FEXCO ETL PoC

Requirement specification.

Goals and context.

Our main goal will be to create a high performance ETL extractor, whose main purpose will be to process information from our many different sources of data.

Currently many of the business processes, whose main purpose had been the processing of data are limited by the number of rows that an excel file is capable of handling, in such case this labor must be done in a manual way in order to perform data reports. In many cases we've see problems in the data processing due to several factors, we're actually relying in manual processes when we're speaking about data transformation, as a consequence this has leaded us to several failures on our reports, and that's is the main issue inside the reporting area. By now this problem has been reported by staff from credit to consumer department at FEXCO, but we're sure it's a widespread problem in the entire corporation.

We can think that there's only way to reinforce the accuracy of the data reporting process, and we are thinking about this like a standardized process. The advantages of a standardized process are well known right know, but we'd like to emphasize that like every automated process this will give us, a solid base to measure stability and with the inclusion of self-driven process we'll be on the track of a modern reporting platforms; and even we are willing to stablish a modern workflow for our data processes, so this could lead us to take advantage of technology and scalability in any desired moment.

By consolidating the large amount of fragmented processes and having an engine capable of growing according to the expectations, there is the opportunity of creating one of the most valuable software projects inside our company.

User Personas

The following are the two key user personas that will be consuming our reports.

	Information
User Type	Internal User
Occupation	Reporting Specialist
Name	John Doe
Age	36
Education	Bachelor’s Degree in Computer Science.
Location	Wessex, United Kingdom.
Objective	John works long hours in the preparation and curation of several reports for the Credit Consumer Area, he wants to be able to bypass the manual generation of the report “Blacklist”.
Additional Notes	-

User Type	Internal User
Occupation	Reporting Specialist
Name	Jane Doe
Age	27
Education	Bachelor’s Degree in Economics.
Location	Dublin, Ireland.

User Personas

The following are the two key user personas that will be consuming our reports.

	Information
Objective	Jane works long hours daily based in the preparation of several reports for the Credit Consumer Area, she wants to be able to bypass the manual calculation process for the “Delinquency-Rate”, and because of that the generation of the report itself.
Additional Notes	-
User Type	Internal User
Occupation	SysAdmin
Name	Eunice Scott
Education	Bachelor’s Degree in Computer Science.
Age	42
Objective	Automate task execution of processes in the IT area. Scheduling jobs, and automated tasks that will occur in a daily based timetable.

Users Stories

Reporting Specialists

- As a **reporting specialist**, I want to see my report generated in an automated scheduled task and being able to download the processed file in csv format.

Systems Administrators

- As a **system administrator,** I want to see the execution roadmap of the component in a log file for each execution set.

Non-Functional Requirements:

- The application must be constructed in order to support scalability.
- The application must be capable of running on scalable infrastructure.
- The application itself must not exceed an upper threshold of 2.5 GB usage of RAM.
- The application itself must process in a daily based agenda.
- The application must run the same in every environment.
- The application must be able to support at least 2 Million records.
- The application must run in a 24/7 environment, so it must no generate latency in the reading process of the data.
- The TTL for this process will be "15 minutes".

Functional Requirements:

- The application must be able to read a data source in format "CSV".
- The data source "CSV", must be available as an input for our process.
- The application must be able to apply a transformation in the "Input" and produce an output.
- The product of the transformation will be saved in a "DataLake", whose accessibility will be granted to all users.
- The execution of the component, will construct a Log file who can be supportive in the tracking process of the execution plan of the application.
- The system needs to be orchestrated by a pipeline in the CI/CD process.
- The execution will be scheduled in an automated driven process.
- The system itself at this point must be able to generate "Scoring Report" & "Blacklist Report"
- The application will embrace acceptance testing powered by cucumber for each business report definition.

- The application must be built, unitary tested, acceptance tested, released by an orchestrated pipeline.

