HENTSŪ
SYSTEMATIC AGILITY

CASE STUDY

Azure Data Factory

REQUIREMENTS

- Process 11,000 files & total compressed size of ~2TB
- Ingested into a database
- Keep raw files
- Parallel and rate controlled
- Account for every file
- Ongoing low effort maintenance, cost-efficient and automated

The Challenge

A client recently approached us with a data science challenge regarding one of their data sets. The data was provided to the client in an AWS environment in a Redshift data warehouse. While this was fast they found it to be very expensive, in AWS the data and compute costs are coupled together. As such, a large data set necessitates a high spend on computing costs, even if this level of speed is not necessary for their analysts.

However, the data was also available in CSV format in an S3 storage bucket, which could be the starting point of a new approach. The client already had all their infrastructure deployed and managed by Hentsū in Azure, so they wanted to consolidate into the existing infrastructure.

After reviewing the challenges, we were able to create an elegant solution leveraging the huge power and scale of the cloud, which is simply not possible in traditional infrastructure.

Key Considerations

The solution had to be able to process this large data set consisting of over 11,000 files and a total compressed size of ~2TB, with additional files every day.

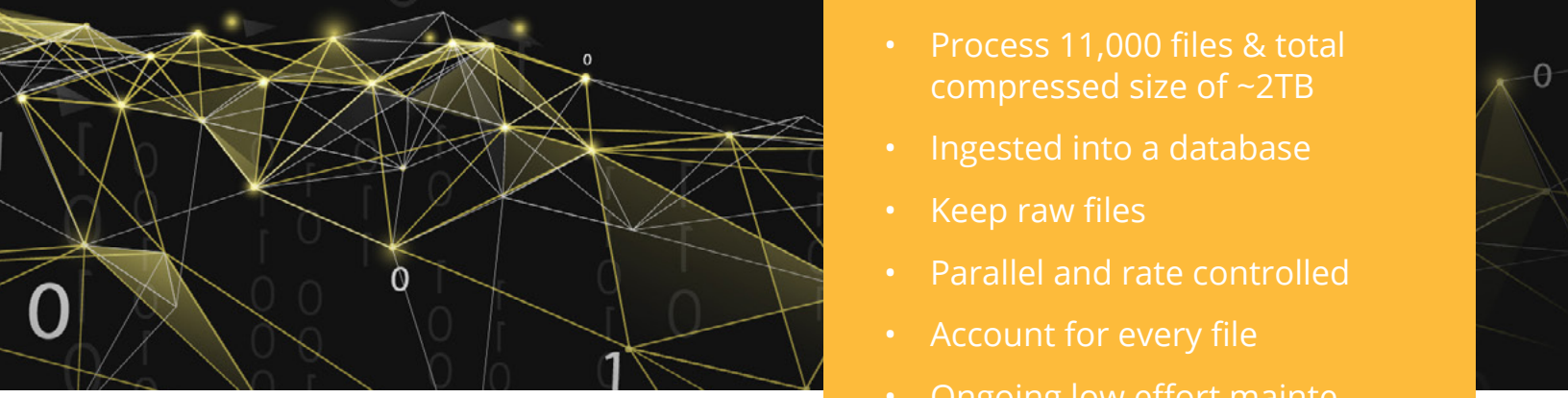
Raw files had to be stored for any future needs, whilst also being ingested into a database.

The ingestion should both be parallelisable and rate controlled, to ensure we manage the number of database connections and have orderly ingestion.

Not only was this to be a one-time load of historical data, but new files created needed downloading and ingesting in an automated fashion.

Every file had to be accounted for to ensure that all the data is moved correctly, so keeping track of each file's status was important. Things happen; connections break, processes stop working, so we must have a system in place when these do occur.

Keep ongoing maintenance low effort, cost-efficient and automated, and delegate as much of the maintenance away from end-users.



