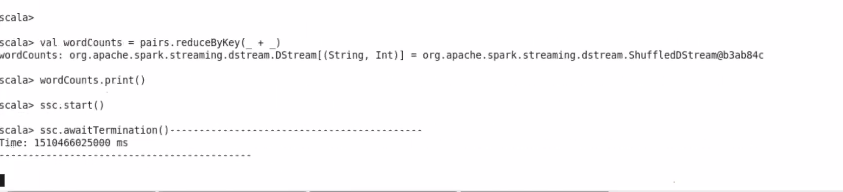


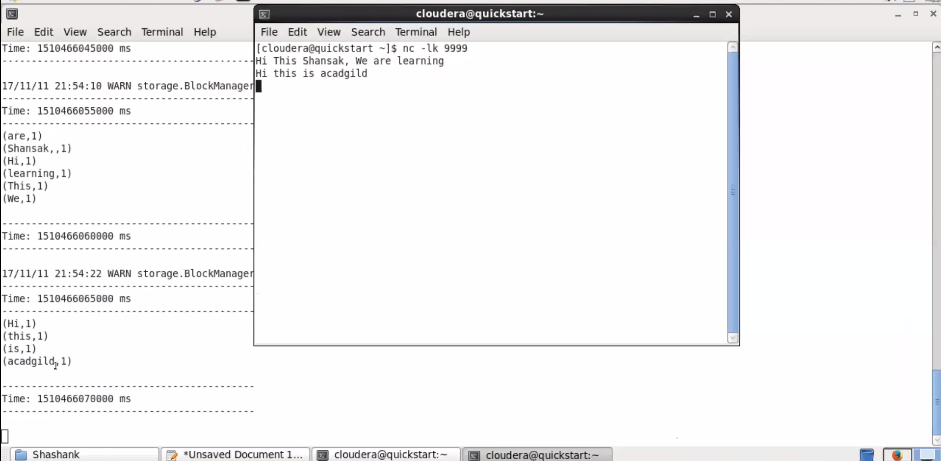
Parallely in another terminal, type **nc -lk 9999** to run netcat as a data server.











This terminal acts as a server where we will continuously feed the words, and our Spark Streaming code will count the number of occurrences (in a batch interval of 5 sec).

I ran this program in cloudera vm as you can see from the screenshot above.

As the interval is set to 5 sec it gives the wordcount after the first batch as we can see from the screenshot.