B. TECH. PROJECT REPORT

On

A Scalable Data Science Platform for Healthcare Product Research and Reliability

BY
Saurabh Sharma

DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2019

A Scalable Data Science Platform for Healthcare Product Research and Reliability

A PROJECT REPORT

Submitted in partial fulfillment of the
requirements for the award of the degrees

of
BACHELOR OF TECHNOLOGY
in

ELECTRICAL ENGINEERING

Submitted by :
Saurabh Sharma

Guided by :
Dr. Srivathsan Vasudevan
Associate Professor
Discipline of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY INDORE
DECEMBER 2019

CANDIDATE'S DECLARATION

I hereby declare that the project entitled **A Scalable Data Science Platform for Healthcare Product Research and Reliability** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Electrical Engineering completed under the supervision of **Dr. Srivathsan Vasudevan, Associate Professor, Discipline of Electrical Engineering**, IIT Indore is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Saurabh Sharma
160002053
B.Tech., Discipline of Electrical Engineering
IIT Indore, India

CERTIFICATE by BTP Guide

It is certified that the above statement made by the students is correct to the best of my knowledge.

Dr. Srivathsan Vasudevan
Associate Professor
Discipline of Electrical Engineering
IIT Indore

PREFACE

This report on **A Scalable Data Science Platform for Healthcare Product Research and Reliability** is prepared under the guidance of Dr. Srivathsan Vasudevan, Associate Professor, Discipline of Electrical Engineering, IIT Indore and Mr. Pavan Burbure, Sr. Software Architect, GE Healthcare, Bengaluru.

Through this report, I have tried to give a detailed description of the project I have worked on, in GE Healthcare, Bengaluru for six months as a part of my B.Tech. project. I have explained the data flow and the architecture which I have developed at GE Healthcare, Bengaluru. I further have explained the working of the developed solution with the help of the results that were generated using a working prototype.

I have tried to the best of my abilities and knowledge to explain the content of my project in a lucid manner. I have also added figures explaining the working of the developed platform to make my description more illustrative.

SAURABH SHARMA
B. Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

ACKNOWLEDGEMENT

I would like to thank my B.Tech. project supervisor Dr. Srivathsan Vasudevan for his constant guidance and support. Without his support, this report would not have been possible.

I am especially grateful to Mr. Pavan Burbure (Sr. Software Architect, GE Healthcare) who was my supervisor in GE Healthcare, Mr. Santosh Kumar Raju (Staff Software Engineer, GE Healthcare) who was my mentor at GE Healthcare for their constant support and valuable insights to help me with various challenges I faced in the course of this project.
I would also like to thank my manager at GE Healthcare Mr. Shivashankar Gangadhar (Senior Manager, GE Healthcare) for his support and providing me the opportunity to work in his team on this project.

Special thanks to all members of Interventional team at GE Healthcare for making my six months project duration full of learning.
I would like to thank my friends and family for their constant support and motivation.

Finally, I would like to thank God and all my elders for their blessings.

SAURABH SHARMA
B. Tech. IV Year
Discipline of Electrical Engineering
IIT Indore

ABSTRACT

Healthcare equipment is critical for patients, and therefore, the reliability of this equipment is of high importance. During their operation, equipment's health and usage data are generated and recorded. Due to the large size, high velocity, and lack of structure, processing and analysis of this data is a key challenge for engineers.

A scalable data science platform was designed and developed to address the above mentioned problem. Relevant literature was studied during the course of this project to learn modern techniques in data engineering. I have used workflow orchestration tool- Apache Airflow for workflow scheduling, No-SQL database Elasticsearch for data storage and Kibana for Visualization.

The developed platform creates a data lake to store this real-time data. This platform also provides robust analytics in near real-time for product research and reliability.

ABBREVIATIONS

IGS - Image Guided System
ETL - Extract, Transform, Load
DAG - Directed Acyclic Graph
JSON - JavaScript Object Notation
UI - User Interface

Contents

List of Figures

Chapter 1

Introduction

In this project, a scalable data science platform was developed for product research and reliability of Interventional Image Guided System (IGS) at Wipro GE Healthcare, Bengaluru, India.

1.1 Company Background

GE Healthcare is a leading global medical technology company that manufactures and markets medical equipment. The Company offers medical imaging, information technology, medical diagnostics, patient monitoring systems, and bio-pharmaceutical manufacturing technology. GE Healthcare has offices in many countries, including India.