Product Design

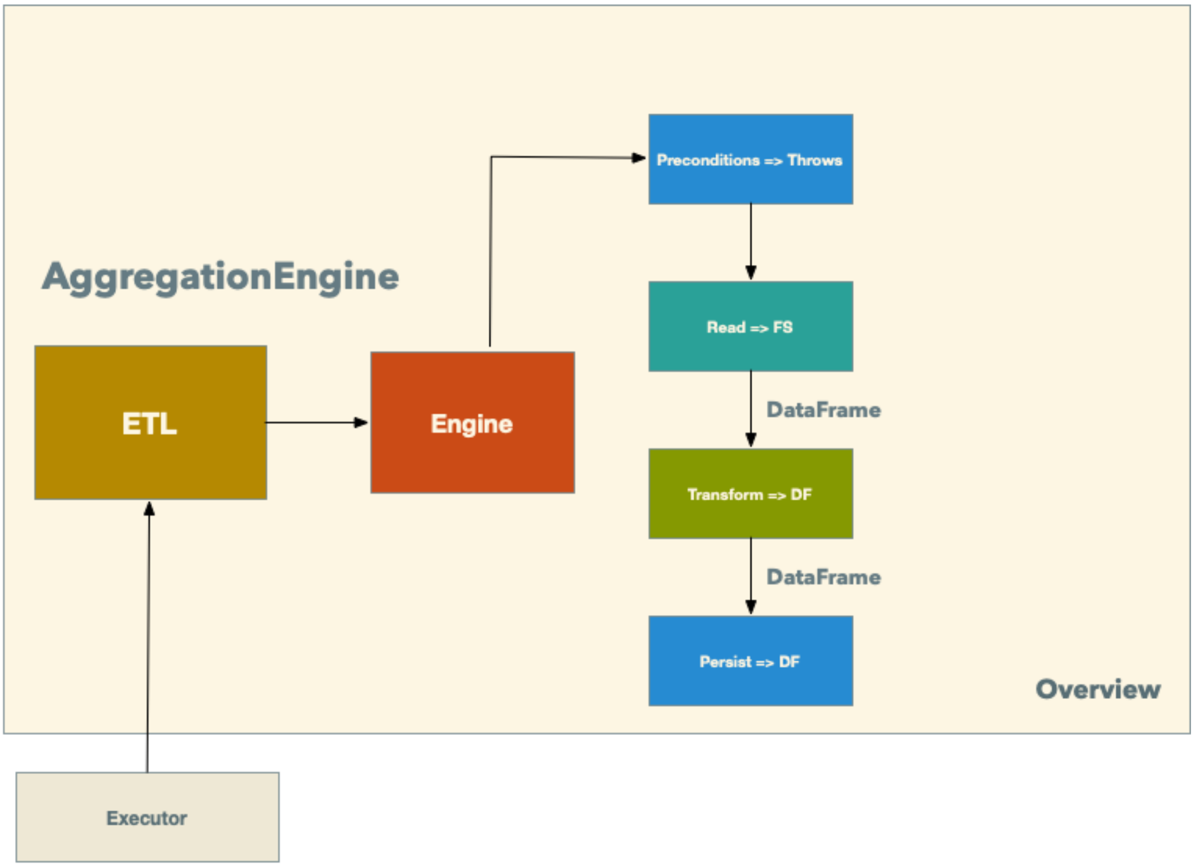
Topic	Selection	Justification
Application Name	AggregationEngine	This should provide value to the business; it'll be an engine for aggregations and transformations.
Environment	Spark / Hadoop	This will provide us with a huge capacity in terms of growing in the future, and obviously bypass the records number limitation, along with the solid framework related to all the development tools available.
Language	Scala	This would be easy to implement, and in the most cases any Java backgrounded developer should be able to jump in. In fact, it's a native language specification able to run in a Spark environment.
Orchestration Pipeline CI	Jenkins	This is the standard in the industry, so we'll have a well community supported product. It has demonstrated trough years, being a solid allied when we're speaking about the lifecycle of a software component. This is also very customizable with tons of artifacts who will

Topic	Selection	Justification
		provide us support in a fast-paced environment.
Run Everywhere	Docker	This would provide us with containerized solutions.
Orchestrating Containers	Compose	This would provide us with a huge number of tools, in order to improve our deployment process. And in the future, we can change this one with a more robust one, e.g. Kubernetes.
SO	Linux	This is the king and ruler of the computing world, as a de-facto standard.
Scripting	Bash	Standard across UNIX environments.
Test Acceptance	Cucumber	Will provide us with the ability of transforming user stories in a well-documented format and ensure the quality, for a CI-CD environment.
Unit Testing	Junit	This has been the de-facto standard in the industry when we're speaking about software development with Scala-Java.
Models of Transformation	Coded in Scala	The model's definition will be described in further section.

