# How did we build a Data Warehouse in six months?

Jérémy Wimsingues Follow
Feb 26, 2019 · 10 min read

## Abstract

At Everoad, we leverage data to revolutionise the truck industry. For that purpose and to keep pushing our growth forward, we set ourselves the following **ambitious challenges** on the data side for 2018:

1. **Collect** the data from **every** single **source** we have in our possession to have a 360° overview of the company (CRM, emails, calls, …)

2. Provide to anyone in the company with **the right information** at **different level of complexity and aggregation**
3. Have one single way to get the data: **one and only source of truth**
4. Foster data analysts' autonomy so they can **extract their own** data and be focused on what they do best: providing insights and helping the whole company to **perform**
5. Build a healthy and forward looking structure that could enable **future data scientists** to use it

As an introduction, if you have limited knowledge on our company, we are a **marketplace** that enables **shippers** and **carriers** to match in a smarter way to ship goods by truck. Our product is both **a platform** and **a homemade back office** to support the operation department that focuses on monitoring and handling the whole process from the chartering part to the follow up part and billing part. With such an overview of the company, you can easily get the basic data we believed our user would need:

- Basic BI on platform activity to better **understand supply/demand flows**
- **Monitoring** the **operational complexity** to have it more automated and streamlined
- Product related data analysis to **drive product growth**

All those issues represented a pretty huge workload. How did we tackle that? The answer is in this article.

We will first talk about the setup we had and **why** we wanted to migrate, then we will quickly describe the **human resources** we got to do so. Then we will mentioned the **whole infrastructure** we use. Finally, we will explain the **global setup** in place and discuss about how we want to **improve it** and what can be the next step and/or possible evolutions.

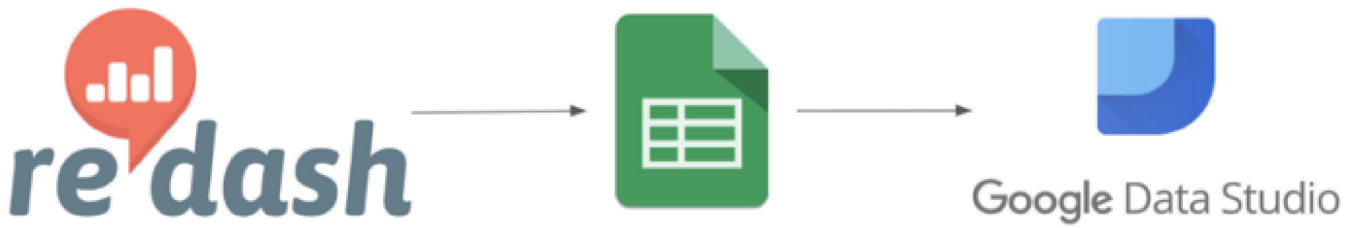
## The legacy of our data: why we needed to migrate

In May 2017, the company already knew that data would be key. Therefore they decided to start using the data better and in a much **more scalable** and forward looking way.

At this point we used **Redash** to extract the data from the BI noSQL database (a daily replication of the production MongoDB database) and import it into Gsheets by using some csv. Then, from these **Gsheets** (related between each over to create an easy-to-go data-pipeline), we plugged **Google Data Studio** to provide some basics dashboards. This was a really handiwork setup, but until then it correctly did the job.

What happened? Gsheets is quite nice and this was a first step. However, when you start to have a “decent” amount of data, Gsheet becomes too slow. Even if Gsheet seems to do the job or if you try to filter/aggregate a lot your data before importing it during the ETL process then you just lose a lot of information and if you still can do so, your **Google Data Studio** dashboards will just become slower and slower, trust us. Obviously in the end, Gsheets wasn’t built for that.

So we decided to migrate it. We already had **Redash**, querying a **dedicated BI noSQL database**, and we already had **Google Data Studio**. We wanted to capitalised on these in order not to lose too much time.



ETL legacy pipeline — May 2018

## Human resources

Two persons were tasked with building the whole data warehouse: **a data analysts** who handled the product-design, Alexandre Laloo, and **a data engineer**, myself, who was responsible of **all the technical support** required: building the BI cluster, building the **ETL** pipeline and all the aspects we will describe in the following developments.

These two resources were **100% dedicated** to the ETL (Extract Transform Load) pipeline and **to build the data warehouse** of the company. In order to do so we needed a strategy. Note that as our team was the first consumer of the data warehouse, we had a good idea of what to build. This point definitely helped us to design and build something **consistent** and **exhaustive**.

Our overall goal was to do it as quickly as possible. A firm aim was to migrate all relevant data we already had inside **Gsheet** to **BigQuery**.

## Global Infrastructure

Regarding the infrastructure, we did not want to take any risk. We chose the top open sources products to help us.

- **Airflow** for the whole ETL infrastructure, using **Composer** inside GCP
- **BigQuery** for the data lake infrastructure
- **Gitlab** for the versioning part and the CI/CD

We have one git repository with the **Python ETL code** where each modification (merge into master post code review) is copied to the bucket used for **Airflow** (DAG folder). And we have a second repository with **all the SQL queries** where each modification is updating **Airflow** variables to modify the ETL. The latter uses the **Airflow CLI** to update the appropriate variable. Both repositories are stored on the private gitlab instance of the company and use *gitlab-ci* to run the post script after being merged on master.