1.2 Interventional Image Guided System

The Interventional Image Guided System (IGS) is equipment which uses X-Ray for imaging and is designed to support a variety of procedures such as interventional radiology, pediatrics, electrophysiology, neuro interventions, and body imaging procedures [1]. Figure 1.1 shows GE Interventional Image Guided System

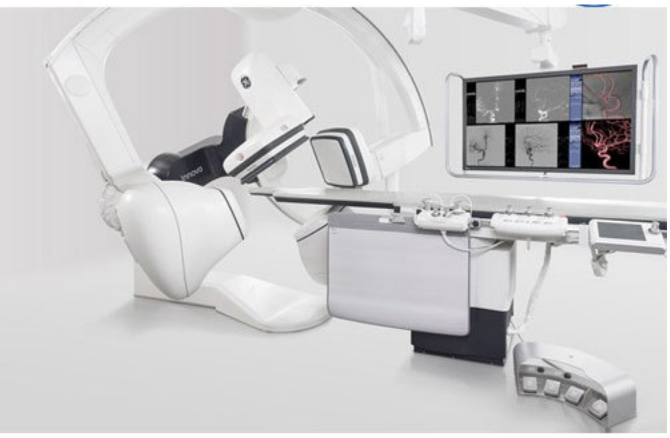


Figure 1.1: GE Healthcare Image Guided System

[1]

1.3 Product Research and Reliability

Reliability is the characteristic of equipment or software that relates to the integrity of the system and the ability to maintain trouble free operation to insure against failure. To ensure the reliability of a system, simulation of actual operation of the system is implemented, and equipment operations data is recorded, later this data is analyzed for potential faults.

Once equipment is installed at the Hospital, equipment usage data containing information like configuration of the machine, sensor data is used to monitor this equipment remotely. Monitoring of medical equipment is of high significance as this data provides information about equipment health, usage and helps to take preventive measures to avoid faults in the future. The data fetched from this equipment is used for analytics and research purposes.

1.4 Current Practices

At present, the central monitoring of IGS is a manual process. Due to the complex nature of the data from the equipment, these manual techniques are slow and unable to provide analytics in real-time. Furthermore, a large amount of data from multiple sources and preprocessing of the data make it difficult for engineers to analyze the data manually. A single day usage of the equipment alone generates a large volume of data making it impossible to monitor all the data manually in real-time.

1.5 Challenges in Product Research and Reliability

Interventional Image Guided Systems are located at different geographic locations around the world which uploads machine operations data at a fixed interval. Central monitoring of these machines will assist in product research and reliability. There are multiple challenges involved in building a central monitoring system:

- Data uploaded by the equipment may contain missing information, the information in local languages, the same information in different formats by different equipment
- Network issue or database connection issue may result in the failure of data ingestion
- Due to the large size, high velocity, and unstructured data, it is near impossible to analyze this big data manually
- Analytics on this data need to be done on a regular basis or as per need, the process to obtain final results constitutes of multiple steps and therefore there is need for automation

1.6 Scope of Study

The above mentioned problem is an excellent case for scalable data science platform where the scalable platform will provide automatic near real-time analytics on the data fetched from equipment.
As there was no existing data pipeline to fetch and store data from the equipment. I contributed to the development of a data pipeline to fetch and clean the raw data. A study is carried out to decide tools for building a data pipeline. Once data is cleaned, and relevant data is merged

together, it is stored in a distributed No-SQL database. The platform also provides tools for visualization and statistical analysis of the stored data.

1.7 Area of Concentration

For this thesis, the chosen area of concentration is to explore workflow schedulers and to design and develop the architecture of the platform. Apache airflow was chosen for building a data pipeline because of its rich tracking and monitoring capabilities, scalable distributed architecture, and active community support. Study of the chosen workflow scheduler- Apache Airflow and NoSQL database Elasticsearch. For this project, Kibana was used as a visualization tool.

1.8 Purpose and Goal

The objective of this project is to build a data science platform for Interventional Image Guided System (IGS) Usage Research and Reliability. IGS is critical for patients as it is used during surgery, and therefore, its reliability is highly important. A platform is developed, and its architecture is detailed in this report.