Assumptions:

- RAW data will be available for the execution.
- Valid model transformation will be defined in the component.
- Bash script modeled will be executed to produce the output labeled as "RESULT".

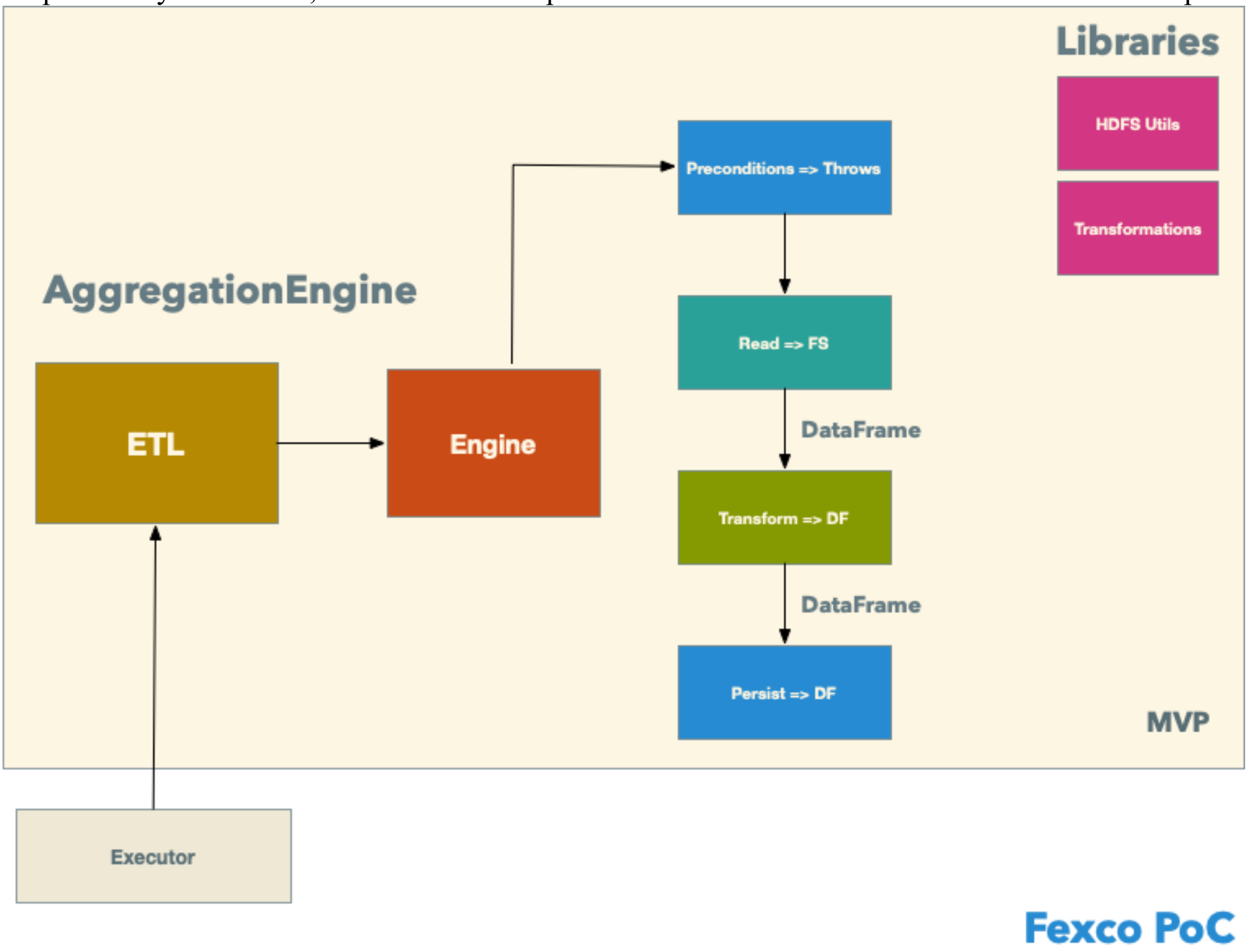
About Application



Fexco PoC

Diagram Application

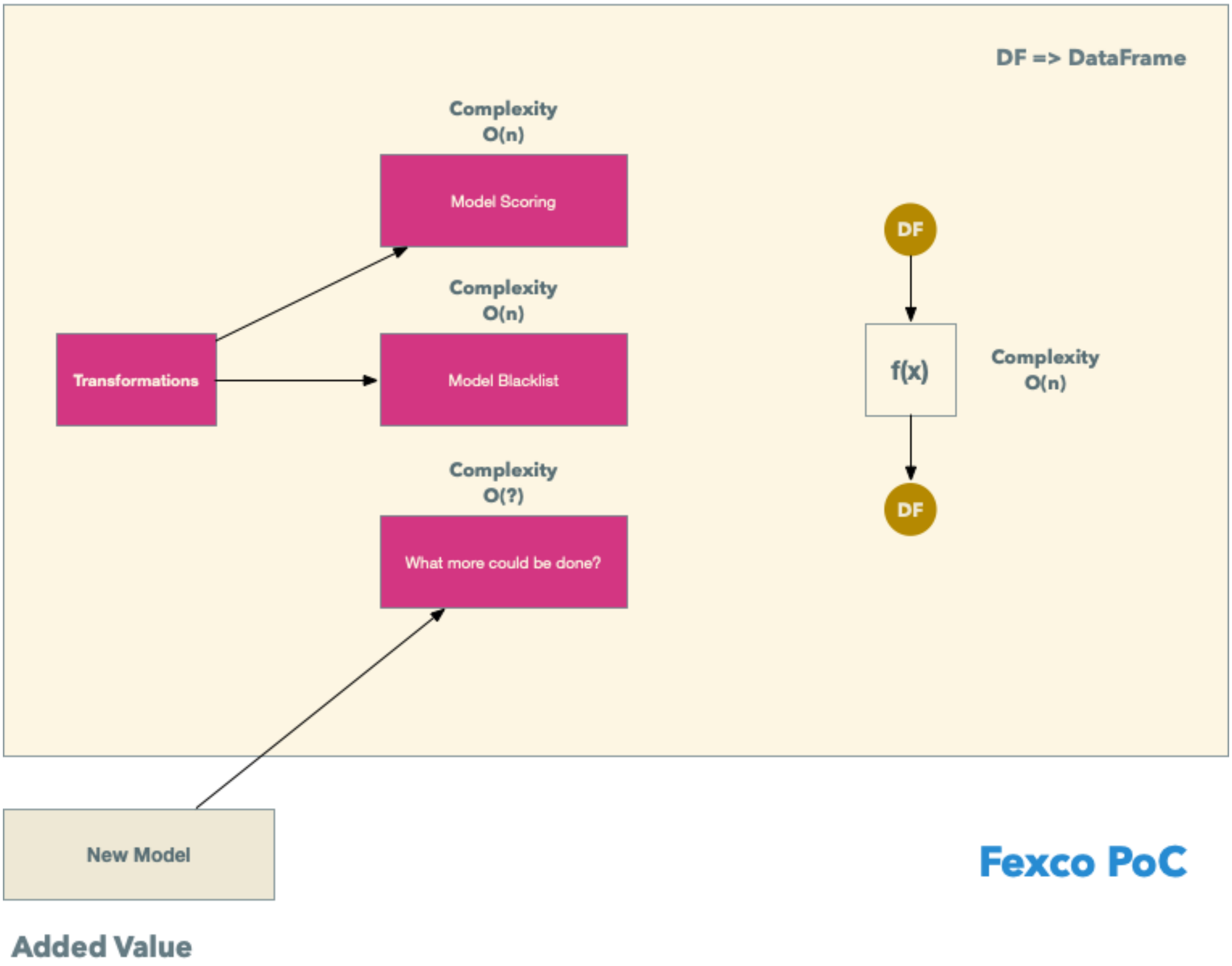
As previously mentioned, we need to develop a solution who embraces the next solution roadmap:



