# http://www.ridgebizdev.com/img/Infare_new_logo_black.pngData Platform Engineer test exercise

## Task 1: ETL/copy of data from S3 to DataBase of your choice I could actually write a Terada Fastload script which would load data to an empty table very efficiently as this is block-wise parallel operation however as the exercises suggest usage of Spark I’ve setup following environment for those tasks:

* ElasticSearch 6.2.2 with Kibana & ElasticSearch-Sql  
  I didn’t used ES before rather Teradata, GreenPlum or pure Hive but it is being evaluated at my current company so I choose it for learning opportunity
* Scala 2.11 with Spark 2.2.1 project built via sbt & and run locally in Intellij. I didn’t used Scala much before (rather Pyspark (Python 2.6.6) with Pycharm & Spark 1.6.2) but I choose it because:
  + Again wanted to use it a learning opportunity
  + There was already some Scala code provided
  + Based on some initial research Scala provides functionality required for the exercises (like easy integration with ES, error handling with Try & Either)

Code for the task is available in GitHub:

<https://github.com/MaciejGajewski/ETL_TestExercise/blob/master/src/main/scala/task1/SimpleDataLoader.scala>

It contains code to read csv data into rdd, wrap it in DataModel case objects, calculating additional week column, measure processing time and save it in a log file. Then it creates 2 types of indexes in EL:

* One with all the data in 1 index
* Second is a group of indexes partitioned by week

Metrics are captured in the log files provided.

Due to limit of my laptop, access key to the files not working and time restrictions I processed only single file and couple of weeks.

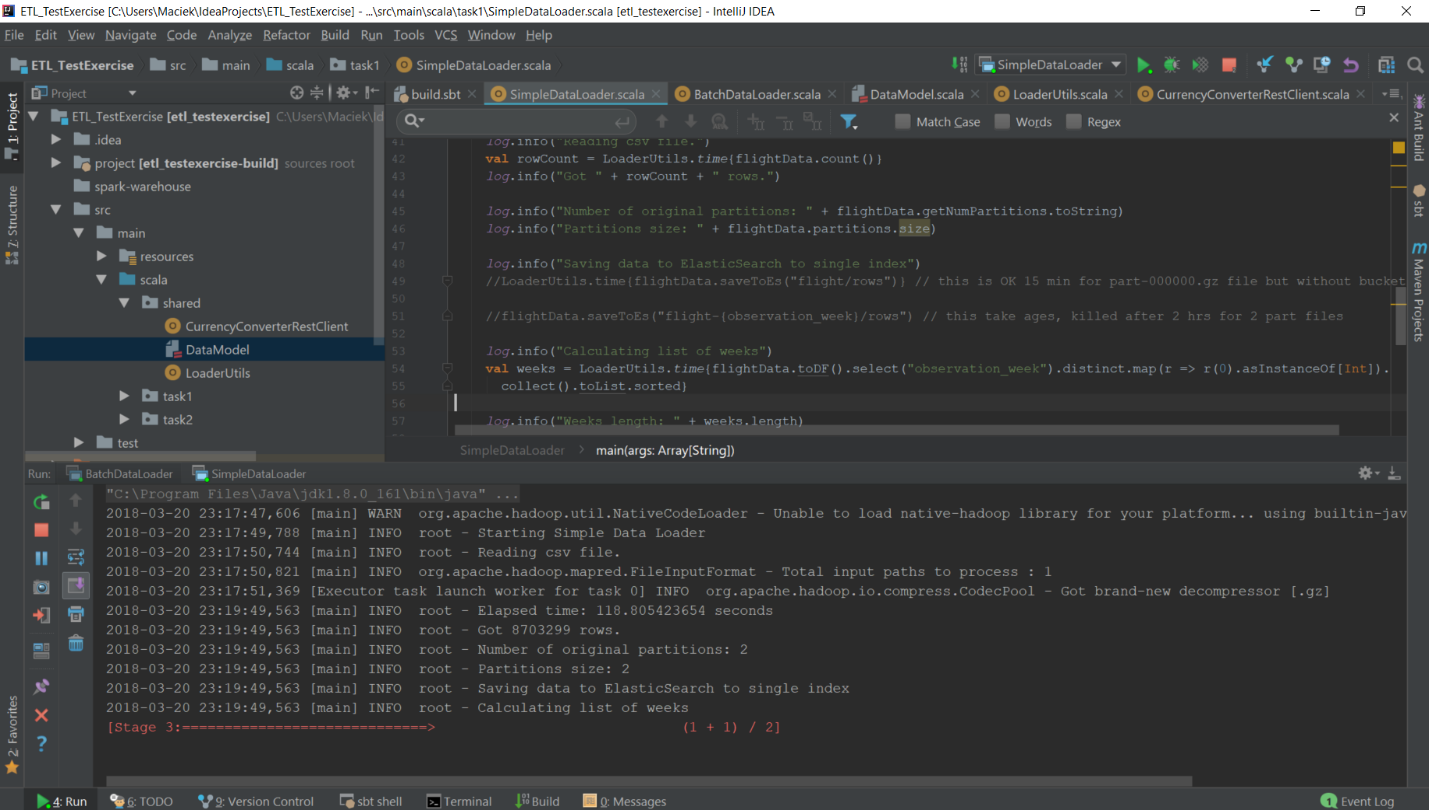
Having more time:

