# CLOUD COMPUTING CAPSTONE PROJECT PART 1

## INTRODUCTION

The goal of the Capstone project in this Cloud Computing specialization was to apply in practice the knowledge and skills gained throughout the different courses by analyzing a public transportation dataset of the US Bureau of Transportation Statistics. A set of questions had to be solved using the techniques and frameworks presented in the courses.

This report covers the first part of the Capstone project, which focusses on using batch processing systems to solve the presented questions. It contains an overview of the data cleaning process, how the system was constructed, the different optimizations used to improve the performance of the system and queries, the actual results and a conclusion to complete the report.

## Data extraction and cleaning

The RAW data set contained far more data than what was necessary for this project. Analysis of the data directory provided by the EBS volume, indicated that only the ‘airline\_ontime’ directory was of use for this project. The tables in this dataset contained about 80 fields, of which only the following were useful and retained during the cleaning process: FlightDate, Year, Month, Quarter, DayofMonth, DayOfWeek, Origin, Dest, UniqueCarrier, FlightNum, CRSArrTime, ArrTime, ArrDelayMinutes, ArrDelay, CRSDepTime, DepTime, DepDelayMinutes, DepDelay, Cancelled.

To process these files and extract the data. I created a temporary EC2 instance which copy the contents from EBS volume to S3 in the zip format. I used the new serverless paradigm called AWS Lambda to process zip and extract relevant data concurrently. In this managed service, AWS handles the operational challenges like scaling at a very low cost. I implemented 2 functions to process the data:

1. The function ‘​*handle\_zipfile*’​ performed the ETL task for a single zip file. From a high-level perspective the function (i) read the RAW zip file from S3 storage, (ii) decompressed it in memory, (iii) extracted the useful fields and (iv) wrote new CSV files onto S3 (uncompressed).
2. The function ‘​*get\_zipfiles*​’ (i) queried S3 and (ii) triggered an asynchronous execution of the first function for each zip file on S3.

The advantage is the automatic scaling this approach offered. Each file would take between 40 and 50 seconds to be processed, however as all 242 files are processed in parallel, the total time was less than 60 seconds. If the total number of files to be processed would increase drastically, the total processing time would remain in the same order of time.

*Note: The processing of the files, revealed there were 2 invalid files included in the data set. The contents of these files indicate the files actually did not exist. I excluded these from further processing.*

*2008/On\_Time\_On\_Time\_Performance\_2008\_11.zip 2008/On\_Time\_On\_Time\_Performance\_2008\_12.zip*