Problem Statement

Implement the below blog at your end and send the complete documentation.
https://acadgild.com/blog/spark-streaming-tcp-socket/

Solution:

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.

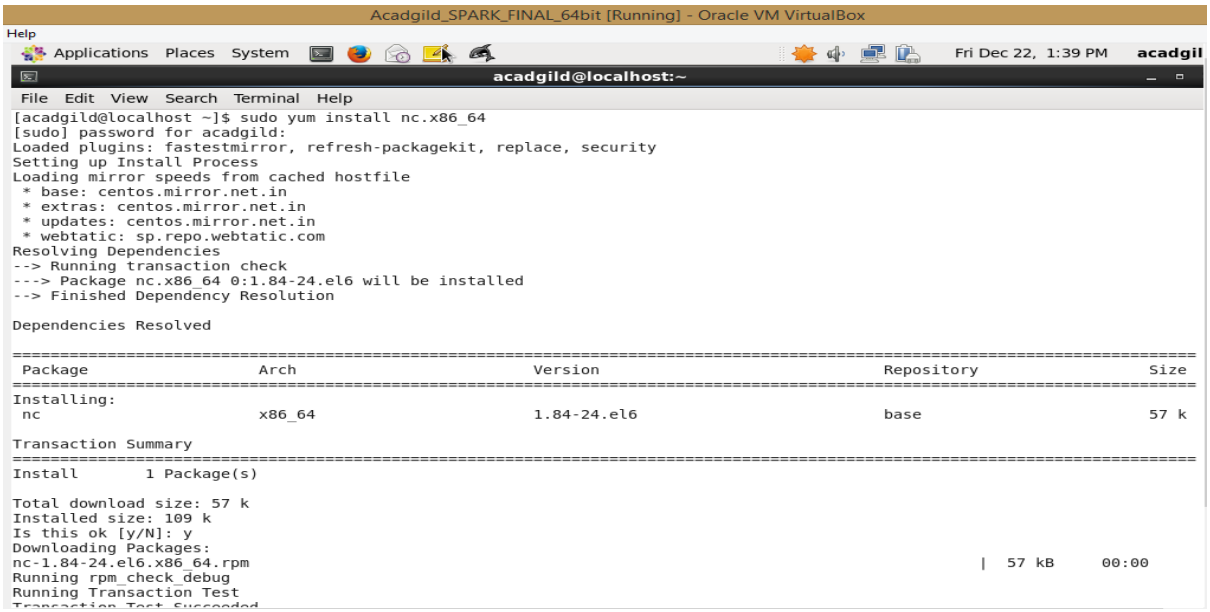
- Enables scalable, high throughput, and fault-tolerant data processing.
- Supports many input sources like TCP sockets, Kafka, Flume, HDFS/S3, etc.
- Uses a micro-batch architecture.

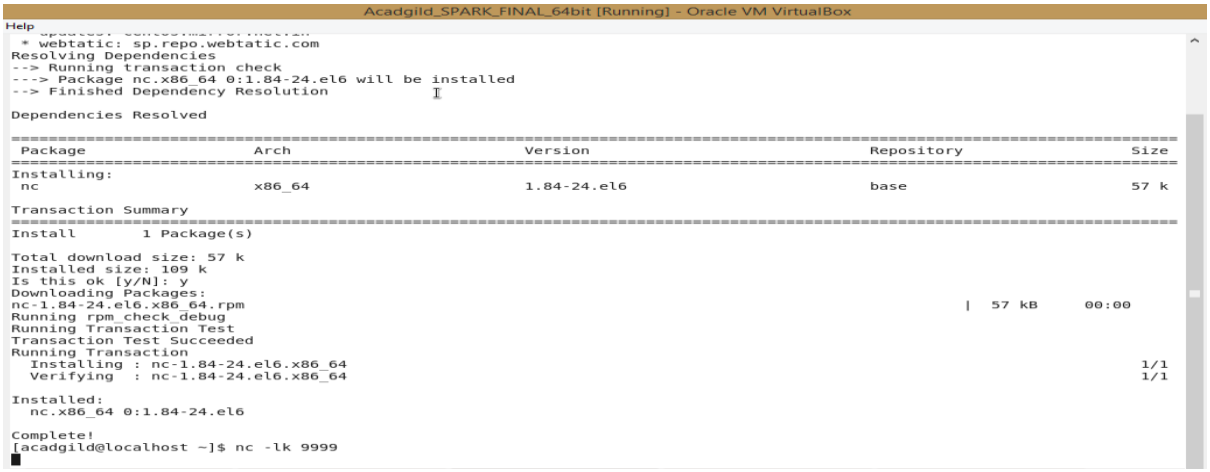
Spark Streaming continuously receives live input data streams and divides the data into multiple batches. These new batches are created at regular time intervals, called **batch intervals.** The application developer can set batch intervals according to their requirement. Any data that arrives during an interval gets added to the batch at the end of a batch interval.
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.
The Restriction here is, we can only have one StreamingContext per JVM.
Once a DStream is created, it allows two kinds of operations: *Transformation* and *Output operation.*
In this document, the Spark streaming concepts are discussed by performing its demonstration with TCP socket. We will perform the task of counting words in text data received from a data server listening on a TCP socket.