We would be able to elaborate much on this part therefore, do not hesitate if you have any question in this regard. Basically, **Airflow** is running **Python** scripts **every hour** to collect the data from our different sources, **BigQuery** is where we store the results.

In the end, most of the “code” part is in **SQL**, transforming the data from a table to another. Python code is only there to provide some **API calls** and **trigger** data transformation. This required **high literacy in SQL** to transform mongo events to activity based and easily understandable tables, but it is perfectly doable even for a non-tech profile.

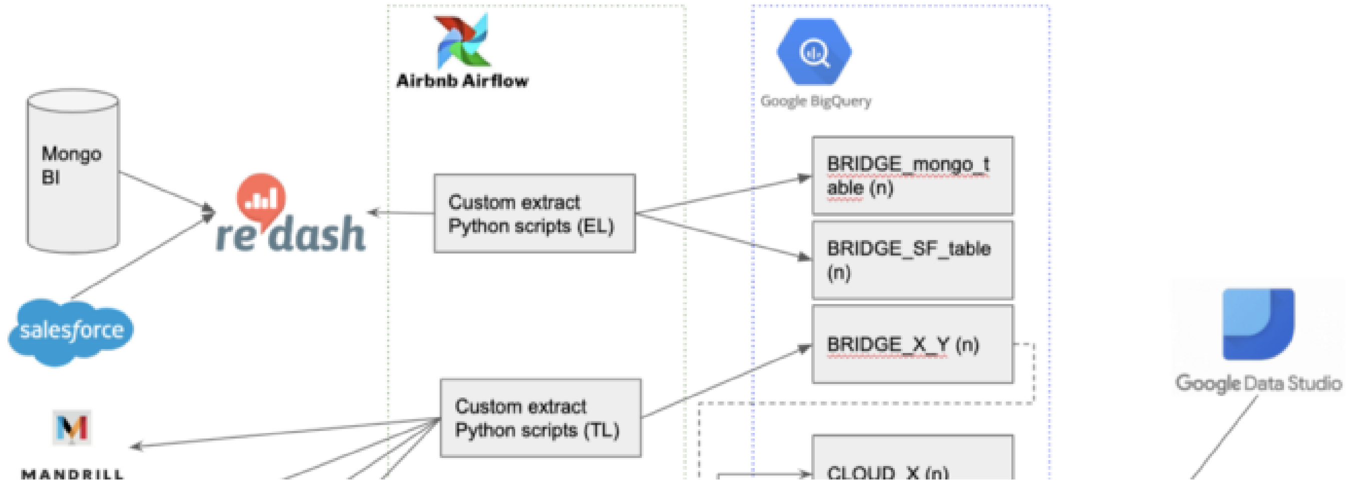
## Data warehouse conception

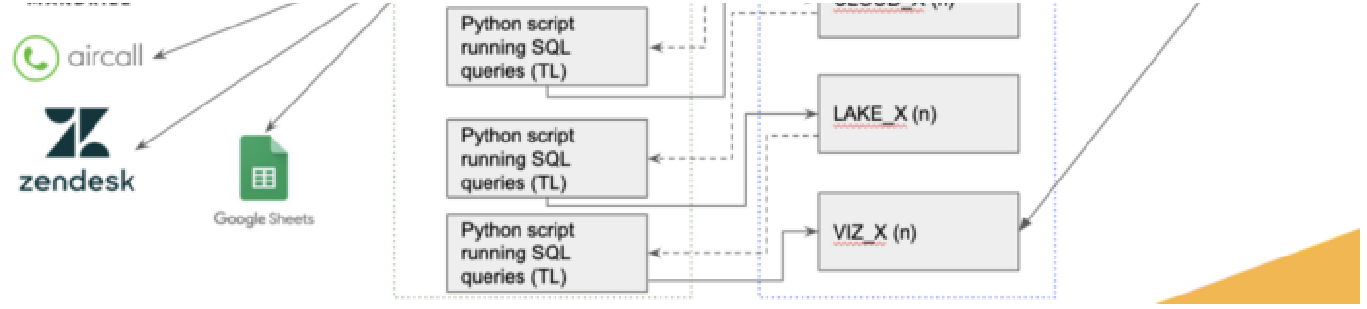
As we wanted to act fast, we spent a lot of time designing the best systems to suit our needs. Here is where we ended up:

1. First, we should use a **BRIDGE** level.
   **BRIDGES** are tables which are the result of the data collection (E from ETL) part. The data in these tables is messy, not that well organised and results from a unique data source.
2. Second, we find a **CLOUD** level.
   **CLOUDS** are tables which are cleaning the data and where the complex data aggregations and processing are being performed. For instance we remove useless information, we aggregate the "event" data to another level of aggregation (if you have event level information, we transform this to offer level information or company level information etc.).
   **CLOUDS** are just a cleaner level of data. These tables are open to our business analysts. Yet, a lot of data levels are still needed for the analysis.

