International Journal of Computer Applications Technology and Research
Volume 11–Issue 08, 315-319, 2022, ISSN:-2319–8656
DOI:10.7753/IJCATR1108.1004

Developing an ETL Pipeline for Data Analysis

A S Prajwal Babu
Computer Science and Engineering
RV College of Engineering
Bangalore, India

Prof. Suma B
Computer Science and Engineering
RV College of Engineering
Bangalore, India

Abstract:The world's most valuable resource these days is the expanding data. Large organisations continuously produce data about their clients, consumers, and employees in real time. This data cannot be easily interpreted in its raw form, but after being processed and changed, it can be widely used for analytics. This improves a number of the aforementioned business entity's existential traits, including organisational management, market capabilities, and consumer feedback.Given the volume of data that a corporation generates, it is obvious that it will need a significant investment of money, time, talent, and resources to achieve the goal of in-house data processing, calibration, and storage. The goal is to overcome the obstacles businesses present for data-pipelining technology and get processed data directly at the conclusion of the data sync cycle. One sync cycle is the continuous fetching of data created or altered over the course of a given time frame, such as a fortnight or a month.

Keywords: Data pipeline,ETL pipeline Cloud, Data Warehouse, Data Analytics

1. INTRODUCTION

An ETL pipeline is a group of procedures used to transfer data from one or more sources into a database, such as a data warehouse. The three interdependent data integration processes called "extract, transform, and load," or ETL, are used to take data out of one database and transport it to another. Once loaded, data can be used for reporting, analysis, and the creation of useful business insights.

The relevance of utilising such data in analytics, data science, and machine learning programmes to gain business insights develops along with the amount of data, data sources, and data types at organisations. Since turning the raw, unclean data into clean, new, trustworthy data is a crucial step before these projects can be undertaken, the requirement to prioritise these activities puts growing pressure on the data engineering teams. ETL, or extract, transform, and load, is a method used by data engineers to gather data from various sources, transform it into a reliable and useable resource, and then load it into the systems that end users may access and utilise later to address business-related issues.

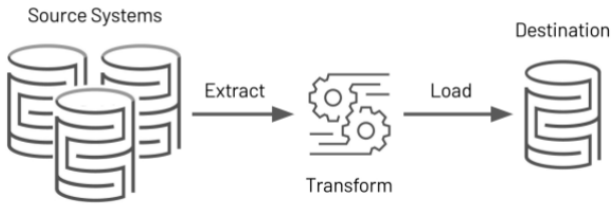

Fig 1: ETL process

Extract

Data extraction from the target sources—which are typically heterogeneous and include business systems, APIs, sensor data, marketing tools, transaction databases, and others—is the initial step of this process. As you can see, while some of these data types are likely to be semi-structured JSON server logs, others are likely the structured outputs of commonly used systems. The extraction can be done in a variety of ways: Three techniques for data extraction:

Partial Extraction - If the primary system alerts you when any data has changed, that is the simplest way to retrieve the data.

With Update Notification of Partial Extraction - Not all systems can send out notifications when an update occurs, but they can still identify the entries that have changed and send out an extract of those records.

Full extract - Some systems are unable to determine which data has been altered at all. In this situation, the only way to obtain the data from the system is through a full extract. For this technique to work, you must have a duplicate of the previous extract in the same format so you can track down the modifications that were performed.

Transform

This stage entails converting the unformatted raw data that has been gleaned from a source into a form that can be accessed by various applications. In order to meet operational requirements, data is cleaned, mapped, and converted during this stage, frequently to a particular schema. This procedure involves many sorts of transformation to guarantee the accuracy and reliability of the data. Instead of loading data straight into the ultimate data source, data is usually placed into a staging database. This procedure guarantees a speedy rollback in the event that things does not proceed as expected. You have the option to create audit reports for legal compliance at this point, as well as identify and fix any data problems.

Load

Last but not least, the load function involves copying converted data from a staging region to a target database, which may or may not have existed before. The complexity of this process will vary depending on the requirements of the application. You can use ETL tools or custom code to complete each of these processes.

2. RELATED WORK

The idea of data pipelines is relatively new, and recent innovations in cloud architecture and cloud storage have advanced this particular field. These are the only new developments in the related area of data pipelining.