Firstly install the software to run netcat as a data server in one terminal

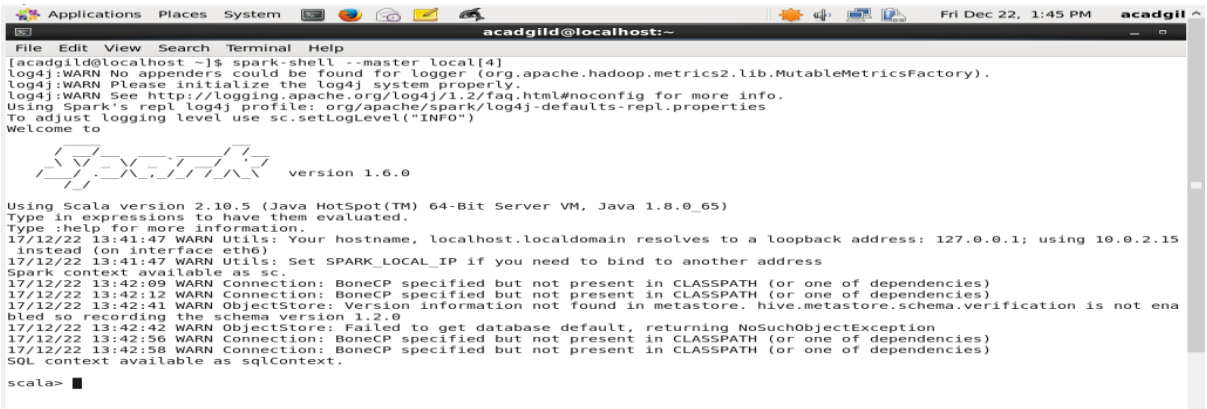
sudo yum install nc.x86_64

In the terminal type **nc -lk 9999** to run netcat as a data server.





In another terminal, invoke spark-shell



Code Expalnation:

- Importing the spark packages for streaming
- Declaring and defining the Stream Context object with the help of SPARK CONTEXT object sc and defining batch interval as 10 sec
- Defining the streaming source with the help of Spark Streaming object and pass the localhost details and port number as arguments
- performing flatmap operation to flatten the input data
- performing map operation to generate value 1 for each key/word and perform reduce operation to sum up the all the values of each key pair post successful map operation
- printing the word count for each word streamed from localhost 9999 port number.
- Start the computation
- Wait for the computation to terminate
- For Terminating streaming press Ctrl + C in spark shell it will terminate streaming

