Open-source ETL Framework using Big Data tools Orchestration on AWS Cloud Platform

MSc Research Project
MSc in Cloud Computing

Sumit Kumar Sahoo
Student ID: x21154589

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sumit Kumar Sahoo
Student ID:	x21154589
Programme:	MSc in Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	01-01-2023
Project Title:	Open-source ETL Framework using Big Data tools & Orchestration on AWS Cloud Platform
Word Count:	7111
Page Count:	17

Signature:	
Date:	1st February 2023

Open-source ETL Framework using Big Data tools & Orchestration on AWS Cloud Platform

Sumit Kumar Sahoo
x21154589

01-01-2023

Abstract

The purpose of this study is to provide a comprehensive review of available open-source ETL(Extract Transform Load) frameworks that can be used to support big data platforms on the Amazon Web Services (AWS) cloud platform. The specific objectives of this review are to (1) identify and evaluate the features of popular open-source ETL frameworks, (2) compare the features of these frameworks with respect to Commercial ETL tools (3) provide recommendations on the best frameworks to use for big data platforms and orchestrating on AWS also considering the cost of Infrastructure. A review of the literature was conducted to identify the most popular open-source ETL frameworks. The frameworks that were identified include Apache NiFi, Apache Beam, Apache Kafka, and Apache Airflow with Pyspark. These frameworks were evaluated based on several criteria, including ease of use, support for multiple data sources and formats, support for multiple data processing engines, and support for cloud-based deployment. Based on the results of the evaluation, it's concluded that its possible to save Infrastructure costs with a refined Cloud Solution Architecture which was improved while doing the development, and an Open Source ETL Big Data framework can be developed using Apache Airflow, Hive, and Pyspark. Terraform (Infrastructure as Code) is used to automate the entire infrastructure on Cloud and promotes the reproducibility of AWS resources with ease in changes.

Index Terms **–ETL, Opensource, Big Data, AWS orchestration, Cloud Cost Optimization**

Contents

1 Introduction

In the current IT landscape, managing data is becoming increasingly complex. As data grows in volume and variety, it becomes more difficult to process using traditional methods. Big data tools and technologies are designed to handle the scale and complexity of modern data. Open-source ETL frameworks can provide a cost-effective way to process big data. These frameworks can be used to extract, transform, and load data from a variety of sources. They can also be used to orchestrate big data processing pipelines on cloud platforms such as Amazon Web Services (AWS). Open-source ETL frameworks can be used to build scalable, reliable, and cost-effective big data solutions. When combined with the power of the cloud, these frameworks can help organizations unlock the value of their data. In this report we are addressing the question - ”With numerous Commercially licensed paid ETL tools in the IT industry, how Open source ETL framework for Big data can be developed by using Cloud services, in order to make it cost-effective & incorporating required ETL features with resilient cloud capbilities?”

Talend is one of the most popular open-source ETL tools available. It is a java-based ETL tool that supports a wide range of data sources and provides a drag-and-drop interface for easy data transformation. Talend is also one of the few ETL tools that offer a free community edition. Pentaho Data Integration (PDI) is another popular open-source ETL tool. It is a java-based ETL tool that offers a rich set of features for data transformation and data integration. PDI also has a community edition that is available for free. Kettle is an open-source ETL tool that is part of the Pentaho suite. Kettle is a java-based ETL tool that offers a drag-and-drop interface for easy data transformation. Kettle also has a community edition that is available for free. Apache NiFi is a powerful open-source ETL tool that can be used for data transformation and data integration. NiFi is a java-based tool that offers a rich set of features for data transformation. NiFi also has a community edition that is available for free. These are just a few of the most popular open-source ETL tools available. Each of these tools has its strengths and weaknesses. You will need to evaluate each tool based on your specific needs to determine which one is the best fit for your organization. In order to give a background to this report, in recent years, big data technologies have emerged as a viable alternative to traditional ETL for data-intensive organizations. Big data tools and platforms are designed to handle large volumes of data at high speeds, making them well-suited for data-intensive applications. In addition, big data technologies are often open-source and cost-effective, making them a more attractive option for many organizations. There are several big data tools and platforms available, each with its own strengths and weaknesses. For this paper, we have selected Apache Hadoop, Spark, and Kafka for data processing, and Apache Airflow for workflow orchestration. Apache Hadoop is a widely used big data platform that is designed for distributed storage and processing of large data sets. Hadoop is scalable and fault-tolerant, making it well-suited for data-intensive applications. Spark is a fast and general-purpose big data platform that is designed for in-memory data processing. Spark is well-suited for iterative and interactive applications. For Apache Hive, Spark and Hadoop ecosystems AWS provides a managed service Elastic Map-reduce (EMR) for big data jobs processing in distributed mechanism. Kafka is a high-throughput distributed messaging system that is designed for real-time data processing. Kafka is used for building real-time data pipelines and streaming applications. Airflow is a workflow orchestration tool that is designed to manage and schedule complex workflows. Airflow is used to orchestrate the data processing workflow in the proposed ETL framework.

2 Related Work

An Open-source ETL framework can be developed using BigData Tools for the parallel processing of exponentially increasing data and Cloud Services can be used for scalable infrastructure orchestration. There are many commercial ETL tools available in the market, but they are expensive and may not be suitable for all organizations. Open-source ETL frameworks provide an alternative that is often more affordable and customizable. BigData tools can be used to develop an Open-source ETL framework that can handle large volumes of data efficiently. Cloud Services can be used to provide a scalable and reliable infrastructure for the framework. The first step is to identify the requirements of the organization and select the appropriate BigData tools. There are many open-source BigData tools available, such as Hadoop, Spark, and Flink. These tools can be used to develop the ETL framework. The next step is to select the Cloud Services that will be used to host the framework. There are many Cloud Service providers, such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure. These providers offer a variety of services, such as storage, computing, and networking. The final step is to deploy the framework on the Cloud Services and test it. Once the framework is deployed, it can be used to process large volumes of data efficiently.

With its implementation in ARKTOS 2, the first contribution to the field of ETL was made. A toolkit for ETL, ARKTOS 2 is in charge of identifying data from the source side. According to the author in Vassiliadis et al. (2002), the system gathers data from many sources, customises it, and then integrates it to send it to a centralised database. This project's objective was to plan and optimise the first data flow. For conceptually modelling the flow of ETL, the author has suggested using unified modelling language. The approach used to simplify operational tasks in ETL, including the integration of many sources, the sources used for transformation, and the properties of the destination, is discussed in greater detail in the article. Additionally, the author discusses the ETL technique and demonstrates how even complex ETL operations may be easily packaged Trujillo and Luján-Mora (2003).

