QS – ETL software selection

Niels van der Burg

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Inhoud

[Revisions 2](#_Toc101970001)

[Introduction 4](#_Toc101970002)

[Methodology 4](#_Toc101970003)

[Requirements 5](#_Toc101970004)

[Systems researched 6](#_Toc101970005)

[Apache Airflow 7](#_Toc101970006)

[Apache NiFI 8](#_Toc101970007)

[Talend 9](#_Toc101970008)

[AirByte 10](#_Toc101970009)

[In house solution 11](#_Toc101970010)

[No ETL solution 12](#_Toc101970011)

[Usability matrix 13](#_Toc101970012)

[Conclusion 13](#_Toc101970013)

[Sources 14](#_Toc101970014)

# Introduction

Quantified Student (QS) is a project to show a students performance in a dashboard based of public data. In this research, the selection for an ETL tool is outlined.

The reason an ETL tool was researched is because a central method of data collection had to be established. This is because QS aims to show data from several sources. Thus needing a method to collect the data from.

# Methodology

Initial phase of this research will focus on proposing several requirements for the system. Once these are verified by the product owner. Once verified these will be prioritized via MOSCoW. After which a selection of tools will be analysed and compared to the requirements. From this a top 3 selection will come out where one system will be selected to proceed forwards.

# Requirements

Before selecting and analysing tools. The requirements for the tool need to be established. These are:

* Collect from web sources
* Collect from database
* Collect from API
* Allow for request of user specific queries (such as canvas API key)
* Write to database
* Allow for transformation operations
* Is actively supported
* Is open-source
* Adding data sources can be done with minimal code (only for transformation)
* Schedule the collection of data
* Logs to see data collection failure
* Has made the proper security steps to prevent hacking attacks
* Proper documentation for future teams

There requirements are verified on 28-04-2022. Out of this interview came the following additional requirements:

The total picture of all requirements will look as following:

(

MODEL-VIEW-CONTROLL

)

# Systems researched

The following systems were researched for this analysis:

* Apache airflow
* Apache NiFI
* Talend
* AirByte
* In house solution
* No ETL processes

Other systems do exist but were not taken along for this research to limit the time spend researching. Off note is that streaming data was not considered desirable for this project. This is because users will never view their own data liv, since it is a reflection on their behaviour. Thus not warranting live data.

## Apache Airflow

Apache Airflow was started in October of 2014 at AirBnB. It became an open-source solution in June of 2015. In March of 2016 it joined Apache the Software Foundation's Incubator program. In January of 2019 it became a top-level project.

The main focus of Apache Airflow is to support programmatic authoring, scheduling and monitoring of workflows. It is not build to support ETL processes at it core but rather a generic overview of scripts.

Several of the use cases describe using Apache Airflow for ETL processes. Including Adyen, big-fish-games and Adobe.

It support a wide variety of data sources, including:

* Apache Druid
* Docker
* MySQL
* HTTP

Other data sources can be collected from but are not directly supported by the Apache Airflow project. Sources such as graphql must be scripted within the script.

It support a non-cyclical process of workflow. Meaning it cannot return on functions previously used later on in the process.

Scripting within Apache Airflow does have quite a learning curve. Users must initially study its format before applying it within projects.

All new data sources must programmatically be added. There is no option to used a user interface for adding new data sources.

## Apache NiFI

Apache NiFi is a software project that focusses on designing and automating data flow between software systems. It’s based of the NiagaraFiles software developed by the NSA. It was made open-source in 2014 as part of the NSA’s technology transfer project. It is build upon Java.

Apache NiFi uses a user interface to construct ETL processes. From this several operations are developed for the processing of data. Data can be collected from several sources such as:

* HTTP
* FTP
* Flat file
* Twitter

Apache NiFi is used by the following companies:

* GoDataDriven
* Ona
* Payoff
* Slovak Telekom

(

TOEVOEGING VAN AANGEPASTE BRONNEN MOET ONDERZOCHT WORDEN EN PRELADEN VAN DATA

)

# Talend

(

ONDERZOEK MOET NOG GEDAAN WORDEN

)

# AirByte

Is a semi-open source ETL software. It support obtaining data from external sources and pushing them to a central server. The reason for stating semi, is that a section of the software has a different license along with product keys. This change in license does not affect the QS project.