Code:

```
scala> import org.apache.spark._
import org.apache.spark._

scala>

scala> import org.apache.spark.streaming._
import org.apache.spark.streaming._

scala>

scala> import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.StreamingContext._

scala>

scala> val ssc = new StreamingContext(sc, Seconds(10))
ssc: org.apache.spark.streaming.StreamingContext = org.apache.spark.streaming.StreamingContext@77961693

scala>

scala> val lines = ssc.socketTextStream("localhost",9999)
lines: org.apache.spark.streaming.dstream.ReceiverInputDStream[String] = org.apache.spark.streaming.dstream.SocketInputDStrea
m@50798d39

scala>
```

```
scala>

scala> val words = lines.flatMap(_.split(" "))
words: org.apache.spark.streaming.dstream.DStream[String] = org.apache.spark.streaming.dstream.FlatMappedDStream@25983534

scala>

scala> val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
wordCounts: org.apache.spark.streaming.dstream.DStream[(String, Int)] = org.apache.spark.streaming.dstream.ShuffledDStream@30
35b9e6

scala>

scala> wordCounts.print()

scala>

scala> ssc.start()

scala>

scala> ssc.awaitTermination()-------------------------------------------
Time: 1513930660000 ms
-------------------------------------------
```

Writing the words in netcat for streaming in spark-shell

```
[acadgild@localhost ~]$ nc -lk 9999
welcome to the world of big data and hadoop
welcome to apache spark
welcome
welcome
welcome
to
hi
hadoop
hadoop
```

Output:

```
scala> ssc.awaitTermination()-------------------------------------------
Time: 1513930660000 ms
-------------------------------------------

17/12/22 13:47:49 WARN BlockManager: Block input-0-1513930668800 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513930670000 ms
-------------------------------------------
(big,1)
(hadoop,1)
(the,1)
(data,1)
(welcome,1)
(world,1)
(to,1)
(of,1)
(and,1)

17/12/22 13:47:58 WARN BlockManager: Block input-0-1513930678400 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513930680000 ms
-------------------------------------------
(spark,1)
(apache,1)
(welcome,1)
(to,1)

17/12/22 13:48:03 WARN BlockManager: Block input-0-1513930683600 replicated to only 0 peer(s) instead of 1 peers
17/12/22 13:48:06 WARN BlockManager: Block input-0-1513930686000 replicated to only 0 peer(s) instead of 1 peers
17/12/22 13:48:08 WARN BlockManager: Block input-0-1513930688400 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513930690000 ms
-------------------------------------------
(welcome,3)
```

```
-------------------------------------------
Time: 1513930700000 ms
-------------------------------------------

-------------------------------------------
Time: 1513930710000 ms
-------------------------------------------

17/12/22 13:48:32 WARN BlockManager: Block input-0-1513930711800 replicated to only 0 peer(s) instead of 1 peers
17/12/22 13:48:34 WARN BlockManager: Block input-0-1513930714000 replicated to only 0 peer(s) instead of 1 peers
17/12/22 13:48:38 WARN BlockManager: Block input-0-1513930718400 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513930720000 ms
-------------------------------------------
(hadoop,1)
(hi,1)
(to,1)

17/12/22 13:48:41 WARN BlockManager: Block input-0-1513930721000 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513930730000 ms
-------------------------------------------
(hadoop,1)

-------------------------------------------
Time: 1513930740000 ms
-------------------------------------------

-------------------------------------------
Time: 1513930750000 ms
-------------------------------------------

-------------------------------------------
Time: 1513930760000 ms
-------------------------------------------
```

Writing more words in netcat and see the result of streaming in spark-shell

```
[acadgild@localhost ~]$ nc -lk 9999
welcome to the world of big data and hadoop
welcome to apache spark
welcome
welcome
welcome
to
hi
hadoop
hadoop
welcome to hadoop world
welcome to hadoop world
```

Output:

```
17/12/22 13:53:28 WARN BlockManager: Block input-0-1513931008200 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513931010000 ms
-------------------------------------------
(hadoop,1)
(welcome,1)
(world,1)
(to,1)

-------------------------------------------
Time: 1513931020000 ms
-------------------------------------------

-------------------------------------------
Time: 1513931030000 ms
-------------------------------------------

17/12/22 13:53:55 WARN BlockManager: Block input-0-1513931035400 replicated to only 0 peer(s) instead of 1 peers
-------------------------------------------
Time: 1513931040000 ms
-------------------------------------------
(hadoop,1)
(welcome,1)
(world,1)
(to,1)

-------------------------------------------
Time: 1513931050000 ms
-------------------------------------------

-------------------------------------------
Time: 1513931060000 ms
-------------------------------------------
```