* I would try to run it on local Spark cluster via spark-submit or researched some free cloud environments.
* Enhanced code to calculate time metrics automatically

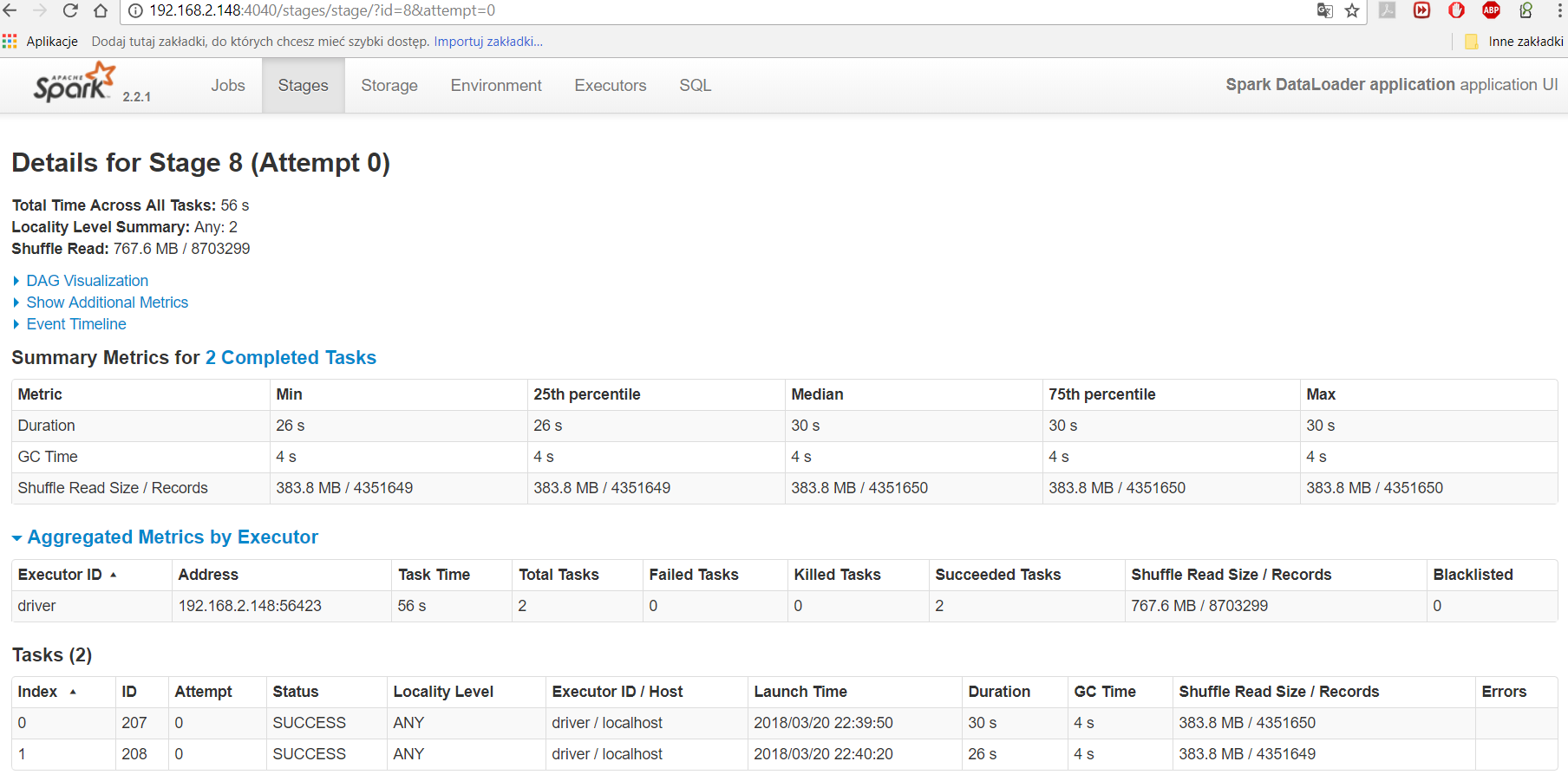
Other files are code for

* some utilities (data model, time metrics & web service call)
* logging config
* build config
* some excel calculation for a observed week based on existing number of days since 01.01.200

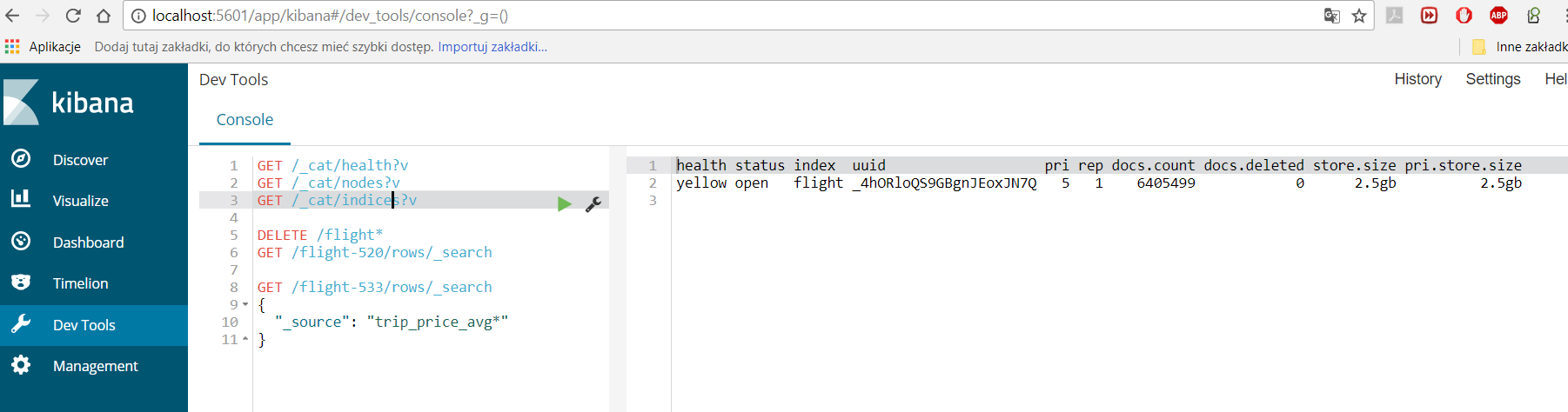
Intellij project:



Spark Console:



ES & Kibana:



## Task 2: Consuming Rest service during batched ETL process

Code for this task is available in <https://github.com/MaciejGajewski/ETL_TestExercise/blob/master/src/main/scala/task2/BatchDataLoader.scala>

It again reads csv file and wraps in DataModel cases objects. Then it prepares list of weeks but due to resource constraints I ran calculation only for couple of weeks. I don’t have much experience in building web services in Scala so I chose simple apache http client to make API calls to currency service.

