Building a Modern Data Platform for Analysing SaaS Startups

Son Nam Nguyen

June 1, 2022

This is my capstone project's summary submitted at the end of the MS Business Analytics program.

1 Project Goal

The client's request was to build a highly scalable standardized platform, which consolidates a cohort of SaaS startups in a single pipeline and streamlines the automation of KPI generation. The driving incentive behind it was to be able to see the performance of a particular cohort, the return on their investments, as well as to develop a product that can be deployed to other companies interested. This should open possible sales opportunities and promote analytics in the financial sector, given the time it takes to extract value from the data in the modern era.

2 Initial Challenges

2.1 Technical

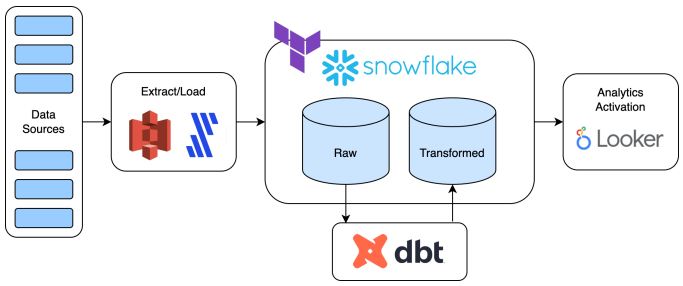
The project I joined had already built out the foundational tools and concepts used currently. The exception was that the tools used were not efficient and cost-friendly by any means. The ingestion process was done via (unnecessarily) repeated micro-batch processing rather than on-demand, and the data integration tool used for loading structured and semi-structured files from our proprietary data lake cost thousands of dollars per month for our client. A cost that can be pushed down to zero very easily. At the time when we onboarded ourselves, a substantial amount of project time was spent on making the existing pipeline work as it is. There were no tests (nor integration), no concept of development or deployment process.

2.2 Circumstantial

Since our client is working in a highly competitive area, the delivery had to be punctual, and often we had tight schedules for shipping a feature. This could result in accumulating technical debts by releasing suboptimal development work. I have personally experienced that between them (analysts) and us (engineers) there is still a notable gap in understanding how much time it takes to deliver a certain feature. The fact that we are not working in the same time zones and there are constant changes in the ongoing sprint scope, as well as ad-hoc requests taking priority, describes how challenging it is to ship data as a feature.

3 Implementation

Our approach was to level the playing field between data analysts and engineers. Luckily, the emergence of analytics engineering and the renaissance of SQL gave rise to easy-to-use building tools. Most of the great practices in the profession are inherited from software engineering, which was never used in data. That is how we test and deploy features, and how we collaborate and document new releases. All the great evolutions of cloud data warehousing made in the last decade made it obvious that the entry fee is substantially low to shift companies to a data-centric direction because of how it uses its resources to separate compute from storage and prevent customers from paying hefty license fees even when the engine is idle. The flowchart below best describes the core pipeline as of today, supercharging the analytics for analysis.



3.1 Data Ingestion

Our data ingestion was completely refactored when I joined the project. We found out that there is a direct integration between our data lake and cloud data warehouse, which makes the integration tool mentioned before redundant. Also, the streaming services used for doing table-to-table refreshes in the warehouse were deemed to be in contrast with the OLAP system we wanted to develop. Therefore, we used open-source, free services to onboard data from startups. At the end of the implementation, we compared billing costs, and it turned out that we brought down the cost of ingestion from 15,000 USD per month to 0 USD per month.

3.2 Data Processing

We processed our data using the Kimball dimensional modeling. The transformation tool we used allowed us to build business logic in a modular fashion. In other words, we had layers in our pipeline (base and staging) specifically allocated for rationalizing the various ways we can receive submissions from SaaS companies. That means we had to map columns together and align calculation values before pulling them together to a join schema definition. Our core integration layer transformed our tables into an aggregated form with all the necessary KPIs our client wanted to see on the other side. The KPIs we focused on were revolving around recurring revenue, customer retention, time to conversion, and sales funnel.