Thomasen Thomsen and Bach Pedersen (2009) created a pygraetl framework that users can utilise to construct ETL packages in the Python programming language. The developers might process data effectively utilizing built-in features via using data access for fact as well as dimension tables. This unique approach was adopted with the goal of cutting down on both execution and development times. Although the performance was superior to GUI tools, only Python users were intended to use it, which severely restricted the user base. The authors suggested a mechanism to perform analytical techniques while integrating different data sources. The method suggested in Macura (2014) outlines the fusion of several data sources and the use of the systematic approach. The sources of incoming data might range widely and are varied. To adopt a proper analytical strategy, the fundamental issue of data integration in the extraction process must be handled. This process of learning from data is known as knowledge discovery. The issue of data integration, in which there is routing in several forms, can be resolved through ETL. Considerably easy administration of the ETL process for data integration is made possible by ETL tools. This work is adaptive to handle data integration and can be automated to complete the daily duty. It can be challenging to provide a connection to the data source using wrapper classes and need programming principles.Bagave (2020)

The paperBansal and Kagemann (2015) explains a definitional ETL framework that is used to aggregate and disseminate data from many sources. The aforementioned study includes manual data analysis, which calls for thorough comprehension. Because it is a manual procedure and takes more time, it is a time-consuming process. Therefore, processing huge data is possible but takes a long time to complete. The difficulty of efficiently and quickly processing a huge volume of data with MapReduce is elaborated in Liu et al. (2012). The paper provides an example of using MR for ETL to import data into a data warehouse. It supports both the star and snow schemas, and the author refers to it as an ETLMR framework. The author of this paper uses map-reduce, which offers high-level details on data ETL. It offers data synchronisation with nodes after integration with MapReduce. Data can be processed in a variety of ways, either by individual activities or by all active operations. In this instance, tasks are handled concurrently once the data is divided into equal chunks. The suggested method can shorten processing time up to a point, but Hadoop still sends data to disc and performs jobs in parallel. The efficiency will undoubtedly rise, however since Map-Reduce is used, the information will be saved in Hdfs on Hadoop, that is less effective than using a spark that keeps data in memory. Big data processing with conventional ETL systems is challenging, according to the research paper Bagave (2020)Bansal (2014), where its observed that it is similar to the issues stated above. The author suggested an ETL system that integrates heterogeneous data using a semantic approach. The approach uncovered fundamental integration and information from several sources. The Resource Description Framework (RDF) is used to build the data as a graph data model, and SPARQL is used to extract useful information. The dataset used for the experiment contains statistics on the economy of transportation and fuel. There is no clear understanding of the approach's effectiveness because the author failed to demonstrate its results. Bagave (2020)

The study by the author in Daneshyar and Razmjoo (2012) demonstrates how Mapreduce on numerous clusters can analyse a sizable amount of big data. Amazon Web Services is used in the experiment to build up Hadoop MapReduce considering multinode cluster. A One GigaByte text data set is gathered and processed using the map-reduce paradigm. Different scenarios with an increased data size are not explained in the article. Additionally, Spark is used for batch files and realtime streaming data as well as ML workloads, whereas MapReduce can only handle batch files. Spark is also much quicker than MapReduce. The implementation, distribution, and execution of jobs for map-reduce are handled by an internal cluster management system. Hadoop is primarily used for batch processing, but we may extract the data in parallel with Hadoop system. Spark is suggested because it processes data in memory and is frequently used for real-time data processing. Another parallel processing strategy is suggested by the research Bagave (2020)paperLiu et al. (2014), where its recommended using CloudETL's parallel ETL architecture, which uses Hadoop with Hive as a warehouse system to put data into databases. Big data tools can be used for a variety of purposes, such as identifying trends, detecting anomalies, and making predictions. There are a number of open-source big data tools available, such as Hadoop, Spark, and Flink. These tools can be used to build a big data processing pipeline.Bagave (2020)

Open Source Big Data ETL frameworks • Apache Hadoop is a popular open-source big data tool. It includes a distributed file system and a MapReduce framework for processing large data sets.

• Spark is a fast and general-purpose big data tool. It can be used for a variety of tasks, such as machine learning, streaming, and SQL.

• Flink is a streaming big data tool. It can be used for real-time stream processing and complex event processing.

• Orchestration tools, such as Apache Airflow, can be used to manage and automate the big data processing pipeline.

• Amazon S3 is a storage that can be utilized to store and recover information. Amazon S3 is a well-known resource for storing Big Data in it. It can be used to write Spark programs as well.

Now let's see how we can use PySpark to build an ETL pipeline. We will be using a sample dataset which is available in the spark-examples library Ravuri and Vasundra (2020). This dataset contains information about the customers of a fictitious company. First, we need to extract the data from the source. For this, we can use the read.csv() method of the PySpark library. This method will read the data from the source and will return a Spark DataFrame. Once the data is extracted, we need to transform it. For this, we can use the map() and filter() methods of the Spark DataFrame. The map() method will apply a function to each row of the DataFrame and the filter() method will select only those rows which satisfy a certain condition. After the data is transformed, we need to load it into the destination. For this, we can use the write.csv() method of the PySpark library. This method will write the data into the destination in CSV format. So, this is how we can use PySpark to build an ETL pipeline. This pipeline can be easily deployed on the AWS Cloud Platform. We can use the Amazon EMR service to launch a cluster of EC2 instances. We can then install PySpark and all the other required software on these instances. Once the cluster is up and running, we can submit our PySpark ETL job to the clusterNagwani (2015). The job will then be executed by the PySpark framework and the results will be stored in the destination. In non-dynamic mode, regardless of whether a hub in the bunch comes up short, information isn't lost, and in failover mode. In many data exchange scenarios, a scalable and persistent database will be able to provide a managed database that is responsible for storing metadata such as job descriptions, job dependencies, or process information. survive to defeat (ie, goal penalty). If one of the nodes in the cluster fails and there is a risk of losing important data, this data center is used to recover the data by repeating the steps required to destroy the line. These products are stored in the store management database. Finally, remember that data storage can be expensive because it takes years to store data. Therefore, unless management requests it, performance data should be stored instead of raw data.

Total Cost of Ownership for infrastructure Provisioning- In the paperBhatnagar and Srinivasa (2013)This case study compares commercial ETL with open source Map Reduce (M/R) based solutions in terms of price and performance. A dual strategy of conducting surveys and experiments has been implemented. The extract and transform (ET) portion of the ETL endeavour is the only focus of the tas where its said to be the main part of the process for business intelligence services. The analysis is being done on basis of processing speed, cost , and resource development they have concluded that using a priperity ETL setup inistial costs a lot and if mapreduce and Spark for processing there is significant improvement in processing of huge datasets.There would be 75% saving in terms of Infrastructure considering the licencing cost and infra setup.