The following idea served as the foundation for a study on ETL technology that was carried out in 2009 [1]. Extraction-Transformation-Loading (ETL) forms are the earliest computer algorithms that promote initial stacking and sporadic warehouse refreshing. There were some limitations to this; information extraction is still a challenge, largely due to the closed nature of the sources; there are also challenges with streamlining and

resume; and the absence of a baseline prevents further research.
Real-time ETL Data Warehousing was then researched in 2012 [2]. The goal was to achieve real-time data warehousing, which is heavily reliant on the selection of an extraction, transformation, and loading (ETL) method in data warehousing technologies (ETL).
In 2013, synchronous research [3] was being conducted in the field of ELT using an information distribution center's ability to directly input unprocessed, raw data while deferring information update and cleaning until needed by pending reports.
ETL was being accepted for a few applications later in 2016 [4, including the healthcare field]. While maintaining its integrity, this information must be appropriately deleted, modified, and packed into the warehouse. It gave the extract, transform, and load (ETL) procedure its seal of approval for correctness, populating the clinical research database as a result.
At the same time, Amazon's S3 [5] service may be useful because it offers bulk storage that doesn't require packing or cleaning. The Simple Storage Service (S3), a cheap capacity utility, had been introduced by Amazon.com. S3 intends to offer storage as a low-effort, widely available assistance with a simple "pay more only as costs emerge" payment approach.
With the introduction of Data lakes after one year in 2017, the adoption of Extract-Load-Transform grew more quickly [6]. The simplest assumption of an information lake is to mash up each piece of data provided by an organisation to produce increasingly important information at finer granularities.
2018 saw the incorporation of a defined methodology to create an R-based platform leveraging SQL. create a framework for R that influences SQL and is predictable and piping-able such that repeatable research on medium-sized data is a simple reality. Therefore, it had scaling issues based on data volume, and algorithms weren't instantaneous for medium data, which increased latency. Another implementation was made later that year to compile scientific data for analysis. To handle scientific data aggregation, transformation, and improvement for scientific data discovery and retrieval, a distributed extract-transform-load system that is horizontally scalable [8].
The improvement of privacy for ETL operations, particularly with biomedical data, was the subject of research in 2019 [9]. Data from many sources can be combined at clinical and translational distribution centres to create the requisite enormous datasets. This was accepted since anonymization was not supported by current ETL tools. Furthermore, at that moment, basic anonymization tools cannot be incorporated in ETL work processes.
Another work procedure related to the widely used On-Demand ETL system was being studied that same year. The Extract Transform Load process (ETL), which is the primary bottleneck in BI arrangements, is addressed creatively by DOD-ETL [10], an instrument that provides it in almost real-time. The main difficulty was to manage several information sources while also providing little latency for real-time responses.
Use in the banking industry was also being investigated later that year. Our new idea (RDD4OLAP) cubes consumed by Spark SQL or Spark Core fundamentals will replace the standard information combination and investigation process. It will do this by utilising Extract-Transform-Load (ETL) concepts, big data processing techniques, and oriented containers clustering architecture [11]. But also provide for very little delay so that you can react instantly.

3. EXISTING FAME WORK

To create a contemporary ETL system, open source frameworks like Apache Airflow might be employed. There are fantastic possibilities to contribute to the open source community that we pretty much rely on when the project is still in the development stage. As a result, we have chosen to release the project as open source under the Apache license.
Below are some of the procedures that Airflow powers:
Data warehousing: prepare, arrange, evaluate the quality of the data, and add information to our expanding data warehouse.
Calculate metrics for both host and visitor for engagement and growth accounting using growth analytics.
Experimentation: Calculate the logic and aggregates of our A/B testing experimentation framework.
Search: Calculate metrics relating to search ranking.
Email targeting: Applying rules to email targeting allows us to target and engage users.
Sessionization: generate datasets for clickstream and time spent
Data infrastructure maintenance: Application of data retention policies, folder cleanup, and database scraping are all examples of data infrastructure maintenance.
Airflow Principles:

- Scalable
- Dynamic
- Extensive
- Elegant

Airflow Features:

- Pure Python
- Useful UI
- Robust Integrations
- Open Source

Architecture
Python has solidified itself as the language of data, much the way English is being used for professional business. Python-like Python was used from the ground up to create Airflow. The code base has extensive unit test coverage, is expandable, well-documented, consistent, and limited.
Python is also used for pipeline creation, making it simple to generate dynamic pipelines from configuration files or other sources of metadata. We adhere to the idea of "configuration as code" for this. Although any language could be used to construct Airflow pipelines using yaml or Json task setup, we thought that some fluidity was lost in translation. It is quite valuable to be able to meta-program, subclass and use import libraries while writing pipelines in code (Python, IDEs). Remember that as long as you create Python that reads these configurations, you can still author jobs in any language or markup.
Airflow can be used for running in just a few commands, however the full architecture consists of the following elements:

A comprehensive CLI (command line interface) for testing, running, backfilling, describing, and clearing DAG components.
An online tool for exploring the definition, dependencies, status, metadata, and logs of your DAGs. The Flask Python web framework serves as the foundation for the web server, which comes packed with Airflow.
A metadata repository which the Airflow uses to maintain track of tasks and jobs statuses and other permanent data, often a MySQL or Postgres database.
A group of workers that distributes the execution of the task instances for the jobs.
The instances of the tasks that are prepared to run are launched by scheduler processes.

International Journal of Computer Applications Technology and Research
Volume 11–Issue 08, 315-319, 2022, ISSN:-2319–8656
DOI:10.7753/IJCATR1108.1004

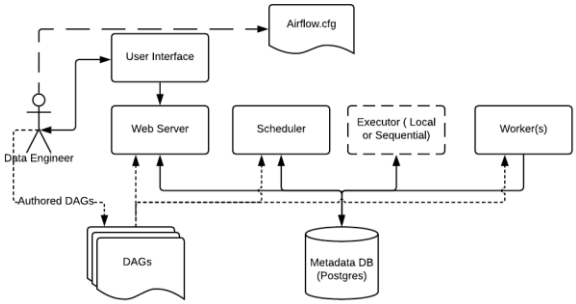

Fig 2: Airflow architecture

There are a few things to consider:
SQL database is used by Airflow to hold metadata about the data pipelines that are being used. This is shown as Postgres in the picture above, which is very popular with Airflow. MySQL is one of the alternative databases that Airflow supports.
- Web Server and Scheduler: The Scheduler and Airflow web server are independent programmes that communicate with the aforementioned database while running locally (in this scenario).
- The Executor is depicted separately above since it is frequently referenced in Airflow and in the documentation, although it actually runs inside the Scheduler and is not a separate process.
- The Worker(s) are independent processes that also communicate with the metadata repository and other elements of the Airflow architecture.
- Airflow.cfg is the configuration file for Airflow, and the Web server, Scheduler, and Workers may all access it.
- DAGs are the Python code-containing DAG files that represent the data pipelines that Airflow will perform. These files must be accessible by the Web Server, Scheduler, and Workers, and their location is specified in the Airflow configuration file.
A DAG defines your process in this manner, but keep in mind that we haven't specified what we actually want to do—A, B, and C could refer to anything. Perhaps A prepares the data that B will use to evaluate it while C emails. It's also possible that A keeps track of your whereabouts so that B can open your garage door and C can turn on your house lights. The DAG's role is to ensure that whatever its constituent activities accomplish occurs at the proper time, in the proper order, or with the proper handling of any unanticipated complications; it is not important what those jobs actually do.

4. PROPOSED FRAMEWORK

The framework which we are discussing in this paper is primarily built using Node JS. The framework is built in such a way that even a person with least programming experience can build an ETL pipeline. Most of the logic which has to be implemented should be done using SQL.

ETL Stack

It is an ETL (Extract/Transform/Load) Stack written in NodeJS. Extract useful data out of raw data, Transform to usable metrics (aggregations) and Load to Enterprise data lake.

ETL Job

An ETL Job (configured as JSON file) is a set of interdependent tasks which run as a single unit of work. It is a logical unit of work - Hourly Viewership metrics, Daily Ad-Analytics metrics.

ETL Task

A Task is a single piece of independent work unit - Compute hourly sessions from Beacon data for example. It can depend on other task(s) to run.