1.9 Methodology

In this project, a study is first carried out to identify the common tools and techniques for developing a platform. Tools are selected from available options based on factors like data privacy, cost, community support. Implementation of data fetching, cleaning, and storage is implemented in Python. The platform architecture is designed, ensuring low cost and high speed.

1.10 Outline

There are four chapters in this thesis. The first chapter introduces the background and purpose of this platform, while Chapter 2 gives a brief description of related theories and tools used in the industry. The third chapter focuses on the developed platform architecture and working. Chapter 4 discusses further work that can be explored in the future and concludes the report.

Chapter 2

Relevant Theory

This chapter explains briefly the relevant theory related to the project.

2.1 Extract, Transform, Load (ETL)

Extraction, Transform, Load popularly called ETL are used by organization with multiple data databases to store different types of data. The ETL process is used to integrate data that was spread across these databases. As the number of different data formats, sources, and systems have expanded, ETL has now become a standard in organization to integrate all these data. It has now become a core component of an organization data science tool kit [5].

2.2 Data Engineering Tools

There are multiple tools available to implement a data engineering solution. For this project I have studies two main tools: Apache Airflow and Elasticstack. Apache Airflow is used for workflow orchestration. Data storage, search, and dashboards are implemented using Elastic stack. The following subtopic will describe both tools in brief:

2.2.1 Apache Airflow

Apache Airflow is an open-source workflow orchestration tool. Airflow is used to programmatically author, schedule, and monitor workflows. Airflow was started in October 2014 at Airbnb. Apache Software Foundation announced Airflow as a Top-Level Project in January 2019 [2].

Airflow Task

Airflow Task is a defined unit of required work which needs to be implemented. The required work is implemented in python.

Airflow Directed Acyclic Graph (DAG)

In Airflow, the workflow is designed as a Directed Acyclic Graph (DAG). A Directed Acyclic Graph or DAG in Airflow is a collection of all the tasks which need to be executed. DAG reflects the relationships and dependencies between all the tasks which constitute the Airflow DAG. Basically, Airflow DAG is a directed graph which does not contain any cycle and nodes of this graph are tasks. It describes how different tasks are executed and is not concerned with what constituent tasks do.

2.3 Elastic stack

Elastic Stack constitutes of two components Elasticsearch and Kibana.

2.3.1 Elasticsearch

In this project, Elasticsearch was used to store, search, and analyze data in near real-time. Elasticsearch is a NoSQL database build on top of Apache Lucene, which is an open-source search engine library [3]. It stores data as structured JSON documents.

2.3.2 Kibana

For visualization and analytics on stored data, Kibana was used in this project. Kibana is an open source data exploration and visualization tool [4]. It provides powerful features to build interactive graphs like histograms, pie charts, and heat maps. It has geospatial support as well, which is useful in this project as these data are collected from different locations. Kibana integrates strongly with Elasticsearch and therefore provides near real-time visualization of data stored in Elasticsearch.

Chapter 3

Platform Architecture Development

In this chapter, the first section explains the architecture developed, the subsequent section describes developed DAG, data flow between various components, hardware setup for deployment, and the last section explains the working of the proposed solution.

3.1 Platform Architecture

Developed architecture constitutes three main components- Data Lake, Apache Airflow, Elastic stack. Figure 3.1 shows high level connections of all components of the architecture.

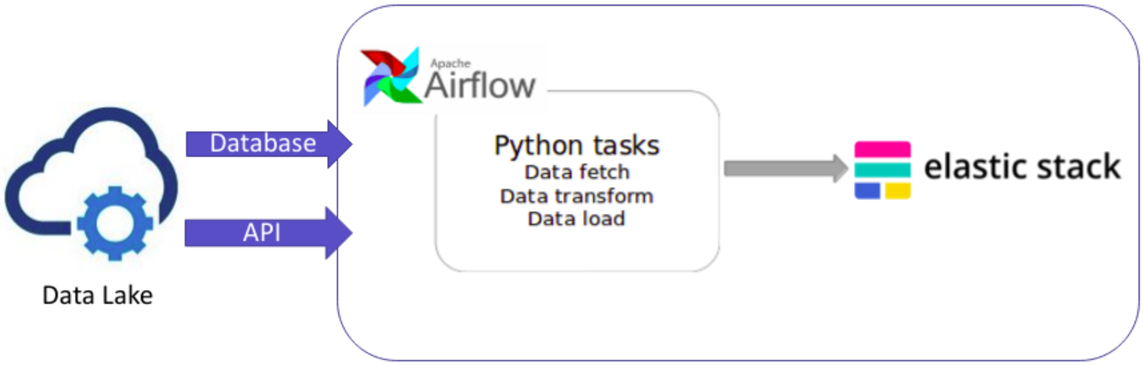


Figure 3.1: High Level Connection between Various Components of Platform Architecture

Data is fetched from data lake on a regular basis or as per need and injected into Elasticsearch. Elasticsearch provides near real time search operations on this data. Kibana, a data visualization and exploration tool, is used to build dashboards containing interactive dynamic graphs for analytics on this stored data.