Component	Purpose	Core Values
ETL	Main executor, orchestrates the parametrization core for the execution.	Being decoupled will provide us with the ability, in the future of parametrization in different ways. (JSON e.g.)
Engine	Orchestrates the chain of events, that need to succeed in order to have a fully working process.	Preconditions: We can define conditions to verify prior the execution; reading, transformation and persistence tasks will be orchestrated by this component.

Component	Purpose	Core Values
HDFS Utils	Inputs and Outputs	Provide us right now with implementations to handle CSV, but we can add tons of data sources format and data output formats. E.G. Parquet, S3, JSON in perfect harmony with the Engine Orchestration.
Transformations	Collection of Models (Transformations)	Decoupled component library shaped, will provide us with a collection of available transformations and obviously an easy implementation path for new ones.

About Models



DF => DataFrame

Complexity
O(n)

Model Scoring

Complexity
O(n)

Transformations

Model Blacklist

Complexity
O(?)

What more could be done?

DF

f(x)

Complexity
O(n)

DF

New Model

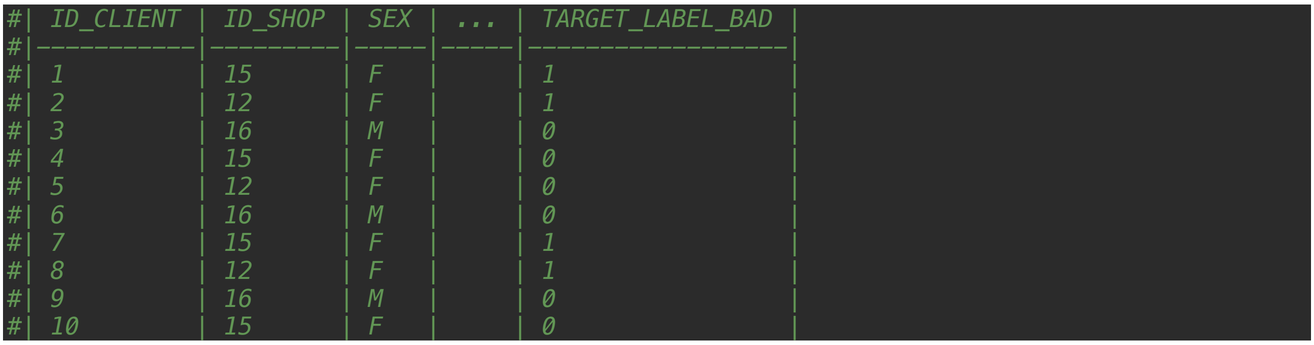
Fexco PoC

Added Value

About Model for “Scoring Report”

*A **scoring report** has her basis in the delinquency rate, it's a very common term in the industry. In a simplified version we'll have the next:*
Given: A total number of data elements in a credit loan portfolio.
When: We extract the ones, who are presenting overdue in their payments.
Then: We divide the loans with overdue in their payments / total number of loans.
This will give us the "Delinquency Rate Index" -> Scoring * 100 (Here we multiply to obtain a score from 0 – 100)
E.G.
Scenario:
Extract a scoring report from the data.
Given: A sample data-source, with the "TARGET_LABEL_BAD" column containing (0,1)

- 1 means overdue in the payment for the period.
- 0 means everything ok with the payment for the period.