3 Methodology

The purpose of this research is to investigate and compare the effectiveness of open-source ETL frameworks using big data tools and orchestration on the AWS cloud platform as compared to commercial licensed ETL tools. To do this, the research will first identify and review several open-source ETL frameworks. A review of the big data tools and orchestration platforms available on the AWS cloud platform will also be conducted. Based on the findings of the literature review, a ETL framework will be selected for further investigation. A case study approach will be used to collect data from a dataset that have implemented ETL frameworks on the AWS cloud platform. The data collected will be analyzed using both quantitative and qualitative methods. The findings of the research will be used to make recommendations regarding the effective development of open-source ETL frameworks using big data tools and orchestration on the AWS cloud platform.

There are many different open-source ETL frameworks available, each with its advantages and disadvantages. In this research, we will be focusing on the most popular open-source ETL framework- Apache Hadoop. Hadoop is a big data platform that is designed to handle large-scale data processing. It is a distributed file system that allows for the parallel processing of data. Hadoop is a very popular open-source big data platform and is used by many large companies, such as Facebook and Yahoo. Hadoop is a very powerful tool, but it can be complex and difficult to use. In this research, we will be focusing on how to use Hadoop to perform ETL on a large-scale data set. We will be using a publicly available secondary data set from the Stack Overflow Annual Developer Survey. The data set we will be using is a 1020MB data set that With over 70,000 responses fielded from over 180 countries,[1] This data set is perfect for our research because it is a large enough data set that could be difficult to process using a traditional ETL approach. To use Hadoop to perform ETL on this data set, we will need to set up a Hadoop cluster. We will make use of Open source Infrastructure as Code(IaC) – terraform which is a open source IaC and is cloud agnostic for spinning up resources in AWS and installing software in AWS services like EC2. The terraform script would make use of Docker-compose.yml files for installing Apache Airflow. Using Airflow's orchestration capability we will spin up an EMR(Elastic Map reduce) server. We will be using the Amazon Elastic Map Reduce (EMR) service to set up our Hadoop cluster. EMR is a managed Hadoop service that makes it easy to set up and run Hadoop on AWS. Once our Hadoop cluster is up

[1]https://insights.stackoverflow.com/survey

and running, we will use Apache Airflow to import the data set into HDFS from AWS. Sqoop is a tool that is used to transfer data between Hadoop and relational databases. Once the data is in HDFS, we will use PySpark to process and transform the data using the memory cache capability. We would make use of Amazon Redshift for querying the data. After the data is transformed, we will use load the result data using Apache Airflow Operators to S3. This will complete our ETL process. We will evaluate the entire ETL framework developed on the basis of how many file formats and databases it can support, can the infrastructure used in AWS can scale up and down on load testing, we would also be trying to improve the solution architecture for cost-effectiveness for the Opensource ETL framework developed once deployed on AWS. Also, we would be evaluating if the GUI is user-friendly and gives all real-time monitoring information of the ETL process or not.

3.1 Open Source ETL Framework

The Extract, Transform, Load (ETL) structure is a well-known information handling framework that assists data mappings and workflows with moving information starting with one framework and then onto the next. The structure is intended to extricate information from various sources, change it into a configuration that can be stacked into an objective framework, and afterward load it into the destination objects. The Figure 1 shows high-level role of ETL in data warehousing:[2]:

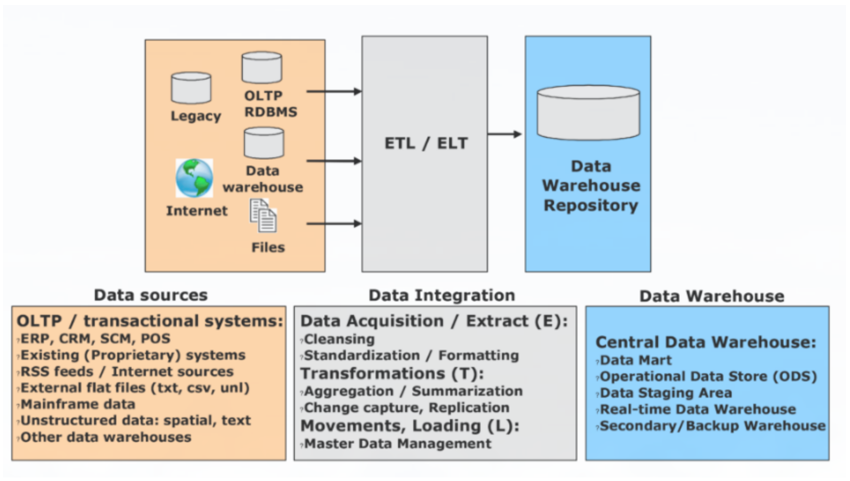

Legacy
OLTP RDBMS
Data warehouse
Internet
Files
ETL / ELT
Data Warehouse Repository
Data sources
Data Integration
Data Warehouse
OLTP / transactional systems:
ERP, CRM, SCM, POS
Existing (Proprietary) systems
RSS feeds / Internet sources
External flat files (txt, csv, unl)
Mainframe data
Unstructured data: spatial, text
Other data warehouses
Data Acquisition / Extract (E):
Cleansing
Standardization / Formatting
Transformations (T):
Aggregation / Summarization
Change capture, Replication
Movements, Loading (L):
Master Data Management
Central Data Warehouse:
Data Mart
Operational Data Store (ODS)
Data Staging Area
Real-time Data Warehouse
Secondary/Backup Warehouse

Figure 1: The high-level role of ETL in data warehousing

The ETL system is used for information handling since it is somewhat simple to utilize and it is truly adaptable. The system can be utilized to remove information from various sources, including data sets, text records, and web apis. It can likewise be utilized to change information into various arrangements, including CSV, XML, and JSON along with different Relational Database engines. The ETL frameworks can likewise be utilized to stack information into a variety of objective frameworks, including data sets, text documents, web APIs and RDBMS engines. The ETL structure is a famous decision for information handling since it is moderately simple to utilize and it is entirely adaptable. The structure can be utilized to remove information from different sources. It can likewise

[2]https://www.redbooks.ibm.com/redbooks/pdfs/sg247788.pdf

be utilized to change information into different arrangements. All these operations of Extraction, Transformation and load need to have an optimum infrastructure depending on velocity, value, veracity, and variety as we as the size of data being processed. It should also have a mechanism to create reusable workflows. The Apache Hive can be used for providing a platform that can take the schema of different data sources and create a common datatype so that there is not datatype incompatibility leading to data truncation or data loss, Pyspark is good for transformations of data which can be used for data filtering, sorting, aggregations and also for Upsert transformations. Apache Airflow on the other hand provides plugins and operators for connecting to cloud storage platform like S3, RDS, as well as for transferring data from different sources to hive and vice versa irrespective of the structured file formats as well as different Database engines.