Calls to current service REST api are done inside mapPartitions and map methods so they would be executed parallel if ran in multimode/multicore clusters. In order to manage service unavailability calls are wrapped in Try/Either object and successful attempts can be later filtered out and processed separately from failures.

If I had more time I would research proper/most robust way of building web service clients, added retries, waits etc.

## Task 3: (optional) Rest service exercise

Skipped this lacking experience in building web services in Scala or Python (I only once built web service client in Java using Spring).

## Task 4: (optional) Data mining exercise

1. Counts per week  
     
   select observation\_week, count(\*)

from flight   
group by observation\_week   
order by observation\_week   
limit 400

1. Duplicates check

SELECT

count(\*),

observed\_date\_min\_as\_infaredate,

observed\_date\_max\_as\_infaredate,

full\_weeks\_before\_departure,

carrier\_id,

searched\_cabin\_class,

booking\_site\_id,

booking\_site\_type\_id,

is\_trip\_one\_way,

trip\_origin\_airport\_id,

trip\_destination\_airport\_id,

trip\_min\_stay,

trip\_price\_min,

trip\_price\_max,

trip\_price\_avg,

aggregation\_count,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

out\_flight\_time\_in\_minutes,

out\_sector\_count,

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_departure\_date\_as\_infaredate,

home\_flight\_departure\_time\_as\_infaretime,

home\_flight\_time\_in\_minutes,

home\_sector\_count,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id

FROM flight2-823

group by

observed\_date\_min\_as\_infaredate,

observed\_date\_max\_as\_infaredate,

full\_weeks\_before\_departure,

carrier\_id,

searched\_cabin\_class,

booking\_site\_id,

booking\_site\_type\_id,

is\_trip\_one\_way,

trip\_origin\_airport\_id,

trip\_destination\_airport\_id,

trip\_min\_stay,

trip\_price\_min,

trip\_price\_max,

trip\_price\_avg,

aggregation\_count,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

out\_flight\_time\_in\_minutes,

out\_sector\_count,

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_departure\_date\_as\_infaredate,

home\_flight\_departure\_time\_as\_infaretime,

home\_flight\_time\_in\_minutes,

home\_sector\_count,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id

having count(\*) > 1

This does not execute in my ED due to performance issues

1. Determining PK

select

observed\_date\_min\_as\_infaredate,

observed\_date\_max\_as\_infaredate,

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

full\_weeks\_before\_departure,

count(\*) as cnt

from flight2-823

group by

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

observed\_date\_min\_as\_infaredate,

observed\_date\_max\_as\_infaredate,

full\_weeks\_before\_departure

having count(\*) > 1

Primary key looks to the flight route (out and home sectors) for a given week at different observation periods

1. One way flight vs Return flights

select is\_trip\_one\_way, count(\*)

from flight2-823

group by is\_trip\_one\_way

Result:

is\_trip\_one\_way COUNT(\*)

0 86320

1 2963

1. Top 5 direct one way flight with max number of observations

select

out\_flight\_sector\_1\_flight\_code\_id as out\_flight\_s\_1,

out\_flight\_sector\_2\_flight\_code\_id as out\_flight\_s\_2,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