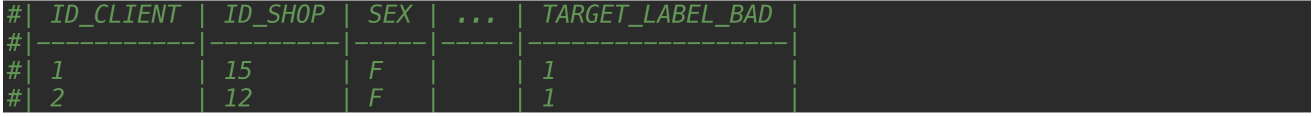
```
#| ID_CLIENT | ID_SHOP | SEX | ... | TARGET_LABEL_BAD |
#|-----------|---------|-----|-----|------------------|
#| 1         | 15      | F   |     | 1                |
#| 2         | 12      | F   |     | 1                |
#| 3         | 16      | M   |     | 0                |
#| 4         | 15      | F   |     | 0                |
#| 5         | 12      | F   |     | 0                |
#| 6         | 16      | M   |     | 0                |
#| 7         | 15      | F   |     | 1                |
#| 8         | 12      | F   |     | 1                |
#| 9         | 16      | M   |     | 0                |
#| 10        | 15      | F   |     | 0                |
```

When: I apply a transformation of type "scoring"
Number of elements => Total number of elements in the dataset.
Number of elements with overdue => Total number of elements marked in the dataset with "TARGET_LABEL_BAD" equals to one.
So, the "delinquency_rate" => (4/10) * 100 = **40**
Then: I will have the delinquency rate calculated for the dataset.

About Model for "Blacklist Report"
A **blacklist report** has her basis on the data extraction from a data set, of the entire credit loans marked as "with overdue payments".
Given: A total numbers of data elements present in a credit loan portfolio dataset.
When: We extract the ones, who are presenting overdue in their payments.
Then: The subset will be marked as "Backlist".
E.G.
Scenario**:**
Extract a blacklist report from the data.
Given: A sample data-source, with the "TARGET_LABEL_BAD" column containing (0,1)

- 1 means overdue in the payment for the period.
- 0 means everything ok with the payment for the period

When: We apply a transformation of type "blacklist" to the data-source origin



```
#| ID_CLIENT | ID_SHOP | SEX | ... | TARGET_LABEL_BAD |
#|-----------|---------|-----|-----|------------------|
#| 1         | 15      | F   |     | 1                |
#| 2         | 12      | F   |     | 1                |
#| 3         | 16      | M   |     | 0                |
#| 4         | 15      | F   |     | 0                |
#| 5         | 12      | F   |     | 0                |
#| 6         | 16      | M   |     | 0                |
#| 7         | 15      | F   |     | 1                |
#| 8         | 12      | F   |     | 1                |
#| 9         | 16      | M   |     | 0                |
#| 10        | 15      | F   |     | 0                |
```

Then: The output must not contain any value with “TARGET_LABEL_BAD” = 0, this means all the data was filtered by just the “loans with overdue in the payment”.

```
#| ID_CLIENT | ID_SHOP | SEX | ... | TARGET_LABEL_BAD |
#|-----------|---------|-----|-----|------------------|
#| 1         | 15      | F   |     | 1                |
#| 2         | 12      | F   |     | 1                |
#| 4         | 15      | F   |     | 1                |
```

Risks

Due to limitations in the hardware and infrastructure available for our product, we need to stablish the basis of the analysis in the complexity and runtimes of the solution.
Let’s take a quick look on the NFR:

- 2.5 RAM
- 15 Minutes of execution.