Big data computation

Most tasks work with a source as Data lake, compute on data from lake and put back computed data into data lake. Some cases, they put the data/metrics back to the end-user reporting system.

Tools Used

- Athena

Athena is a partition supported realtime big data crunching system using Facebook's PrestoDB underneath. You can write a SQL query which runs on the data on S3. It can extract, filter, aggregate, group data to create metrics.

- Data lake(S3)

S3 is big data storage system to store objects, logs, records in formats like JSON, Parquet, CSV, Regex parsable text records.

- Postgres

RDBMS database used to store the end-user facing reporting metrics with right indices to fetch data faster. Analytics portal and APIs can use this database to provide reports and visualizations to customers

The framework allows us to create a pipeline which is often referred to as a job. This job will have many interdependent tasks. The tasks are the individual work items which carry out a specific function. All these tasks are joined and interlinked to create a pipeline.

All the jobs in the framework are automated using cron schedules so that without any human intervention all the jobs are running at prescribed time. From extraction of information till loading the useful information into the database is automated. And the data in the database is used to build dashboards, send reports, monitoring and alerting etc.

The picture below Figure 3 shows the basic architecture of the whole process of how the ETL pipeline is working.

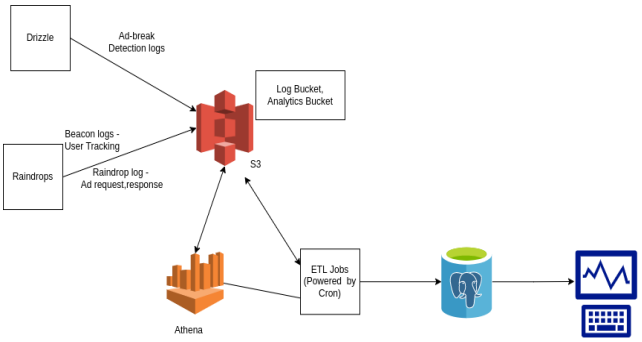

Fig 3: Basic Architecture

The architecture displays how the data is extracted from different data sources or deployments. All this data is stored in Amazon S3, which is the data warehouse. The data which is needed for the pipeline is extracted from the data warehouse and specific transformations are applied to the raw data using Amazon Athena. The transformed data is then loaded into the database. From the database the data is queried and displayed into the dashboard.

International Journal of Computer Applications Technology and Research
Volume 11–Issue 08, 315-319, 2022, ISSN:-2319–8656
DOI:10.7753/IJCATR1108.1004

5. METHODOLOGY

The following discussion touches upon the methodology of how the pipeline is implemented from the first step of extraction to the last step of displaying the data.

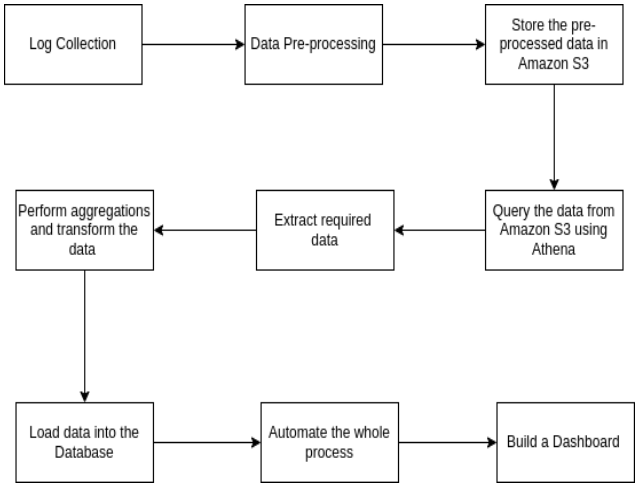

Fig 4: Flow Chart of the methodology

Figure 4 represents the flow of events happening while building a pipeline:

Log Collection

The log data or the raw data is collected from different data sources or deployments and stored in the Data Warehouse.

Data Pre-processing

The raw data or the logs which are collected can be any format, usually the logs will be in a Json format. The data is processed and converted to csv format with the useful data in it. The pre-processed data is again stored in Amazon S3.

Extraction

Using Amazon Athena, few useful pre-processed data is queried and used in the pipeline for getting some insights upon that data.