Scheduling and monitoring of each step from fetching data from different sources to injecting it into Elasticsearch is implemented using airflow, workflow orchestration tool, airflow provides Airflow User Interface (Airflow UI) to monitor and manually manage DAG. Since airflow execution is distributed, it can run tasks in parallel and therefore decreasing the overall execution time.

Detailed architecture with the flow of data is shown in figure 3.2:

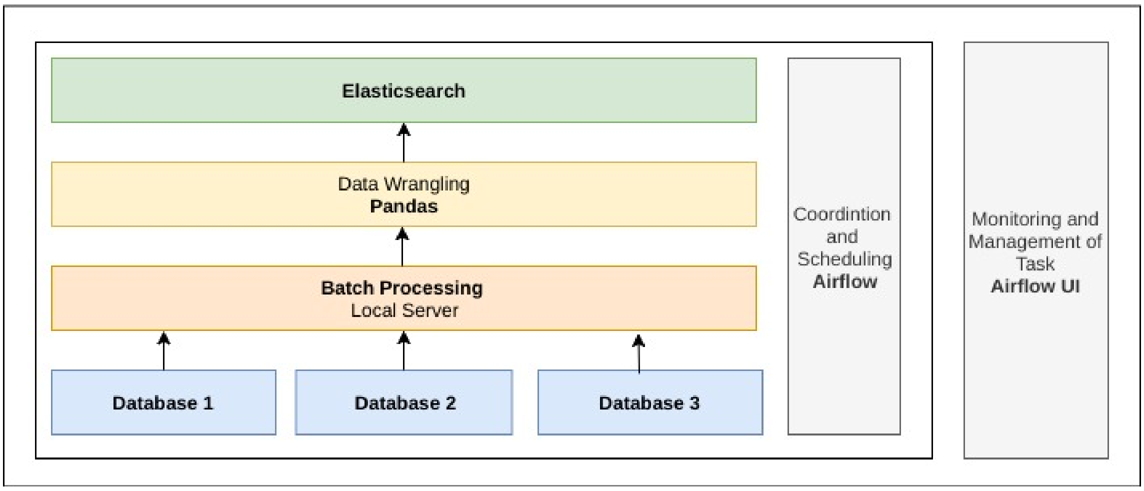


Figure 3.2: Developed Platform Architecture

Working of the developed architecture is as follows:
Data is fetched from multiple sources in batches and stored in the local server, fetched data is cleaned and relevant information in the data from different sources is merged, the last stage is injecting cleaned and merged data into Elasticsearch. These stages are implemented on a daily basis and can also be implemented as per need. The daily execution of the developed platform is implemented using Apache Airflow DAG. Each stage is programmatically author in airflow DAG as Task. Each Task is a custom python script written to implement the logic of each stage.

3.2 Data Flow

In this section, the data flow between the different components is explained. A popular concept called Extract, Transform, Load (ETL) is used in this project. Implementation of ETL process means the extraction of data from multiple sources, transforming the extracted data into required formats, and then loading the data into desired storage. Figure 3.3 explains the flow of data in the developed platform as ETL process.

