**NAYA College**

**Data Engineering Course Project**

**Contents**

[**Project Overview**](#_2lp48zb9qmcl) **3**

[**About This Document**](#_q9hs679x6qp5) **3**

[**Project Main Goals**](#_mhkegmkg7iuw) **3**

[**Project Primary Assumptions**](#_18w79c7zsmmg) **3**

[**Project High-Level Objectives**](#_9c8s0asmfjy) **4**

[**Project Low-Level Objectives**](#_wiwj2681n1ox) **4**

[**Project Success Criterias**](#_6fm4lmy0p0h5) **6**

[**Appendix**](#_9qvu9t933mev) **7**

# Project Overview

As a Data Engineer, you are asked to create a multi-purpose system that acquire data from the Twitter developers API (in real-time), which is based on certain keywords of interest. Once the data is collected, you will create a platform for performing data filtering, cleansing and enrichment to perform analytical operations for the organizational and business purposes and to transform raw data into knowledgeable data.

# About This Document

This document describes the technological goals and methods in which we will create a real-time system for storing data for archive and analysis purposes, configuring alerts and creating multiple data platforms for further analysis.

# Project Main Goals

* Create a solution architecture (design) according to the project description.
* Create data pipelines to support the project requirements.
* Implement the course acquired knowledge from different fields (infrastructure/logical) into a complete solution.

# Project Primary Assumptions

* The project will be done in pairs or individually (Recommended in pairs).
* The project will be done on the course VM.
* The following services/tools must be used for the project solution:

Kafka, Spark (Spark Streaming), Hadoop, HDFS, Hive, Impala, MySQL

* The students must create a Twitter account.
* The students will create the low level design of the project workflow.
* The data pipeline must be written in Python using the required APIs.

# Project High-Level Objectives

* Creating multiple data pipeline to support the following workflows:
  + **Source**: Twitter | **Destination**: HDFS Archive
  + **Source**: Twitter | **Destination**: HDFS for Hive/Impala reports/dashboards
  + **Source**: Twitter | **Destination**: MySQL for alerting on specific conditions
* Adding a monitoring process for the data pipeline workflow which will report on an event level to a log file/table for process investigating purposes.

# Project Low-Level Objectives

* Gain access as a Twitter developer
  + Open a Twitter account (if you don’t have, then create one)
  + Register as a Twitter developer (Go to <https://apps.twitter.com> and then click "Apply". After you fill the form, it takes Twitter a few hours to approve your application.)
  + Get your API keys (Consumer key, Consumer secret, Access token & Access secret token)
  + Be familiar with the [Twitter API docs](https://developer.twitter.com/en/docs/api-reference-index) and specifically the [Tweet object](https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object).
* Use python to Create a Kafka Producer (see starter code in the appendix section)
  + Extract specific column object from twitter API
  + Topic for HDFS archive in JSON format
  + Topic for real time alerts
  + Topic for HDFS Hive/ Impala tables
* Lucky for us, Spark has an integrated consumer (and producer) for Kafka. In order to work with it you will have to read about [structured streaming](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html) (not everything...) and [Kafka integration](https://spark.apache.org/docs/2.2.0/structured-streaming-kafka-integration.html) (where you will also find code samples).
  + You will have to add the following line which integrate Kafka with Spark: os.environ['PYSPARK\_SUBMIT\_ARGS'] = '--packages org.apache.spark:spark-sql-kafka-0-10\_2.11:2.4.1 pyspark-shell'

(and don’t forget to restart your kernel)

* Use spark structured streaming write data to HDFS for data analysis
  + Prepare the following Hive/Impala tables:
    - users - parquet file with the information of the (unique) users
    - last\_week\_tweets & last\_day\_tweets with (at least) the following columns:
      * User ID
      * User number of followers
      * Timestamp
      * Tweet text
      * Tweet geo (optional)
      * Text sentiment (see below)
  + Filter the data to include only tweets with at least 10 words.
  + Enrich the data: the company would like to be notified when an influential user says something extreme about the brand. To achieve that, we enrich the data with the text sentiment, as implemented by [TextBlob](https://textblob.readthedocs.io/en/dev/quickstart.html).
* Use python / Hadoop to Monitoring and error handling
  + The company would like to be notified about several events in real-time, and in this section we will define the required alerts:
    - If a user has more than X followers and their sentiment is higher than Y (or lower than -Y)
    - If a specific geo area has more than X tweets within a single day (optional)
  + Note that instead of raising real alerts, we will be writing them to a MySQL table called `alerts`, and having the following columns:
    - Timestamp
    - Alert name
    - Details (your customization)
* Finally, the tables you've created should be available for querying at all time. Use your favourite querying technique to create the following “reports”:
  + Daily basis
    - How many unique users have tweeted about our brand? How many of them are new?
    - What was the average sentiment?
    - Do we have a user who tweets more than X times within a single day? Maybe we should check what does he want…
  + Weekly basis
    - What are the tweets with the best and worst sentiment?
    - What are the tweets with the highest potential [reach](https://en.wikipedia.org/wiki/Social_media_reach)?

# Project Success Criterias

* The solution should answer to approximately 80% of the project requirements

(solution design, code etc.)

* The project should be submitted according to a defined date.
* The project solution should be demonstrated by the students at the course last project session.
* The solution will be examined on three levels:
  + Solution implementation percent.
  + Solution design (architecture diagram).
  + Python Code

# Appendix

Use the following code as a starting point for your own producer.

import json

import tweepy

from kafka import KafkaProducer

consumer\_key = 'TWITTER\_APP\_CONSUMER\_KEY'

consumer\_secret = 'TWITTER\_APP\_CONSUMER\_SECRET'

access\_token = 'TWITTER\_APP\_ACCESS\_TOKEN'

access\_secret = 'TWITTER\_APP\_ACCESS\_SECRET'

class TwitterStreamListener(tweepy.StreamListener):

def \_\_init\_\_(self):

self.producer = KafkaProducer(bootstrap\_servers='localhost:9092',

value\_serializer=lambda v: json.dumps(v).encode('utf-8'))

self.tweets = []

def on\_data(self, data):

text = json.loads(data)[u'text']

self.producer.send('tweets', text)

self.producer.flush()

print(text)

def on\_error(self, status\_code):

if status\_code == 420:

return False

def initialize():

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

api = tweepy.API(auth)

stream = TwitterStreamListener()

twitter\_stream = tweepy.Stream(auth=api.auth, listener=stream)

twitter\_stream.filter(track=['python, pyspark, spark'], languages=['en'])

if \_\_name\_\_ == "\_\_main\_\_":

initialize()

Note the [filter()](https://developer.twitter.com/en/docs/tweets/filter-realtime/api-reference/post-statuses-filter.html) method, which controls the tweets you will get from the API. Use it wisely to get all the relevant tweets for your brand (more about the API options find [here](https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/standard-operators)).