Spark Project Documentation

# Architecture Design

Shell Script Coordinator

# Architecture Details

## Data Source System

* Here we will use a **Python** script that will call **Twitter API** every **Five Minutes or More (Depending on your convenience and Laptop Specs)** and fetch the latest tweets that relate to your topic or query of choice and pushes those tweets to an arbitrary port that will be used for the communication between our data source and later stages in the pipeline.
* Make sure you understand the Twitter API Documentation correctly; this will help you better refine the Python Script that you are going to use for this step.
* Understand the attributes that you can pass in your **GET** request.
* Remember to use the **start\_time**, **end\_time** and **max\_results** to limit the number of tweets that you are going to fetch and avoid redundancy and repetition of fetched tweets.

## Data Collection System

* This part is mainly used as the data collector from the port, a link between **Twitter API** and a distributed storage system **HDFS**.
* You will keep the stream up and running always **(Long Running Job)** and it will keep receiving data from the port that was opened in the previous part of the pipeline.
* Inside it, you will need to parse the data coming from the port, give it structure **(Column Names and Data Types)**, and any light transformations that are needed, and then store the output to **HDFS** partitioned by the columns (Year, Month, Day, Hour) that are extracted from the **created\_at** column of each tweet.

## Landing Data Persistence

* Here is the place where the data are stored persistently in its base format, just partitioned by (Year, Month, Day, Hour) in a directory called “twitter-landing-data”, stored as **Parquet**.
* Also you will need to create a **Hive Table** on top of this directory to be used later in the pipeline.

## Landing To Raw ETL

* Here we run some **HiveQL** queries stored as **.sql Scripts** to extract the dimensions from the landing data.
* Dimensions can be “tweet-text (id, text)”, “users (id, name)”, etc… **Be Creative…**
* The output dimensions should be stored under a directory called “twitter-raw-data” and each new table must have “-raw” in its suffix
* The output tables should be partitioned by the same four columns.

## Raw To Processed ETL

* Here, the aggregations to create the final **Fact Table(s)** is going to be created.
* You will use a SparkSQL application for this step.
* It’s data source is going to the dimensions coming from the last step, this is why you need to make sure they were created carefully and correctly.
* The fact table(s) can contain count of tweets at a specific point in time, sum of # of favorites on the tweets related to a specific topic at a specific point in time, etc… also **Be Creative…**
* The output fact table(s) should be stored under a directory called “twitter-processed-data” and each new table must have “-processed” in its suffix
* Store the output table(s) should be created directly from the inside of the Spark App, look into “Store as table in Spark”

## Shell Script Coordinator

* This can be a single script or multiple scripts, depending on how you think about it.
* This will handle the calling of every patched part of our pipeline:
  + Twitter API Script
  + HiveQL Script(s)
  + SparkSQL App
  + And Making Sure each time it’s triggered, that the SparkStream is working
* You can use Linux Cron Jobs to handle triggering it periodically, or if you dare, **AirFlow.**

And that’s all folks,

Let me know if you need anything,

And Best Of Luck.