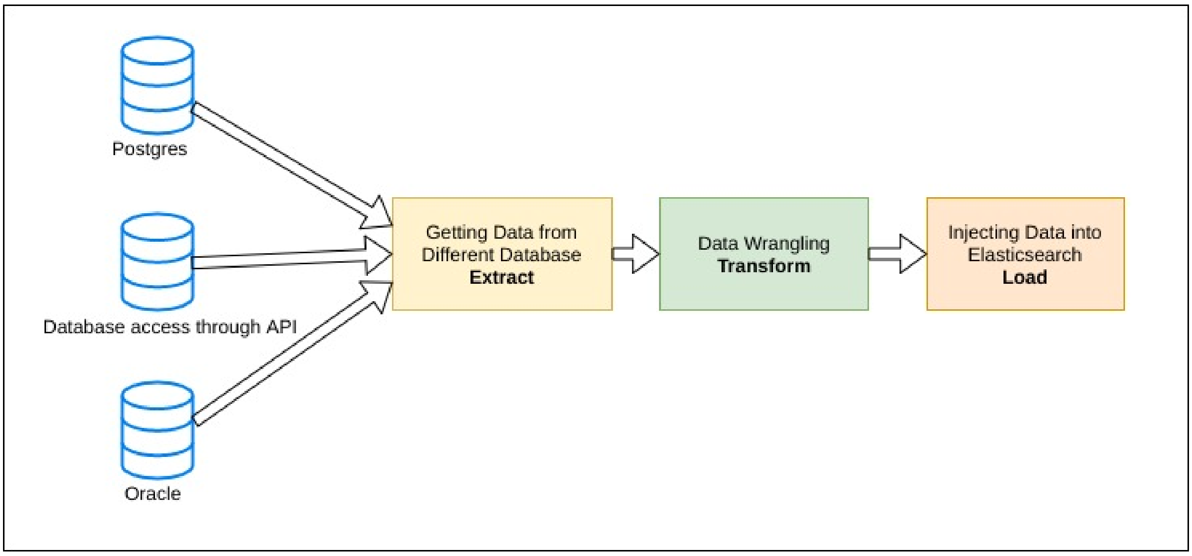


Figure 3.3: Extract, Transform, Load (ETL)

3.3 Directed Acyclic Graph (DAG)

In computer science and mathematics, a directed acyclic graph (DAG) is a graph that is directed and contains no cycles connecting the other edges. The edges of the directed graph only go in one way. In airflow, DAG nodes are the tasks and edges are the connection between tasks. Airflow DAG also defines the dependencies of tasks with each other.

Figure 3.4 shows developed DAG.

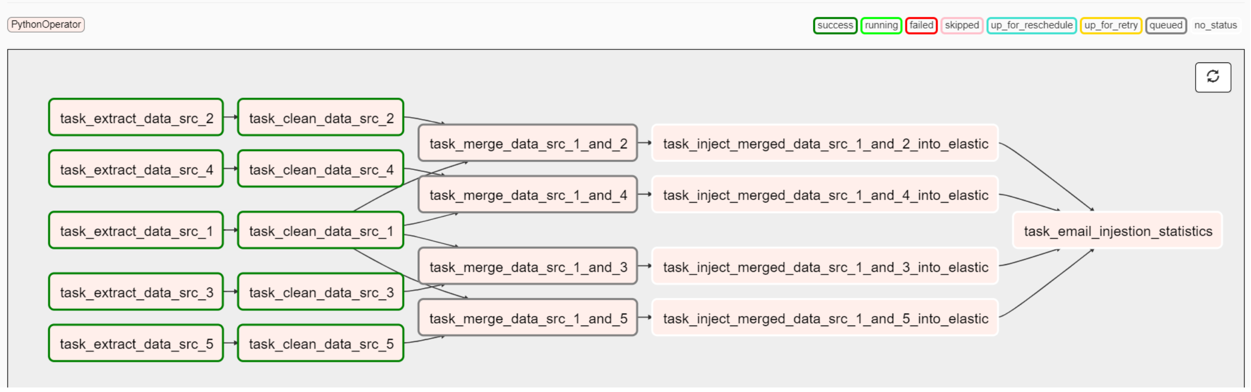

PythonOperator
success
running
failed
skipped
up_for_reschedule
up_for_retry
queued
no_status
task_extract_data_src_2
task_clean_data_src_2
task_extract_data_src_4
task_clean_data_src_4
task_extract_data_src_1
task_clean_data_src_1
task_extract_data_src_3
task_clean_data_src_3
task_extract_data_src_5
task_clean_data_src_5
task_merge_data_src_1_and_2
task_merge_data_src_1_and_4
task_merge_data_src_1_and_3
task_merge_data_src_1_and_5
task_inject_merged_data_src_1_and_2_into_elastic
task_inject_merged_data_src_1_and_4_into_elastic
task_inject_merged_data_src_1_and_3_into_elastic
task_inject_merged_data_src_1_and_5_into_elastic
task_email_injestion_statistics