As the world is moving towards more digitization, data is also increasing exponentially. No doubt the Oil of this era is Data. So, to maintain this data and to make some sense of it, ETL tools are required. ETL stands for Extract, Transform and Load. It is a process of how data is collected, transformed, and loaded into a destination database. There are many ETL tools available in the market like Informatica, DataStage, SSIS, Talend, etc. But, these tools are not open source and are very costly. So, in this report, we will be discussing an open-source ETL framework called PySpark.

PySpark is an open-source Python library that provides APIs to support Apache Spark. Apache Spark is a fast and general engine for laraggregationsa processing. It can be used for processing structured, unstructured, and streaming data. There are many benefits of using PySpark as platform for ETL tool. Some of them are:
1. PySpark is easy to use and it has a friendly API.
2. PySpark is very fast. It can process large amounts of data very quickly.
3. PySpark is very versatile. It can be used for processing structured, unstructured, and streaming data.
4. PySpark can be easily integrated with other Big Data tools and frameworks.
5. PySpark is open source and it is very cost-effective.

3.2 Orchestration on AWS Cloud Platform

AWS Cloud Orchestration is the process of managing and provisioning cloud resources using templates or scripts. This allows you to manage your resources in a more organized and automated way. With orchestration, you can provision new resources, update existing resources, and delete resources in a controlled and coordinated manner. You can also use orchestration to manage your application deployments and infrastructure changes. Orchestration can help you save time and money by automating the provisioning and management of your cloud resources. It can also help you improve your application's resilience and reduce the risk of human error. You may develop, modify, and upgrade AWS infrastructure in a safe and predictable manner using the open-source software application called Terraform which is an open source Infrastructure as Code(IaC) and is cloud agnostic. that permits you to spin up, update, and erase AWS services in a controlled and facilitated manner. It works with AWS assets, including Amazon EC2 instances, Amazon RDS databases, and Amazon S3 services for making changes to the information stored in the bucket. Amazon EC2 instances. Consequently, handles the provisioning and resources of your AWS assets, including Amazon EC2, Amazon RDS data sets, and

Amazon S3 services. terraform also provides several features to help you manage your application deployments, such as versioning, rollbacks, and health monitoring.Orozco-GómezSerrano (2020).

AWS Cloud Formation:AWS CloudFormation is a service that allows developers to define and provision AWS resources in a template [1]. CloudFormation templates are written in JSON or YAML and can be used to provision resources such as Amazon EC2 instances, Amazon S3 buckets, and Amazon DynamoDB tables. CloudFormation templates can be used to provision resources in a consistent and repeatable manner and can be used to provision resources across multiple AWS accounts and regions. CloudFormation templates can also be used to share configurations between teams and can be used as a part of an automated deployment process. AWS CloudFormation templates are stored in Amazon S3 and can be launched from the AWS Management Console, the AWS Command Line Interface (CLI), or the AWS SDKs.Bhatnagar and Srinivasa (2013)

Infrastructure As Code- There are a variety of orchestration tools available, each with its strengths and weaknesses. The three most popular orchestration tools are Ansible, Puppet, and Chef.

• Ansible is a popular configuration management tool that is known for its simplicity and ease of use. Ansible can be used to automate a variety of tasks such as provisioning, configuration management, and application deployment. Ansible is designed to be lightweight and easy to install, making it a good choice for small and medium-sized deployments, such as applying security patches in an automated way on Linux/Windows Servers.

• Puppet is another popular configuration management tool that is known for its flexibility and extensibility. Puppets can be used to automate a variety of tasks such as provisioning, configuration management, and application deployment. Puppet is designed to be highly scalable and can be used for large and complex deployments.
• Chef is a newer configuration management tool that is known for its speed and efficiency. Chef can be used to automate a variety of tasks such as provisioning, configuration management, and application deployment. Chef is designed to be highly parallelizable and can be used for large and distributed systems.

• Terraform When using a declarative method as Terraform does, the code is always an accurate representation of your infrastructure's most recent state. Without having to think about the past or the present, you can quickly determine what is deployed and how it is constructed. Because you don't have to manually take into consideration the status of the world at any given time, this also makes it simple to write reusable code. Instead, you simply describe the state you want to be in, and Terraform figures out how to transition between them effortlessly. As a result, Terraform codebases typically remain compact and simple. Of course, declarative languages have drawbacks as well. Your expressive potential is constrained if you lack access to a complete programming language. For instance, it can be challenging to specify some infrastructure modifications in simply declarative terms, such as a rolling, zero-downtime deployment. Similarly, writing generic, reusable code might be challenging without the ability to perform "logic" (such as if-statements and loops) (especially in CloudFormation). Input variables, output variables, modules, create before destroy, count, and interpolation functions are just a few of the strong primitives offered by Terraform, which enables the creation of clear, comprehensible, modular code even in declarative languages.Brikman (2016)

4 Design Specification

The Open Source ETL Framework is an ETL framework that provides a unified approach to extract, transform, and load (ETL) data from a variety of sources, including structured and unstructured data. The framework is designed to be extensible and scalable, and it includes several built-in connectors for popular data sources. The framework is open source and available under the Apache License. The Open Source ETL Framework is based on the Apache Hadoop platform and uses the MapReduce programming model to parallelize data processing across a cluster of nodes. The framework includes several built-in connectors for popular data sources, such as relational databases, NoSQL databases, and file systems. The framework is extensible and scalable, and it can be used to process both batch and streaming data Bala et al. (2017). The Open Source ETL Framework is an ideal choice for organizations that want to process large amounts of data in a distributed and parallelized manner. The framework is also a good choice for organizations that want to use a unified approach to ETL data from a variety of sources.

The proposed ETL Solution Architecture shown in Figure 2 is designed to be scalable and extensible, making it well-suited for data-intensive applications. The framework makes use of Apache Hadoop, PySpark for transformation, and Hive for data processing for standardizing DataTypes and processing platforms within AWS EMR, Apache Airflow for workflow orchestration and Terraform for automating Infrastructure deployments.

