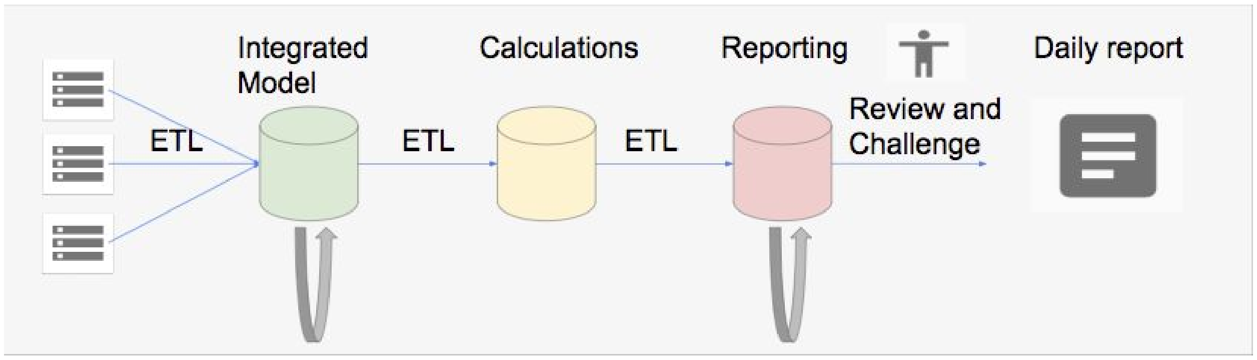
Case Study 01

I worked with a client that had a very complex reporting pipeline. They had a very large amount of data that had to be processed into a report that was going to regulators on a daily basis. They had to demonstrate that the risk in the financial data from the previous day indicated that they were following the regulatory rules. They put the data through multiple systems or stages. Each stage had a separate Extract-Transform-Load sequence and then performed unique processing on the data.

A customer had this interesting business requirement...

- A daily reporting pipeline with multiple sources and complex dependencies
- Human intervention to data check quality, inputs, and proceed to next stage
- Need daily updates on yesterday's data, but takes >24 hours to run



The complexity in the simplified diagram makes it look like it was a linear progression from start-to-finish. But this is meant to be symbolic. The actual processes were much more complicated and it took 30 hours from start to finish to generate one report. The processes were actually more of a spider-web, with many dependencies. So one part would run and then halt, waiting until other dependent parts were complete before proceeding. And there were some processes that could run in parallel.

We mapped that to technical requirements like this…

We first diagrammed out all the processes. And then we started to look at how to implement this on Google Cloud using the available services. We initially considered using Dataproc or Dataflow. However, the customer already had analysts that were familiar with BigQuery and SQL. So if we developed in BigQuery it was going to make the solution more maintainable and usable to the group. If we developed in Dataproc, for example, they would have had to rely on another team that had Spark programmers. So this is an example where the technical solution was influenced by the business context.

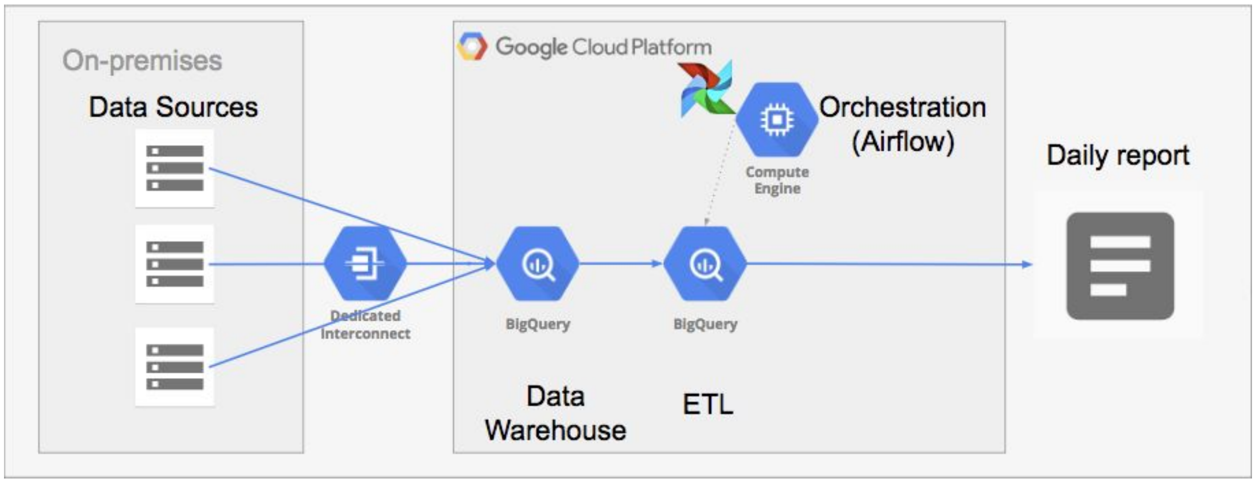
To make this solution work, we needed some automation. And for that we chose Apache Airflow. In the original design we ran Airflow on a Compute Engine instance. You might be familiar with the Google service called Cloud Composer, which provides a managed Apache Airflow service. Cloud Composer was not yet available when we began the design.

BigQuery and Cloud Composer (aka Apache Airflow)

- **BigQuery:** Reduce overall time to run with BQ as data warehouse and analytics engine
- **Apache Airflow:** Control to automate pipeline, handle dependencies as code, start query when preceding queries were done

And this is how we implemented that technical requirement.

- Common data warehouse in BigQuery. Apache Airflow to automate query dependencies



In this particular case we used open source Apache Airflow. But if we were implementing it today we would use Cloud Composer.

Cloud Composer / Apache Airflow allowed us to establish the dependencies between different queries that existed in the original reporting process. BigQuery served as both the data storage solution and the data processing / query solution.

We were able to implement all their processing as SQL queries in BigQuery. And we were able to implement all the dependencies through Airflow.

One of the time-sinks in their original process had to do with that 30 hour start-to-finish window. What they would do is start processing jobs and sometimes they would fail because the data from a previous dependency wasn't yet available. And they had a manual process for restarting those jobs. We were able to automate away that toil and the re-work by implementing the logic in Apache Airflow.