TECHNOLOGIES USED

- Process 11,000 files & total compressed size of ~2TB
- Ingested into a database
- Keep raw files
- Parallel and rate controlled
- Account for every file
- Ongoing low effort maintenance, cost-efficient and automated

The Solution

Hentsū recommended a solution built on Azure Data Factory (ADF), Microsoft's Extract-Transform-Load (ETL) solution for Azure. While there are many ETL solutions that can run on any infrastructure, this is very much a native Azure service and easily ties into the other services Microsoft offers.

The key functionality is the ability to define the pipelines to move the data in a web user interface, set the schedules which can either be event based (such as a creation of a new file) or on a time schedule, and then Azure handles the execution of the pipelines to process the data. The pipeline creation requires relatively little coding experience so it makes it easy to delegate this to staff with little technical experience

Technical Details

Hentsū built out the data pipelines to move the data from AWS into Azure. The initial load was triggered manually, but then the update schedules were set to check for new files at regular intervals.

Hentsū created status tables to keep track of each file. This allows us to keep track of the state of the data as it passes through the pipelines and use a decoupled structure so that any troubleshooting or manual intervention can happen at any stage of the process without creating dependencies. The decoupled structure meant that individual files and steps can be fixed in isolation, and then the rest of the pipelines and steps continue uninterrupted. The clean decoupling means any errors on a particular step were easily identified and notified to users for investigation.

All the data was then mapped back to these tables, to be used if we ever needed to do further processing or cleaning on the final tables. The data was further transformed with additional schema changes to match the client's end use and to map it to the traditional trading data.

The pipelines were deliberately abstracted to allow for the least amount of work to add new data sources in the future. The goal was to make it easy for the client's end users to do themselves as and when required.

The Benefits of Azure Data Factory

ADF can run completely within Azure as a native serverless solution. This means there is no need to worry about where the pipelines are run, what instance types to choose upfront, manage any servers/operating systems, configure networking, and so on. The definitions and schedules are simply set up and then the execution is handled.

Running as a serverless solution means true “utility computing”, which is the entire premise of cloud platforms such as Azure, AWS, and Google. The client only pays for what is used, there are no times with idle servers costing money without producing anything, and it can scale up as needed.

ADF also allows the use of parallelism while keeping your costs to only what is used. This scaling up was a huge benefit of ADF for the client and when time is of the essence; one server for 100 hours or 100 servers for one hour cost the same, but the work is done in 1/100th of the time. Hentsū tuned the solution so the speed of the initial load was only restricted by the power of the database, allowing the client to balance the trade-off between speed and cost.

ADF has some programming functionality, such as loops, waits, and parameters for the whole pipeline. Although there is not as much flexibility as a full language (Python for example) it allowed Hentsū significant flexibility to design the workflows.

Caveats

There are limited sources and sinks (i.e. inputs and outputs). The full list is available in the Microsoft documentation. Microsoft's goal with ADF is to get data into Azure products, so if one needs to move data into another cloud provider a different solution is needed.

The pipelines are written in their own proprietary "language", which means the pipelines code does not integrate well with anything else, which would not be the case if they were written in a language like Python, as many other ETL tools will provide. This is also the key reason we have developed our own ETL platform for more complex solutions which uses Docker and more portable Python code.

There were some usability issues when creating the pipelines, with confusing UI or vague errors on occasion; however, these were not showstoppers. Our advice when using the ADF UI is to make small changes and save often. We can see that Microsoft is already aggressively addressing some of the issues we encountered.

Impact

The client was very pleased with the ADF and Azure SQL Data Warehouse solution. The solution automatically scales the compute power to process the data as it changes week by week, it scales up when there is more data, and scales down with less data. Overall the solution costs a fraction of what it did previously whilst keeping it all within the client's Azure environment.

HENTSŪ LONDON

30 Crown Place
London, EC2A 4EB
United Kingdom
+44 203 857 1630

HENTSŪ NEW YORK

600 Fifth Avenue
New York, NY 10020
United States
+1 315 257 4284

HENTSŪ BOSTON

125 High Street
Boston, MA 02110
United States
+1 315 257 4284