**Data Science full-stack developer – Case Study**

As a full stack Data science engineer I would like to build/test & support large-scale data processing systems and be an expert in the latest Big Data, NoSQL, advanced analytic tools & AI concepts. Here I will be demonstrating a complete data application lifecycle - data ingestion, modeling, processing & prediction solutions using AI techniques (Deep Learning/Machine Learning) to meet business needs of the organization. From my experienced I came up with the following problem solving solution.

**Use-Case: Airline on-time performance**

Have you ever been stuck in an airport because your flight was delayed or cancelled and wondered if you could have predicted it if you'd had more data?

Well as a Data Science Engineer I would like to come up with the solution by analyzing all the existing data (flight arrival and departure information from October 1987 to April 2008) and streaming real time data and come up with prediction and updated our customer with the real time data, so customer can benefitted from it and avoid all the consciences.

**The Framework Implementations:**

For implementing the framework I have used the following lists of tools:

* Spark 2.1.1
* Kafka 1.0.0
* Flume 1.8.0
* Hadoop 2.7.2
* Pig 0.17.0
* Zeppelin 0.7.1
* R Studio 1.1.423

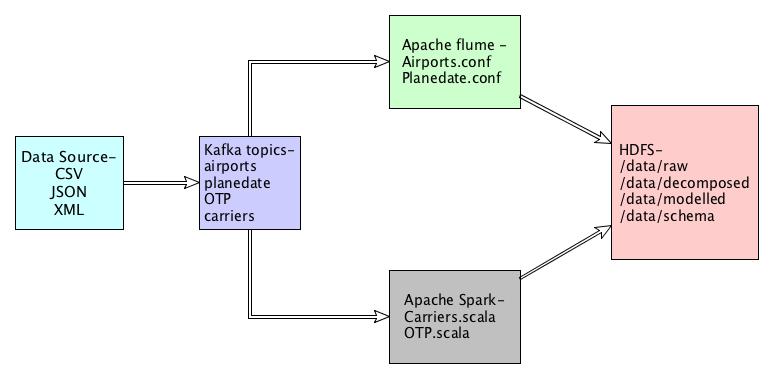
**Step by Step guide to install and configured these tools:**

* 1. **Install Spark**
* Here is the official site for downloading Spark
* <https://spark.apache.org/downloads.html>

tar –xvzf [spark-2.2.1-bin-hadoop2.7.tgz](https://www.apache.org/dyn/closer.lua/spark/spark-2.2.1/spark-2.2.1-bin-hadoop2.7.tgz)

* 1. **Install kafka**
* Here is the official site for downloading Kafka
* **https://kafka.apache.org/**
* <http://redrockdigimark.com/apachemirror/kafka/1.0.0/kafka_2.11-1.0.0.tgz>
* tar -xvzf [**kafka\_2.11-1.0.0.tgz**](http://redrockdigimark.com/apachemirror/kafka/1.0.0/kafka_2.11-1.0.0.tgz)
  1. **Install Flume**
* Here is the official site for downloading Flume
  1. **Install Hadoop**
* Here is the official site for downloading Hadoop
* **http://hadoop.apache.org/releases.html**
  1. **Install Pig**
* Here is the official site for downloading Pig
* **https://pig.apache.org/**
* To get a Pig distribution, do the following:
* Download a recent stable release from one of the Apache Download Mirrors (see [Pig Releases](http://hadoop.apache.org/pig/releases.html)).
* Unpack the downloaded Pig distribution, and then note the following:
  + The Pig script file, pig, is located in the bin directory (/pig-n.n.n/bin/pig). The Pig environment variables are described in the Pig script file.
  + The Pig properties file, pig.properties, is located in the conf directory (/pig-n.n.n/conf/pig.properties). You can specify an alternate location using the PIG\_CONF\_DIR environment variable.
* Add /pig-n.n.n/bin to your path. Use export (bash,sh,ksh) or setenv (tcsh,csh). For example:   
  $ export PATH=/<my-path-to-pig>/pig-n.n.n/bin:$PATH
* Test the Pig installation with this simple command: $ pig -help
  1. **Install Zeppelin**
* Here is the official site for downloading Zeppelin
* [**https://zeppelin.apache.org/docs/0.7.1/quickstart/install\_with\_flink\_and\_spark\_cluster.html**](https://zeppelin.apache.org/docs/0.7.1/quickstart/install_with_flink_and_spark_cluster.html)
* git clone https://github.com/apache/zeppelin.git
* cd zeppelin
* mvn clean package -DskipTests -Pspark-1.6 -Dflink.version=1.1.3 -Pscala-2.10

**Data Pipeline:**



**Implementation of the data Pipeline:**

* 1st of all I read the data from the csv file, which is proved in the case study.
* Next I have ported those data into the Kafka topics
* From topics Airport and planedate I have read the data through Apache Flume and save the data into HDFS under the directory /data/raw
* From topics OTP and Carriers I have read the data using Apache Spark(Spark Streaming) and save those data into HDFS under the directory /data/raw

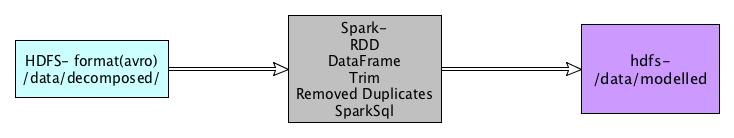
**Data decomposition:**



For adding timestamp and UUID, I have used Pig Latin Scripts

In which, I’ve read the contain from HDFS data/raw directory and those decomposed data I have added back to HDFS under the directory /data/decomposed in AVRO format.

**Data Modeling and Processing:**



* In data modeling, I have read the data from data/decomposed directory and convert the data into data frame by using the package com.databricks.spark.avro
* Next I trimmed those data, remove special characters and null
* For removing duplicate data I have used dataframe.distinct and save the refined data into HDFS into parquet format