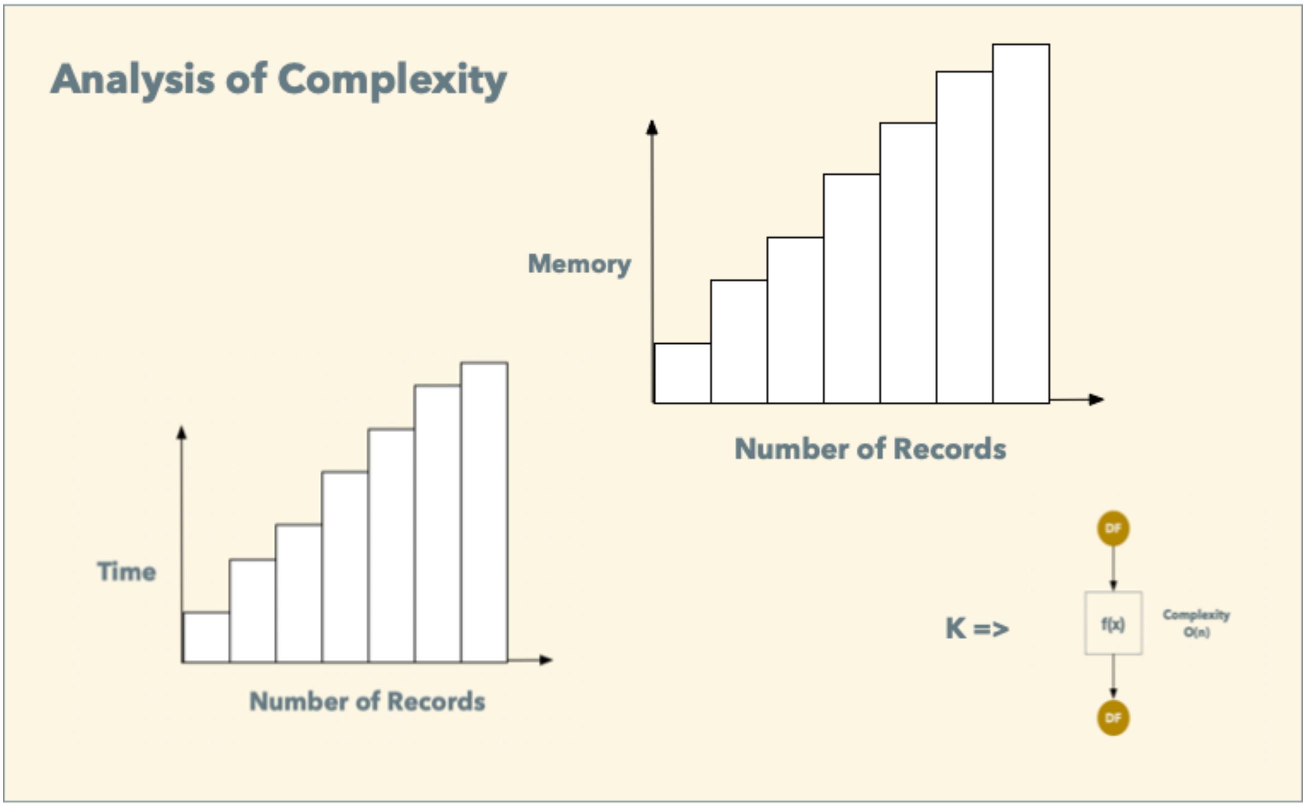
If we take a closer look in a Big(O) notation we’ll see something like this:

n = number of records; n > 2 Millions of records.
t = time.
r = RAM.
k = Complexity of the transformation model.

If we stablish a correlation between the variables:

$t = n * k$
$r = n * k$

In the best of the cases we can state the next syllogism:
If *t* is representing time, r representing memory usage, k the *cost of the operation (Time Complexity – Space Complexity)* and *n* as the total number of records, it’s merely evident that in certain point that I’ll lack of resources at certain point, event when we assume that the “k” -> will be based on a linear growing we’re going to be outpaced by the “physical limitations”. It’s nice to mention this and how I’ve theoretically tested the component, in a single cluster machine. (The complexity of the algorithm ensures the predictability of the component even at a distributed cluster, here resides the importance of this analysis). We’ll need to perform stress testing with the infrastructure in order to have the upper limits for the system that we’ve built. Theoretically this should be done like the next image:



Analysis of Complexity

Fexco PoC

N	T	R	Verdict
1,000,000	16s	500MB	Passing
2,000,000	30s	1000MB	Passing
3,000,000	45s	1000MB	Passing
4,000,000	60s	1000MB	Passing
?	900	2.5GB	Threshold

About Workflow

It has been described in very depth detail in their own isolated repository.

You can find more info clicking here.

https://github.com/rkobismarck/continuous-integration