3.3 Data Quality

As I mentioned before, there wasn't a testing framework around the pipeline. Since we joined, we set up more than 500 testing suites, constantly checking descriptives and references in our models. This was we ensure that before each feature delivery there are no discrepancies in the data. Furthermore, we made it possible to continuously integrate and deploy in an isolated environment after each pull request submission. One more thing I wanted to mention is how we deploy new changes. We this particular process blue-green deployment, simply because when we deploy to the production database, we load all the data into a staging environment instead. When the data is loaded and tested, we switch over the traffic by swapping out the two databases. This way, we guarantee that there is no downtime during a release and that bad data is not provisioned to production.

4 Results and Discussion

We have identified a lot of strengths as well as weaknesses of an arbitarily chosen Company X (which remains nameless because of confidentiality). For instance, we saw from the change in quarterly net bookings (+160%) and average seats purchased (+50%) that there was a steady growth in both. Not only did they onboard more customers, but also the average contract value of a subscription also increased. The latter is verified by a 45% increase in the compound annual growth rate of the average contract value. One reason why customers are buying larger contracts can be attributed to the new packages Company X has introduced, which are more extensive compared to what was on the counter before. We have seen metrics trailing the competition in our customer retention analysis, where we noticed that the majority of customers are only active for four quarters after their first purchase. In

the recent quarters after the pandemic, we have also seen that active user growth has trended lower than usual. Finally, we wanted to see how hard is it to pull customers through the sales funnel, and we concluded that from the total number of seats purchased, only around 7% of the prospects are integrated and only 0.4% of the total is active every week.

5 Future Challenges

We pinpointed two major concerns in how the product will evolve in the future. One of them originates from the fact that at the end of the day, we want to onboard as much startup data as we can on a single platform. One can easily see the exponential growth in terms of how much computing and storage we need. Latency is what keeps our users tied to our platform, but in the meantime, we tackled optimization steps such as selective processing, sampling to development, and handling rows that do not have to be aggregated (e.g. missing values). Our goal is to simplify our onboarding process because a large chunk of the deployment is taken by figuring out the proper mapping and implementing temporary data imputation due to source issues.

The other one is observability and transparency. There is a lack of centralized documentation, especially in complex business transformations. There is a lot of friction when our client asks for a definition of a column when he/she could easily browse that from a data catalog. What's more, even ourselves, developers find it hard to keep track of schema changes imposed in the backend when we expose new dimensions to the users in the visualization layer. The introduction of a data catalog would help immensely in keeping everything organized, even on a larger scale. Besides, a data monitoring tool can also be useful to catch silent data bugs not introduced by code.

6 Resolution

I have already mentioned a couple of ways we are working on to resolve the situation, but none of them affects the user experience, rather the developer experience. To improve the former, we were thinking of building another cloud data warehouse focusing on "last mile queries" as an acceleration layer on top of our ELT stack. This would benefit from the rigorous indexes, partitioned storage format, and materialized views stored in memory. We already did a POC on mirroring dashboards between the two warehouses, and query times were three times faster on average. We somehow still need to automate the process of deploying in production in our ELT stack and syncing newly added calculations and columns to the acceleration layer. We expect to overhaul this issue with the help of an orchestrator tool that will be finely integrated with our project.

7 Recommendations

The way how we imagined the future of the product is we supply a spreadsheet for potential client companies who configure the necessary mapping for us. After that, the data goes through a profiler where we perform sanity checks on the data and flag data issues with actionable feedback to the source. Then on, it would be processed in our ELT warehouse driven by a modular framework and loaded into the acceleration layer, energizing user queries in the dashboarding layer. For each new company onboarded, we generate the dashboards based on the template to save resources on building the same charts over and over again. All these steps are going to be documented end-to-end in our future data catalog, while the backend will monitor the tables for any anomalies deviating from the predicted descriptives based on past data.