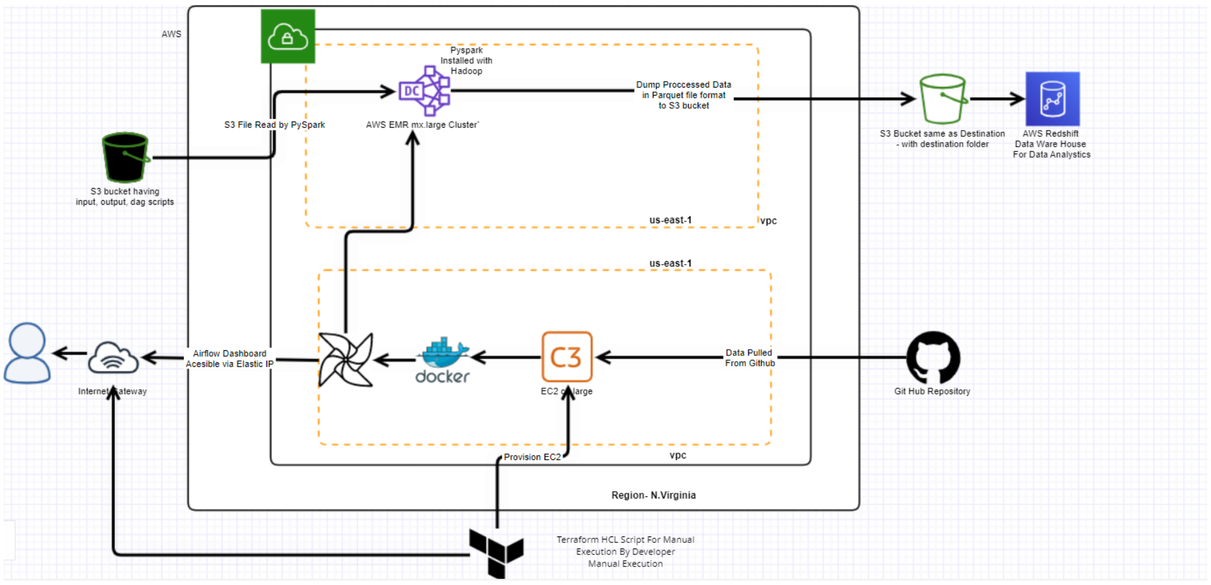

AWS
Pyspark Installed with Hadoop
S3 File Read by PySpark
AWS EMR mx.large Cluster'
Dump Proccessed Data in Parquet file format to S3 bucket
S3 bucket having input, output, dag scripts
S3 Bucket same as Destination - with destination folder
AWS Redshift Data Ware House For Data Analystics
us-east-1
vpc
us-east-1
Airflow Dashboard Acesible via Elastic IP
Internet Gateway
docker
EC2 large
Data Pulled From Github
Git Hub Repository
Provision EC2
vpc
Region- N.Virginia
Terraform HCL Script For Manual Execution By Developer Manual Execution

Figure 2: ETL Solution Architecture

The above architecture flow is explained as follows:
1) Terraform HCL scripts are run, it will spin up EC2 instance and install Apache Airflow using docker-compose, inside it and provision S3 bucket also.

2) The DAG script put inside the S3 bucket has folder having scripts folder to contain the script to transfer data from S3 to EMR and after Pyspark transformations to put the processed parquet format file to S3
3) After deploying the pipeline script scheduling the DAG, it would trigger the DAG which would spin up an EMR inside Auto-scaling Group which is useful for horizontal scaling during CPU usage thresholds being reached.
4) The dataset's .csv file present in S3 bucket would be processed inside EMR.
i. The dag script in EMR which will pull data from the GitHub repo has code to pull .csv from S3 and transfer to EMR for Pyspark transformations
ii. Pyspark is used for Transformations
5) After the data is processed, it the task in DAG will stop EMR after checking with Airflow Operator Xcoms if the file is processed completely and put in destination folder of same S3 bucket 6) The resulting data in parquetData is then mapped and imported to Redshift using last task whose script is present in Scripts folder of S3 bucket and later the imported data can be queried for data analysis using Amazon Redshift and the result can be staged inside Redshift.
) .

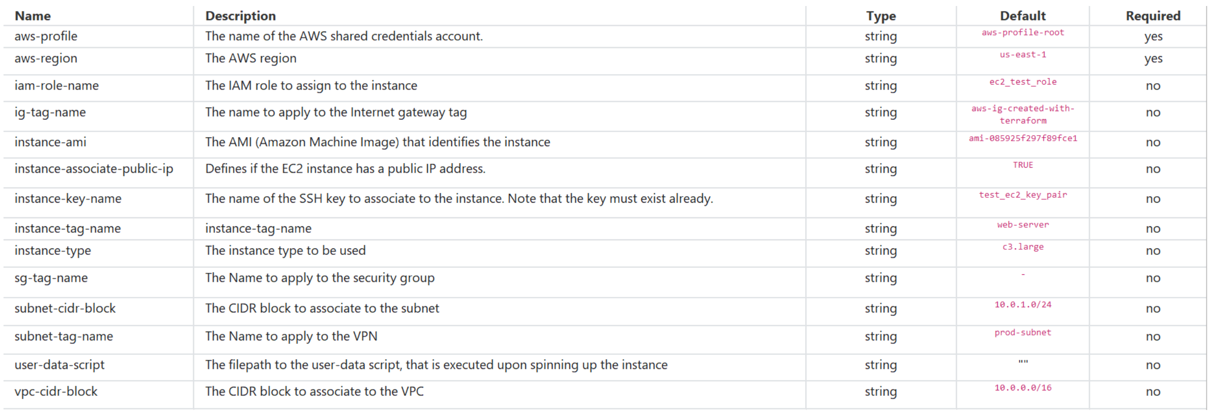
The Public Secondary Dataset used here is a .csv file which is taken from [3] Stack Overflow Annual Developer Survey

5 Implementation

There have been 3 setups first using reserved AWS services, second using AWS Managed Services and third which is the improved Architecture as shown in Figure 2. Third and final setup is what I will explain here. As the setup is comprised of Terraform, Docker-compose for Apache Airflow, EC2 instance, AWS EMR, S3 and AWS Redshift, I'll reveal the details of Implementation of relevant tools and services.
1. Cloudtrails Create an S3 bucket to store logs for the trail by default settings in order to track all the operations of services. This is for audit purpose
2. Terraform: The stack created by the terraform module is composed of the below components and Implementation details as shown in Figure 4
Compute -EC2 instance **Network** -Virtual Private Cloud -Internet Gateway -Security Group -Route table **Storage** -S3 **Security** -IAM roles

i. This terraform script spins up an Unix c3.large EC2 instance with a .cer file which is a certificate and public key for SSH in the instance.
ii. Using docker-compose.yml file we set up Apache Airflow inside the EC2 instance using th automated Terraform scripts.
2. S3: Create a S3 bucket with source and destination folder object for ETL purpose.
3. AWS Redshift: Create a Redshift Cluster of Production Cluster configuration of dc2.large with 1 node for high availability and persistent performance and setup a 'dev' Database with connection setup with user defined Database username and password. The queries can be written and saved in the Query editor.
4. AWS EMR: Its preconfigured in Apache Airflow DAG etl pipeline script.py steps to spin up an EMR cluster with m5.xlarge master node during the entire life cycle of etl pipeline run. This EMR cluster would terminate automatically at the end of the DAG