3. Third, we have **LAKES**.
   **LAKES** are tables aggregating the data from different **CLOUDS** to provide all the information from a specific angle. For instance, we can find a lake dedicated to the operation team, a lake for the finance team, a lake for the incident part, etc. This data is open to **every user** of the company and is close to the concept of data lake we can observe pretty much everywhere today.
4. Then, we use **VIZ** objects.
   **VIZ** are filtered data from one or several **LAKEs** and are dedicated to plug the data visualisation part. To make it simple, each dashboard inside **Google Data Studio** is plugged to a **VIZ** table.

In the end, we ended up with the following design:




aircall
zendesk
Google Sheets
Python script running SQL queries (TL)
Python script running SQL queries (TL)
Python script running SQL queries (TL)
LAKE_X (n)
VIZ_X (n)
Global schema infrastructure — December 2018

## What to do next?

We worked/created a lot and we could have said "our job is done, let's move on to another project". But obviously, this is the exact opposite from that: monitoring a data warehouse is **a continuous job**, we probably only did 5% of the whole journey. As any product, it is by definition never over and it needs to be **continuously improved**.

But here is the list of aspects to be improved — if you have not started yet your Data Warehouse project within your company — to help you building it with even more efficiency:

1. **Remove useless components.** As we wanted to migrate the data as quickly as possible, we did not focus on removing **Redash**. As for now, this is kind of a pain for us as we need to maintain it, etc. So the next

level is to use python scripts to directly collect the data from Salesforce and the MongoDB dedicated database to populate **BRIDGES**.

2. **Improve data consistency.** For now, we collect the data and we push it directly into **BRIDGES**. The goal is to clean the data before it is loaded. Meaning be able to detect strings in a better way for instance. Even if we put a lot of efforts in reducing the data consistency problem, as we still use a lot of download/upload of csv files, the data is not fully consistent. Sometimes, for instance, fields change from a type to another, which is not something you like to see (european zip code is a great example for that, phone numbers as well). Luckily we can trick the stuff with SQL casts but this is not the best way to do so.
3. **Improve performances of the data lake.** Implementing a strategy quickly is often linked with lack of performances. As we are growing day after day, week after week, we keep on collecting each time more data. So we need to be aware of performances issues we might face in an uncertain but close future. We have to make some partition inside **BigQuery** (clustering fields), improve some queries, add custom indexes inside MongoDB, etc. This is very important if you know you are about to scale up the process and you must be ready for it.
4. **Do “real time” data.** For now, we have a dedicated BI database which is a dump/restore from the production one. This is done once every hour.

If manageable at the moment, when we scale up, we will need to do some “diff import” instead of dump/restore. The best solution would be to be directly linked with the production (I can hear from my desk all the engineers say “What the hell? BI directly connected to the production?? Are you crazy dude! You can impact the production performances!!”. Wait.). The production at Everoad is using an event sourcing approach, which means that we do have a message broker dispatching the messages to micro services. The goal would be to create a new micro service, listening to all the queues & messages and to push all of them inside the data warehouse. Then we would use this kind of “second event store” to start our transformation pipeline, meaning that we would have almost “real time” data.

5. **Improve the data access**. For now, people have an access to BQ and templated queries to query the data warehouse. The next goal would be to provide a means to create easily data visualisation without our intervention. A tool like Superset (again from Airbnb) would be a nice instrument to have for instance.
6. **Improve the data visualisation part**. When you want to set up some dashboards, Google Data Studio is an asset, pretty simple, straightforward and easy to use. As we grow, again, depending on the evolution of this tool (new features, etc.) we might change for another,

using for instance *Tableau* or another software if appropriate (easier filtering system, ability to download the dataset you are seeing, etc.). For now, this would be a huge gap as every single dashboard is on Google Data Studio.

## Conclusion

We had two dedicated persons for the project, we gained **a lot of experience**. We already knew the company quite well (which does help to understand operational metric you have to build etc.), we had a clear objective and we spent some time to design something which would be cost efficient.

Using open source technologies will make you save decades. Basically, the **Python** code is not that hard, **Airflow** has a lot of connectors making it easier to start with and **BigQuery** is an incumbent of the market and you can honestly go blind and use it.

Note that we had already done an important job, which is not mentioned in this article: the needs collection. **We already knew what our users wanted**. We knew which data to collect, which metrics to build, relevant dashboards to create etc. This is a really important starting point: you have to know what your clients want and need **NOW** and also know what they are going to ask **tomorrow**. This is really key as it is the main difference between “doing an extract” or “doing an ad hoc analysis” and “building a data warehouse”. By building a data warehouse you want to answer the present needs but also anticipate future needs. Meaning that you build something which can be easily changed, maintained, etc.

Let’s conclude with the figures part:

- 70+ **BRIDGES** created
- 30+ **CLOUDS** created
- 20+ **LAKES** created
- 40+ **VIZ** created
- 160+ SQL queries
- Around **2000 lines** of python code
- 350+ **variables** inside Airflow
- Our main pipeline takes around **30 minutes** to process
- **20+ millions rows**