Not all sources needed for the project are support but can be made using de AirByte CDK. Airbyte advertises that the modelling of this data should at most take 2 hours. Documentation on creating custom connectors is provided and several tutorials are available as well.

Airbyte takes an alternative approach to ETL, namely ELT. Meaning data is stored prior to transforming. The reason AirByte choices for ELT is due the flexibility it provides organizations. Since you are collecting the data from a central state and allows for merging on different variables.

Besides that Airbyte provides monitoring, notifation and debugging tools to find and locate issues in code. It supports the collecting of data via API streams, this is because Airbyte allows for customized collection methods.

Airbyte also provides a direct access functionality. Allowing users to directly connect to an airbyte API and obtain their own data from it. Further investigation into these functionalities can be performed if Airbyte is selected for these reasons.

# In house solution

The choice for an inhouse solution allows us to create a solution specifically for the current environment of FHICT. Allowing FHICT to obtain data from sources directly instead of using a middle man. Which is what the ETL and datawarehousing solutions are.

The main issues with this are that the addition of new sources will quickly become unmanageable. Beyond that the repeated requests for data, if QS becomes widely used, could block usage from data providers. On a legal basis there is little to no difference in the collection of user data, since we are still processing the data but not saving it.

Beyond these 2 issues. There have little to no instances where projects were maintained by an FHICT delta team. Meaning that the maintenance and security patches will not be provided for future development. Increasing the likelihood of data leaks.

# No ETL solution

It is possible to just use flat files exports from users. Focussing on a core solution where a user will export their data from sources and our systems will to go through the information.

In the end you are than still using an ETL system, which is heavily reliant on users actively providing their data. That said, I believe that this solution is easier to implement than alternatives. Users only have access to their own file directory, which had been cleaned using a pre-processor and is encrypted by the system. Thus following AVG and GDPR privacy laws.

# Usability matrix

The below matrix describes the functionality (on the left) with atop the specific system (header). These can have the following values:

N (for not supported)

Y (for supported)

? (Currently unknown)

\* (look below the table for more info)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Apache Airflow | Apache NiFi | Talend | AirByte | In-house | No solution |
| Collect from web sources | Y | Y | ? | Y | Y | N |
| Collect from database | Y | Y | ? | Y | Y | N |
| Collect from API | Y | Y | ? | Y | Y | N |
| Allow for request of user specific queries (such as canvas API key) | Y | ? | ? | Y | Y | N |
| Write to database | Y | Y | ? | Y | Y | N |
| Allow for transformation operations | Y | Y | ? | Y\* | Y | Y |
| Is actively supported | Y | Y | ? | Y | N | N |
| Is open-source | Y | Y | ? | Y\* | Y | Y |
| Adding data sources can be done with minimal code (only for transformation) | N | Y | ? | Y\* | N | Y |
| Schedule the collection of data | Y | Y | ? | Y | Y | N |
| Logs to see data collection failure | Y | Y | ? | Y | ? | N |
| Has made the proper security steps to prevent hacking attacks | ? | ? | ? | Y | Y | Y |
| Proper documentation for future teams | Y | Y | ? | Y | N/A | Y |
| Allow for new data sources | Y | Y | ? | Y | Y | Y |

\*AirByte tranformations.  
Transformations in Airbyte are done after loading. Since it is an ELT system instead of ETL.

\*Airbyte open-source  
The whole project is open-source but a different license applied for several back-end functionalities. These do not influence QS project.

# Conclusion

# Sources

By Remyrosenbaum - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=68935520>

<https://airflow.apache.org/use-cases/big-fish-games/>

<https://airflow.apache.org/use-cases/adyen/>

<https://airflow.apache.org/use-cases/adobe/>

<https://airflow.apache.org/docs/>

<https://airflow.apache.org/docs/apache-airflow/stable/tutorial.html>

<https://en.wikipedia.org/wiki/Apache_NiFi>

<https://nifi.apache.org/docs/nifi-docs/html/user-guide.html#User_Interface>

<https://nifi.apache.org/powered-by-nifi.html>

<https://airbyte.com/license-elv2>

<https://airbyte.com/embed-airbyte-connectors-with-api>