[3]https://insights.stackoverflow.com/survey



Name	Description	Type	Default	Required
aws-profile	The name of the AWS shared credentials account.	string	aws-profile-root	yes
aws-region	The AWS region	string	us-east-1	yes
iam-role-name	The IAM role to assign to the instance	string	ec2_test_role	no
ig-tag-name	The name to apply to the Internet gateway tag	string	aws-ig-created-with-terraform	no
instance-ami	The AMI (Amazon Machine Image) that identifies the instance	string	ami-085925f297f89fce1	no
instance-associate-public-ip	Defines if the EC2 instance has a public IP address.	string	TRUE	no
instance-key-name	The name of the SSH key to associate to the instance. Note that the key must exist already.	string	test_ec2_key_pair	no
instance-tag-name	instance-tag-name	string	web-server	no
instance-type	The instance type to be used	string	c3.large	no
sg-tag-name	The Name to apply to the security group	string	-	no
subnet-cidr-block	The CIDR block to associate to the subnet	string	10.0.1.0/24	no
subnet-tag-name	The Name to apply to the VPN	string	prod-subnet	no
user-data-script	The filepath to the user-data script, that is executed upon spinning up the instance	string	""	no
vpc-cidr-block	The CIDR block to associate to the VPC	string	10.0.0.0/16	no

Figure 3: Terraform Implementation details

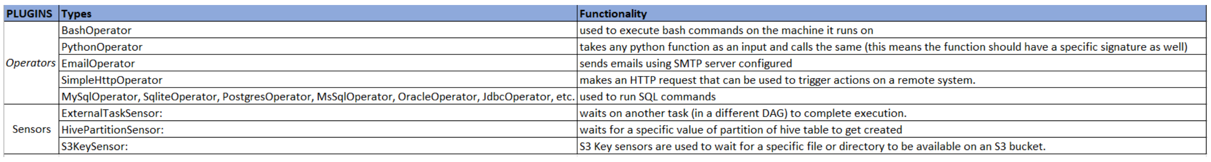
run, in order to save the price incurred during running the EMR cluster.

6 Evaluation

We have evaluated the Open-source Big data ETL framework various features which are available in Commercial ETL Tools like Informatica Power Centre Designer Tool. The evaluation parameters are described below:

6.1 Integration with different types of Data Source & File Formats

With the combined capability of Apache Airflow Plugins and Hive, Hue it supports integration with various data sources and File Formats.
DAGs are made up of several tasks s. To define jobs in Airflow, we employ operators and sensors (that are a sort of operator).A typical Operator offers integration to another service making it possible to use such services via Airflow.[4] These helps to pull data from different Types of Data sources i.e from S3, RDBMS engines, File Systems.The function of Airflow Plugin Sensors, a particular class of operators, is to wait for an internal or external trigger. These are frequently used to start some or all of the DAG in response to an outside event. Figure 4 shows All the necessary available plugins to integrate with different Sources the good feature of Airflow is we can make custom plugins as well:



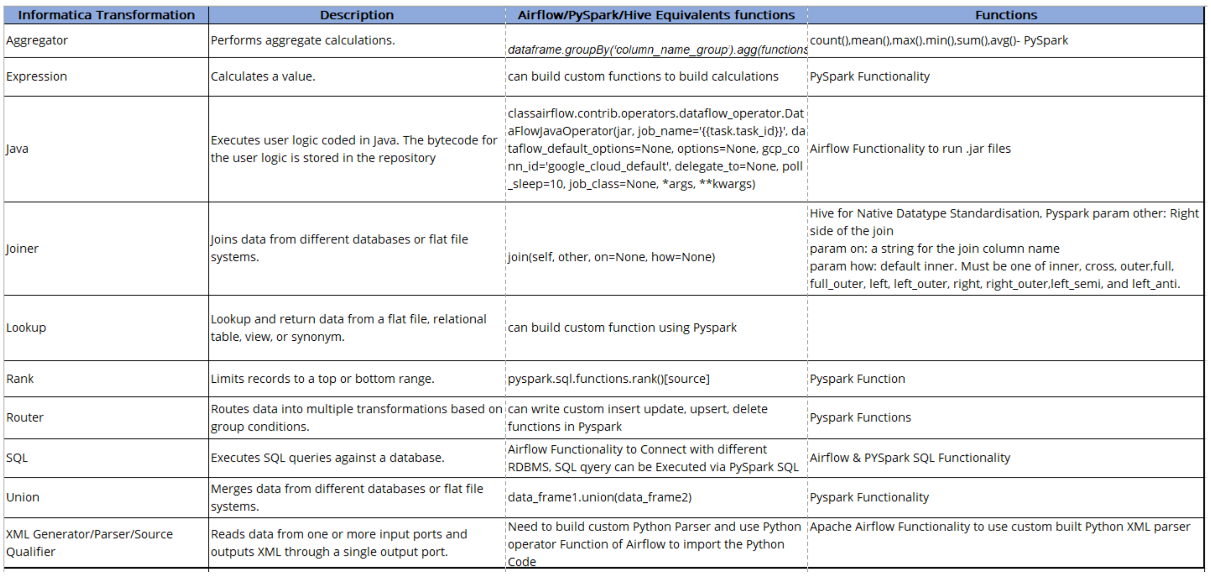
PLUGINS	Types	Functionality
Operators	BashOperator	used to execute bash commands on the machine it runs on
	PythonOperator	takes any python function as an input and calls the same (this means the function should have a specific signature as well)
	EmailOperator	sends emails using SMTP server configured
	SimpleHttpOperator	makes an HTTP request that can be used to trigger actions on a remote system.
	MySqlOperator, SqliteOperator, PostgresOperator, MsSqlOperator, OracleOperator, JdbcOperator, etc.	used to run SQL commands
Sensors	ExternalTaskSensor:	waits on another task (in a different DAG) to complete execution.
	HivePartitionSensor:	waits for a specific value of partition of hive table to get created
	S3KeySensor:	S3 Key sensors are used to wait for a specific file or directory to be available on an S3 bucket.