Perform Aggregation and transformation

The raw data is aggregated and then transformed to get desired output using the query engine which is the Amazon Athena.

Loading

The final transformed data is loaded into the database, and stored over there.

Automation

The whole process is automated using cron schedulers. The process is scheduled at a particular time, depending on the frequency of which the process should be run. Accordingly the process is scheduled and automated.

Dashboard

A dashboard is built upon the data in the postgres to provide business insights to the customers. The dashboard can also be used for monitoring and alerting the development team if there are any data discrepancies.

6. RESULTS AND DISCUSSION

In a perfect world, data analysts have access to all the information they need and don't have to worry about how or where it is stored since analytics just function.

The reality of analytics has been far more convoluted up until recently. Building and maintaining flimsy ETL (Extract, Transform, Load) pipelines that pre-aggregated and filtered data down to a consumable level was required in order to access data because expensive data storage and underpowered data warehouses made this impossible. ETL software providers compete based on how specialised and adaptable their data pipelines were.

We are now getting closer to the analysts' ideal thanks to technology. Fragile ETL pipelines are a thing of the past thanks to practically free cloud data storage and significantly more powerful contemporary columnar cloud data warehouses. Extract and load the raw data into the destination, then transform it after the load, is the modern data architecture. Numerous advantages result from this distinction, including improved adaptability and usability.

7. CHALLENGES

The challenges involved on this framework are:

- Retries for Failures: NO Retries for failed jobs. Manual retries should be done for failed jobs
- No Cross-job Dependency: A task can depend on other task(s) in the same job, but can't depend on a task of another job.
- Non-parsable Logs: Logs collected from ETL Jobs are not parsable. So debugging requires manual analysis of Logs.
- Too many Slack notifications: Slack notifications are too many - causing more ignorance for failed jobs.

8. CONCLUSION

Technology trends are aware that processing, storage, and bandwidth are now accessible to anyone. The price of computation has decreased over time due to technological advancements. Similarly, the price of a gigabyte has dropped from around $1 million to a few cents in a period of about 35 years. Data warehouses can now hold significantly higher data volume as a result of these drastic cost reductions. It is no longer necessary for organisations to pre-aggregate and, in the process, delete a significant amount of source data. This makes it possible for analysts to do analysis that is both deeper and more thorough than previously. Despite the World Wide Web not existing until 1991, internet transport costs have drastically fallen. It fell from approximately $1,200 per Mpbs to a few cents in less than twenty years. The cloud, or the utilisation of remote, decentralised, web-enabled computer resources, is the result of the convergence of these three cost-reduction developments. A wide variety of cloud-native applications and services have also emerged as a result of cloud technology.

Many firms adopt a manual, ad hoc approach to data integration; in fact, 62 percent utilise spreadsheets like Google Sheets and Excel to combine data from several files and visualise the information. 2 In order to do this, files must be downloaded, values must be manually changed or cleaned, intermediate files must be created, and other similar operations.

Ad hoc data integration has a number of disadvantages, to mention a few:

- only suitable for very modest data amounts
- Slow
- Human error prone
- Not safe enough for critical information
- Often not reproducible

Maintaining the boundaries between distinct data sources' silos while filling in the gaps with "federated" queries, which directly

query various source systems and integrate data in real time, is a more long-term solution. Organizations can use SQL query engines like Presto to accomplish this. This federated approach's drawback is that it has a lot of moving pieces and performs poorly with big amounts of data. The truth is that a methodical, repeatable strategy to data integration—a data stack—is necessary for a scalable, sustainable approach to analytics.

9. FUTURE WORK

The framework can be improved in many ways. A few updates which are thought to be added are :

- Run in Pods: Dockerize, build for Kubernetes and Run Jobs and Tasks in Kubernetes. Different ETL tasks require different sized/resourced Pods.
- Parsable Logs: Collect Logs from ETL Jobs and parse them to understand what is failing and reason for failing
- Automated Retries: Retry a failing job for n number of times (already exists for few type of tasks) at job level and various time periods (retry a job after 1 hour if daily job failed)
- ETL Dashboard: ETL Dashboard to show the progress of the job execution, what is failing, one click retries etc.
- Non-NodeJs Tasks: Few tasks are better to run in other languages - Shell script, Python etc. Wrapper for such tasks to embed into NodeJS Job/Task Framework
- Monitoring Support: Monitor Analytics Pipeline also as a first grade component - Raise alerts for failures, handle/recover from failures etc.
- Cost and Stats: Costs of Athena and other operations at fine grain level, building stats dashboard etc.