sum(aggregation\_count) as sum\_aggregation\_count

from flight2-823

where

is\_trip\_one\_way = 1

group by

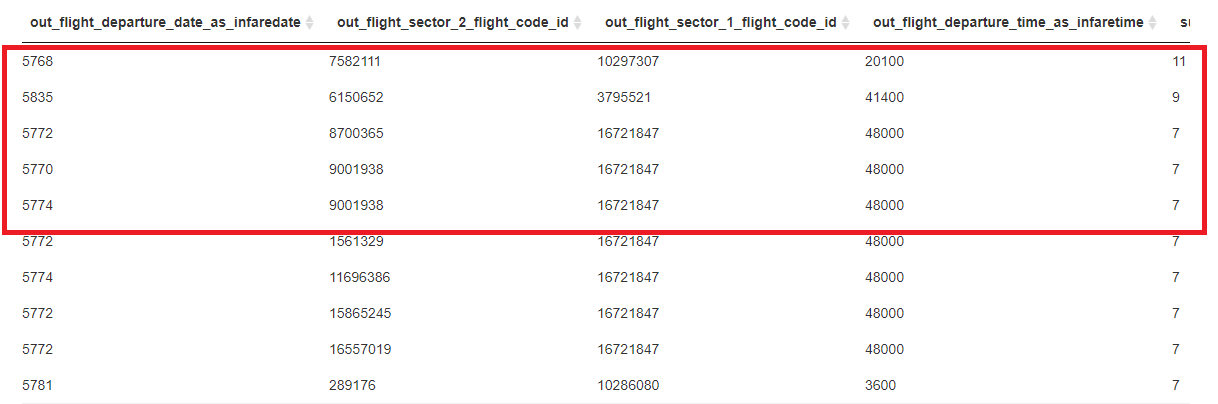
out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime

-- order by sum\_aggregation\_count desc



TODO: need to research how to do ordering, ranking and subquery in ES to properly do that.

Or use different DB.  
Also null are not handled properly (agg functions return nulls too). Need to research coalesce equivalent or load data with string “n/a” instead of null

1. Subsample

Using random\_score in Kibana. But this will have limit of 10k only.

GET /flight2-823/rows/\_search

{

"from" : 0, "size" : 10000,

"query": {

"function\_score": {

"query": {

"match\_all": {}

},

"functions": [

{

"random\_score": {}

}

]

}

}

}

TODO: Investigate how to use SAMPLE SQL function

1. Top 10 flights with largest min/max difference for a single departure  
     
   This would be sth like this:

select \* from (

select \* from (

select

observed\_date\_min\_as\_infaredate,

observed\_date\_max\_as\_infaredate,

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

full\_weeks\_before\_departure,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

row\_number() over (order by trip\_price\_max - trip\_price\_min desc) as price\_diff

from flight2-823 TabA

)

where price\_diff < 11

) TabB inner join flight2-823

on

/\* all the PK columns \*/

TODO: Would need to better understand the business problem - what is exactly single departure – as my understating is that price for a given flight (which occurs weekly on the same route) was observed multiple times. So are were looking for top10 observations of a flight, top 10 distinct flights, top 10 routes?

1. Compute average price per origin airport

select trip\_origin\_airport\_id, (trip\_price\_avg \* aggregation\_count) / sum(aggregation\_count)

from flight2-823

group by trip\_origin\_airport\_id

1. 10% cheapest flights

insert into TabD

select \* from (

select

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

trip\_price\_min,

row\_number() over (trip\_price\_min trip\_price\_min asc) as trip\_price\_rank

from (

select

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime,

min(trip\_price\_min) as trip\_price\_min

from flight2-823

where

out\_flight\_departure\_date\_as\_infaredate - observed\_date\_min\_as\_infaredate >= 14

group by

out\_flight\_sector\_1\_flight\_code\_id,

out\_flight\_sector\_2\_flight\_code\_id,

out\_flight\_sector\_3\_flight\_code\_id,

home\_flight\_sector\_1\_flight\_code\_id,

home\_flight\_sector\_2\_flight\_code\_id,

home\_flight\_sector\_3\_flight\_code\_id,

out\_flight\_departure\_date\_as\_infaredate,

out\_flight\_departure\_time\_as\_infaretime

) TabA

) TabB

where trip\_price\_rank < (select count(\*)/10 from flight2-823) TabC

select count(\*) from TabD

select trip\_origin\_airport\_id || ‘-‘ || trip\_destination\_airport\_id

from TabD

group by trip\_origin\_airport\_id || ‘-‘ || trip\_destination\_airport\_id

## Task 5: (optional) Visualization exercise

TODO