Figure 4: Airflow Plugin

[4]https://www.qubole.com/tech-blog/apache-airflow-tutorial-dags-tasks-operators-sensors-hooks-xcom

6.2 Data Transformers - Transformation

This section is evaluated on the Basis of Data Transformations available in Informatica Power Centre ETL Tool [5] with Respect to Apache Airflow, Hive , and Pyspark SQL Functions as shown in Figure 5. We can conclude that Pyspark has capabilties to do allmost all the necessary SQL transformations which are available in Informatica PowerCenter Tool for Big Data Management.



Informatica Transformation	Description	Airflow/PySpark/Hive Equivalents functions	Functions
Aggregator	Performs aggregate calculations.	*dataframe.groupBy('column_name_group').agg(functions*	count(),mean(),max().min(),sum(),avg()- PySpark
Expression	Calculates a value.	can build custom functions to build calculations	PySpark Functionality
Java	Executes user logic coded in Java. The bytecode for the user logic is stored in the repository	classairflow.contrib.operators.dataflow_operator.DataFlowJavaOperator(jar, job_name='{{task.task_id}}', dataflow_default_options=None, options=None, gcp_conn_id='google_cloud_default', delegate_to=None, poll_sleep=10, job_class=None, *args, **kwargs)	Airflow Functionality to run .jar files
Joiner	Joins data from different databases or flat file systems.	join(self, other, on=None, how=None)	Hive for Native Datatype Standardisation, Pyspark param other: Right side of the join param on: a string for the join column name param how: default inner. Must be one of inner, cross, outer,full, full_outer, left, left_outer, right, right_outer,left_semi, and left_anti.
Lookup	Lookup and return data from a flat file, relational table, view, or synonym.	can build custom function using Pyspark	
Rank	Limits records to a top or bottom range.	pyspark.sql.functions.rank()[source]	Pyspark Function
Router	Routes data into multiple transformations based on group conditions.	can write custom insert update, upsert, delete functions in Pyspark	Pyspark Functions
SQL	Executes SQL queries against a database.	Airflow Functionality to Connect with different RDBMS, SQL qyery can be Executed via PySpark SQL	Airflow & PYSpark SQL Functionality
Union	Merges data from different databases or flat file systems.	data_frame1.union(data_frame2)	Pyspark Functionality
XML Generator/Parser/Source Qualifier	Reads data from one or more input ports and outputs XML through a single output port.	Need to build custom Python Parser and use Python operator Function of Airflow to import the Python Code	Apache Airflow Functionality to use custom built Python XML parser

Figure 5: Data Transformation in Informatica vs Opensource Tools used in this Research

6.3 Cloud Security, Encryption, GDPR Compliance

Data security: Data security and privacy are major concerns when working with big data. The Apache Hadoop project has implemented several features to help protect data security and privacy. For example, Hadoop supports data encryption at rest and in transit, and Hadoop applications can be configured to run with Kerberos authentication to help ensure that data is only accessible to authorized users. In addition, the Amazon EMR service provides several features to help secure data in transit, including SSL encryption and Amazon S3 server-side encryption. Data security and privacy are complex issues, and there is no one-size-fits-all solution. The best approach to data security and privacy will vary depending on the specific requirements of the organization. In this paper, we will discuss some of the data security and privacy considerations that should be considered when using Hadoop and Amazon EMR. Another relevant feature in Amazon Managed Apache Airflow Managed Service was it had Data Protection and Privacy Tab for managing data in different regions.

Data Encryption: Data encryption is a process of transforming readable data into an unreadable format. Data encryption can be used to protect data at rest (i.e. data that is stored on disk) and data in transit (i.e. data that is being transferred over a network)

[5] https://www.edureka.co/blog/informatica-transformations/

Chhabra et al. (2020). Hadoop supports data encryption at rest using the Hadoop Transparent Encryption feature. Hadoop Transparent Encryption allows data to be encrypted without changing the way that applications access the data. Hadoop Transparent Encryption uses the Java Cryptography Extension (JCE) to provide encryption for Hadoop data. Hadoop also supports data encryption in transit using SSL. SSL is a protocol that provides communication security over a network. SSL uses encryption to protect data in transit from being intercepted and read by unauthorized users. Hadoop applications can be configured to use SSL to encrypt data in transit. In addition to the data encryption features that are built into Hadoop, the Amazon EMR service provides additional features to help secure data in transit. Amazon EMR utilizes Amazon S3 server-side encryption to encode information very still in Amazon S3. Amazon S3 server-side encryption utilizes AES-256 to encode information before it is kept in touch with Amazon S3, and decodes information when it is perused from Amazon S3. Amazon EMR additionally utilizes SSL to encode information on the way between Amazon EMR hubs and Amazon S3.

Data Audit GDPR Compliance: A data audit is the process of tracking and logging data access and data changes. Data audits can be used to track and log data access and data changes. Data audits can also be used to help detect and investigate security incidents. Hadoop applications can be configured to use data audits. Amazon EMR supports data audit through the use of Amazon CloudTrail. Amazon CloudTrail is a service that records AWS API calls made by Amazon EMR. Amazon CloudTrail records information about the identity of the user who made the API call, the time of the API call, the API method that was called, and the parameters that were passed to the API method. Amazon CloudTrail can be used to track and log data access and data changes. Amazon CloudTrail can also be used to help detect and investigate security incidents.

Incase we want to use the AWS incorporated GDPR compliance features in AWS (Amazon Managed Workflows for Apache Airflow (MWAA) through a combination of infrastructure, tools, and services that help customers manage and secure their personal data, we can make use of it although it might lead to increased cost. Some of these include:
Encryption,Access controls, Data deletion of personal data, Auditing for tracking persona data, Compliance certifications - AWS has achieved multiple certifications, including GDPR, that demonstrate its commitment to protecting personal data.
By using MWAA, customers can take advantage of AWS's security and compliance capabilities and help ensure that their Apache Airflow workflows are GDPR compliant.