10. ACKNOWLEDGEMENT

Prof. Suma B, Mr. Venkatesha and Mr. Sai Charan deserve special thanks for their insightful remarks and ideas as well as for giving the authors the chance to examine the development process, artefacts, and records.

11. REFERENCES

[1] Panos Vassiliadis, “A Survey of Extract-Transform-Load Technology.,” International Journal of Data Warehousing and Mining, July 2009

[2] Kamal Kakish, Theresa A Kraft, “ETL Evolution for Real-Time Data Warehousing”, presented at Conference: Proceedings of the Conference on Information Systems Applied Research, At New Orleans Louisiana, USA,2012

[3] Florian Waa, Tobias Freudenreich, Robert Wrembel, Maik Thiele, Christian Koncilia, Pedro Furtado, “On-Demand ELT Architecture for Right-Time BI: Extending the Vision”, International Journal of Data Warehousing and Mining 9(2):21-38 · April 2013

[4] Michael J. Denney, MA,1 Dustin M. Long, PhD,2 Matthew G. Armistead, BS,1 Jamie L. Anderson, RHIT, CHTS-IM,3 and Baqiyyah N. Conway, PhD4, “Validating the Extract, Transform, Load Process Used to Populate a Large Clinical Research Database,” Int. J. Med. Inform., 94, 2016

[5] Valerio Persico, Antonio Montieri, Antonio Pescapè, “On the Network Performance of Amazon S3 Cloud-Storage Service”,5th IEEE International Conference on Cloud Networking (Cloudnet), 2016

[6] Pwint Phyu Khine, Zhao Shun Wang, “Data Lake: A New Ideology in Big Data Era”, 4 th International Conference on Wireless Communication and Sensor Network [WCSN2017], At Wuhan, China, 2017

[7] Benjamin S. Baumer, “A Grammar for Reproducible and Painless Extract-Transform-Load Operations on Medium Data”, arXiv:1708.07073v3 [stat.CO], 23 May 2018

[8] Ibrahim Burak Ozyurt and Jeffrey S Grethe, “Foundry: a message-oriented, horizontally scalable ETL system for scientific data integration and enhancement”, Database (Oxford). 2018.

[9] FabianPrasser, HelmutSpengler, RaffaelBild, JohannaEicher, Klaus A.Kuhn, “Privacy-enhancing ETL-processes for biomedical data”, International Journal of Medical Informatics, Volume 126, June 2019, Pages 72-81

[10] Gustavo V. Machado, Ítalo Cunha, Adriano C. M. Pereira, Leonardo B. Oliveira , “DOD-ETL: distributed on-demand ETL for near real-time business intelligence “, Journal of Internet Services and Applications volume 10, Article number: 21, 2019.

[11] Noussair Fikri, Mohamed Rida, Noureddine Abghour, Khalid Moussaid & Amina El Omri, “An adaptive and real-time based architecture for financial data integration”, Journal of Big Data volume 6, Article number: 97, 2019.

[12] Aiswarya Raj,Jan Bosch,Tian J. Wang,Helena Holmström Olsson, “Modelling Data Pipelines”, at 46th Euromicro Conference on Software Engineering and Advanced Applications (IEEE), 2020.

[13] Marko Jamedžija, Zoran Đurić, “Moonlight: A Push-based API for Tracking Data Lineage in Modern ETL processes”, at 20th International Symposium INFOTEH-JAHORINA(IEEE), 2021.

[14] Noussair Fikri , Mohamed Rida, Noureddine Abghour, Khalid Moussaid, Amina El Omri, “An adaptive and real-time based architecture for financial data integration”, in Springer open, journal of big data, 2019.

[15] Valerio Persico,Antonio Montieri, Antonio Pescapè, “On the Network Performance of Amazon S3 Cloud-storage Service”, at 5th IEEE International Conference on Cloud Networking, 2016.