**Natural and Efficient ETL alternative using KAFKA, SPARK AND HDFS**

Following diagram indicates the flow of data in ETL alternative approach using Kafka, Spark Streaming and HDFS storage.

Incoming Transactional data

**KAFKA TOPIC**

Spark Streaming APIs will be used for reading from Kafka topic partitions in parallel. Parallelism is inbuilt in Spark

**SPARK STREAMING FOR KAFKA**

Spark Streaming APIs will ingest the data into HDFS files system

**HDFS (Hadoop Distributed File System) or equivalent**

Store the analyzed data in destination database or any other preferred location

**DATABASE / S3 / HDFS**

**SPARK CORE or SPARK SQL or HIVE SQL**

Reading from HDFS into Spark RDDs in parallel

**EXAMPLE APPLICATION FUNCTIONALITY FOR POC**

For the proof of concept, I have provided user a screen from where user can ingest data into Kafka topic.

A form is provided to the user to order a product. The product has 2 attributes, product name and brand name.

This form is available at following url:

**http://localhost:8080/IBMInsight/**

After spark performs the ETL analytics, the output is stored inside H2 database. The output for this proof of concept is going to be a grouping of number of products ordered by product name and brand name.

This output can be viewed at following url**:**

[**http://localhost:8095/IBMInsightResult/**](http://localhost:8095/IBMInsightResult/)

**NOTE: Please note that there is 2 minutes delay between the time an order is submitted and time it is reflected on the output screen.**

**STEPS TO RUN THE PROOF OF CONCEPT APPLICATION**

**START KAFKA**

1. Install kafka
2. Open command prompt and go to kafka installation directory

**Cd C:\EclipseWorkSpaces\kafka\_2.11-0.9.0.0**

1. Start zookeeper on default port 2181

**bin\windows\zookeeper-server-start.bat config\zookeeper.properties**

1. Open new command prompt and go to kafka installation directory and start kafka broker on default port 9092

Cd C:\EclipseWorkSpaces\kafka\_2.11-0.9.0.0

**bin\windows\kafka-server-start.bat config\server.properties**

1. Open 3rd command prompt and list existing topics and verify that there is a topic with name **myadopttopic**

Cd C:\EclipseWorkSpaces\kafka\_2.11-0.9.0.0

**bin\windows\kafka-topics.bat -list -zookeeper localhost:2181**

Myadopttopic

If there is not any topic with that name, create one using following command

**bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic myadopttopic**

1. In case you want to see messages arriving at kafka using command line consumer, run following command in a new command prompt window

**bin\windows\kafka-console-consumer.bat -zookeeper localhost:2181 -topic myadopttopic**

**MANUAL FOLDER CREATION FOR HDFS**

1. The data from kafka will be read by spark stream processor and stored on HDFS. To test the same on local windows environment, create a folder with name **KafkaSparkStreamingOutput** on c:\ drive on your machine.
2. Provide the path of this folder inside **application.properties** file within KafkaSparkStreamingReader and SparkDataProcessor projects for property **hdfs.folder.location**
3. Inside application.properties file within SparkDataProcessor projects, provide any temporary folder location on your machine for property **spark.sql.warehouse.dir**. This property is used by spark internally.

**INSTALL FOLLOWING 3 SPRING BOOT APPLICATIONS FROM GIT REPOSITORY AND EXECUTE**

1. Make sure that Kafka is running (Steps 1 to 5 from **START KAFKA** section)
2. Run the Spring Boot projects in following sequence
   1. **KafkaSparkSender**

Run **KafkaSparkSenderApplication** file. This application will provide a UI for user to submit order. This order data will be read and fed to Kafka topic named **myadopttopic** by this application. The UI is available at following location

**http://localhost:8080/IBMInsight/**

* 1. **KafkaSparkStreamingReader**

Run **KafkaSparkStreamingReaderApplication** file. This application will read the myadopttopic kafka topic using spark streaming APIs and write the same to HDFS. As we are running the application in local environment, it will write to folder configured by **hdfs.folder.location** property.

* 1. **SparkDataProcessor**

Run **SparkDataProcessorApplication** file. This application will **extract** all the data from HDFS location configured by **hdfs.folder.location** property into Spark’s abstraction called Resilient Distributed Datasets (RDD), it will perform the **transformation** of this data into intelligence and **load** the same into destination (in H2 database for the proof of concept). This application also has a UI which queries this H2 database for displaying the resultant data. The UI is available at following location

[**http://localhost:8095/IBMInsightResult/**](http://localhost:8095/IBMInsightResult/)

**NOTE: Please note that there is 2 minutes delay between the time an order is submitted and time it is reflected on the output screen.**