6.4 Total Cost of Ownership - Experiment

As the Tools used are open source , and using a cloud platform like AWS gives us the pay as you go methodology. The TCO of the entire implementation of infrastructure for 1 year using AWS Pricing Calculator i.e. $20,827.72 USD for using On-Demand Services as well as improvement in Infrastructure to Automate shutting of EMR cluster after use. The experiment done as Per Figure 2 was for 10 days of similar infrastructure being running along with an On-Demand EMR cluster running even when not in use resulted in the Highest contender for AWS pricing. This resulted in taking actions for using the EMR cluster only during the Data Pipeline runtime lifecycle which means that once the data is extracted, processed by Pyspark inside EMR and put back into S3 destination folder after the validation of both processes the EMR cluster was closed resulting in just 100 minutes of runtime charges for a span of 10 days. This was done using the orchestration capabilities of Apache Airflow installed in EC2 instance.

Due to lack of trustworthy data available for licensing Commercial ETL tool it cant established if the Opensource Big Data ETL framework could be cost-effective or not with respect to Commercial ETL tools. It may vary from different use cases. Although an article[6] explains about how Open-source Big Data ETL framework can be adopted with 80% data integration costs. Which states that the initial Cost of Implementing Data Integration Application may range from 70,000 to 220,000 USD if its setup On-premises considering Software and Hardware costs.

6.5 Features Of Orchestration Tool- Apache Airflow

The Apache Airflow gives the below features which gives the upper hand on selecting it as ETL orchestration tool:
1) Airflow CLI can be used without GUI in case of web-server Internet gateway issues for Dashboard.
2) real Time logging via task logs
3) Operators, Sensors, Hooks to connect with S3, Hive, and Database engines
4) Connection String Parameters for connecting with AWS, DB engines.
5) Python Based APIs to build custom plugins.
6) Rich Apache Airflow Developer Community which gives space for more features and improvements.

6.6 Discussion

Considering the most important part of this case study is utilising the pay as you go framework of AWS. It can be very costly if infrastructure management is not done properly. this is the reason why in enterprise-level setups , if there the is no IAM roles provided to specific group of users then it can tools to huge costs just the way it lead me into during early phase of experimenting cost-effective architecture setup. AWS is a good choice as Cloud Vendor as it gives completely 'minute level' of configuration changes. Considering the Pyspark's capability to ally do all the functionality given by Informatica power centre Designer tool, Without writing the data to storage, Spark may read the data in, complete all the ETL in memory, and then give the data to MLlib/Redshift for analysis. Informatica is a closed system. Open source development tools are used by Spark, including Python/PySpark, Scala, Java, SQL, and R/SparkR. In Spark, you can perform all lookups, joins, data purification, data transformation, and enrichment operations. The most popular Spark use case at the moment is ETL. A general-purpose networked computing engine is called Spark. Examples of data standardisation and MDM on Spark can be found in Databricks. Your ETL tasks will move considerably more quickly on Spark.Apache Airflow on the other hand along with its good features as mentioned in Section 6.5 has few issues, it doesnot give tracking details of every tasks' sub task unlike

[6]https://www.altoros.com/blog/guide-to-reducing-etl-and-data-integration-costs-by-80-percent/

Informatica ETL tool, one more challenge is we cant do debugging on the Apache Airflow platform GUI. We have to heavily depend on prgramtically logging. Terraaform as a declarative language is cloud agnostic and can be used in major popular cloud platforms and goes hand in hand with steps of init, plan, apply and destroy infrastructure on the cloud along with it, Terraform enables the reproducibility of infrastructure with ease.

7 Conclusion and Future Work

On the conclusion part I would like to propose that its feasible to develop Open Source ETL framework using Pyspark, Apache Airflow for task, orchestration as well as for AWS services provisioning. Use open source Infrastructure as code like Terraform makes use of Declarative language and is cloud agnostic. The only blocker could be lack of GUI level subtask tracking and runtime debugging. Also if the Solution architecture is planned considering the analysis of load of processing and data volume, optimum AWS services should be used in order to reduce cost of Infrastructure as in this case large capacity Datawatre house - Redshift, EC2 , EMR were used for small datasets

References

Bagave, R. (2020). *Enhancing Extraction in ETL flow by modifying as P-ECTL based on Spark Model*, PhD thesis, Dublin, National College of Ireland.

Bala, M., Boussaid, O. and Alimazighi, Z. (2017). A fine-grained distribution approach for etl processes in big data environments, *Data & Knowledge Engineering* **111**: 114–136.

Bansal, S. K. (2014). Towards a semantic extract-transform-load (etl) framework for big data integration, *2014 IEEE International Congress on Big Data*, IEEE, pp. 522–529.

Bansal, S. K. and Kagemann, S. (2015). Integrating big data: A semantic extract-transform-load framework, *Computer* **48**(3): 42–50.

Bhatnagar, V. and Srinivasa, S. (2013). *Big Data Analytics: Second International Conference, BDA 2013, Mysore, India, December 16-18, 2013, Proceedings*, Vol. 8302, Springer.

Brikman, Y. (2016). Why we use terraform and not chef, puppet, ansible, saltstack, or cloudformation, *Retrieved April* **24**: 2020.

Chhabra, G. S., Singh, V. P. and Singh, M. (2020). Cyber forensics framework for big data analytics in iot environment using machine learning, *Multimedia Tools and Applications* **79**(23): 15881–15900.

Daneshyar, S. and Razmjoo, M. (2012). Large-scale data processing using mapreduce in cloud computing environment, *International Journal on Web Service Computing* **3**(4): 1.

Liu, X., Thomsen, C. and Pedersen, T. B. (2012). Mapreduce-based dimensional etl made easy, *Proceedings of the VLDB Endowment* **5**(12): 1882–1885.

Liu, X., Thomsen, C. and Pedersen, T. B. (2014). Cloudetl: scalable dimensional etl for hive, *Proceedings of the 18th International Database Engineering & Applications Symposium*, pp. 195–206.

Macura, M. (2014). Integration of data from heterogeneous sources using etl technology, *Computer Science* **15**.

Nagwani, N. K. (2015). Summarizing large text collection using topic modeling and clustering based on mapreduce framework, *Journal of Big Data* **2**(1): 1–18.

Orozco-GómezSerrano, A. (2020). Adaptive big data pipeline.

Ravuri, V. and Vasundra, S. (2020). Moth-flame optimization-bat optimization: Map-reduce framework for big data clustering using the moth-flame bat optimization and sparse fuzzy c-means, *Big Data* **8**(3): 203–217.

Thomsen, C. and Bach Pedersen, T. (2009). pygrametl: A powerful programming framework for extract-transform-load programmers, *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pp. 49–56.

Trujillo, J. and Luján-Mora, S. (2003). A uml based approach for modeling etl processes in data warehouses, *International Conference on Conceptual Modeling*, Springer, pp. 307–320.

Vassiliadis, P., Simitsis, A. and Skiadopoulos, S. (2002). Conceptual modeling for etl processes, *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pp. 14–21.