Figure 3.4: Developed DAG

Developed DAG contains tasks with functionality as follows

task_extract_data_src_1: This task fetches data from source 1

task_extract_data_src_3: This task fetches data from source 3

task_extract_data_src_4: This task fetches data from source 4

task_extract_data_src_5: This task fetches data from source 5

task_clean_data_src_1: This task cleans data fetched from source 1

task_clean_data_src_2: This task cleans data fetched from source 2

task_clean_data_src_3: This task cleans data fetched from source 3

task_clean_data_src_4: This task cleans data fetched from source 4

task_clean_data_src_5: This task cleans data fetched from source 5

task_merge_data_src_1_and_2: This task merges the data fetched from source 1 and source 2

task_merge_data_src_1_and_3: This task merges the data fetched from source 1 and source 3

task_merge_data_src_1_and_4: This task merges the data fetched from source 1 and source 4

task_merge_data_src_1_and_5: This task merges the data fetched from source 1 and source 5 task_inject_merged_data_src_1_and_src_2_into_elastic: This task injects merged data from source 1 and source 2 into Elasticsearch

task_inject_merged_data_src_1_and_src_3_into_elastic: This task injects merged data from source 1 and source 3 into Elasticsearch

task_inject_merged_data_src_1_and_src_4_into_elastic: This task injects merged data from source 1 and source 4 into Elasticsearch

task_inject_merged_data_src_1_and_src_5_into_elastic: This task injects merged data from source 1 and source 5 into Elasticsearch

task_email_ingestion_statistics: This task sends an email to the engineer containing statistics about data, it also informs engineer regarding successful completion of the pipeline.

3.4 Hardware Setup

This section describes the setup used for the deployment of developed architecture. Figure 3.5 shows the configuration of Hardware used for deployment.

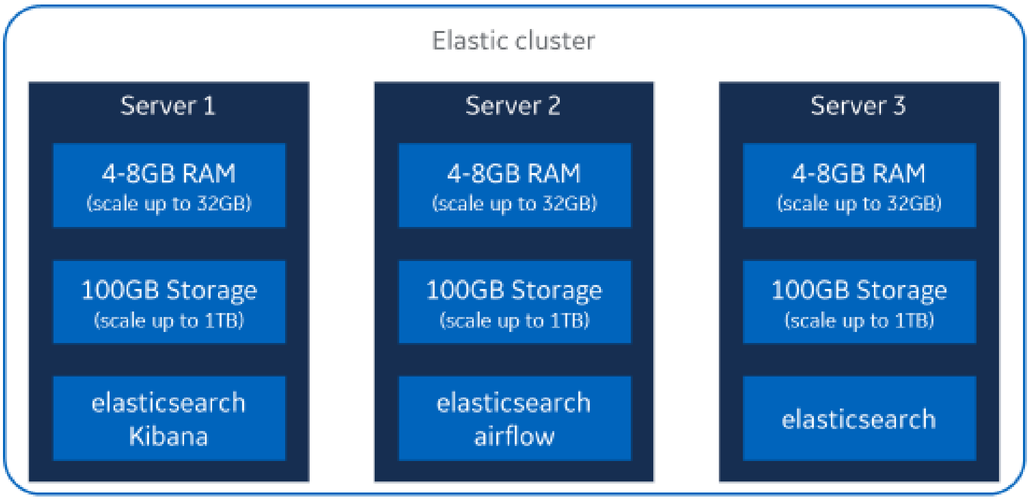


Figure 3.5: Deployment Setup

It consists of three node cluster, Elasticsearch is distributed in each node i.e. node 1, 2, and 3. Kibana runs on node 1 and airflow on node 2. Each node has 4 GB RAM, which can be extended up to 32 GB. Each node has 100 GB storage, which is extendable up to 1TB.
In the future, as the need for more storage will increase, new nodes can be added to the cluster. Since Elasticsearch is distributive, it runs on all the three nodes and as it is scalable in future more nodes can be added to increase storage capacity.

3.5 Working of the setup

Working of the setup is explained in two parts: Part I explains the overall working of the architecture and Part II explains the working of the developed DAG.

3.5.1 Part I: Overall working of the setup

Figure 3.6 shows the data flow graph. Data is fetched from the database through the Airflow Task, which contains python code for connection to the database and fetching of the data.

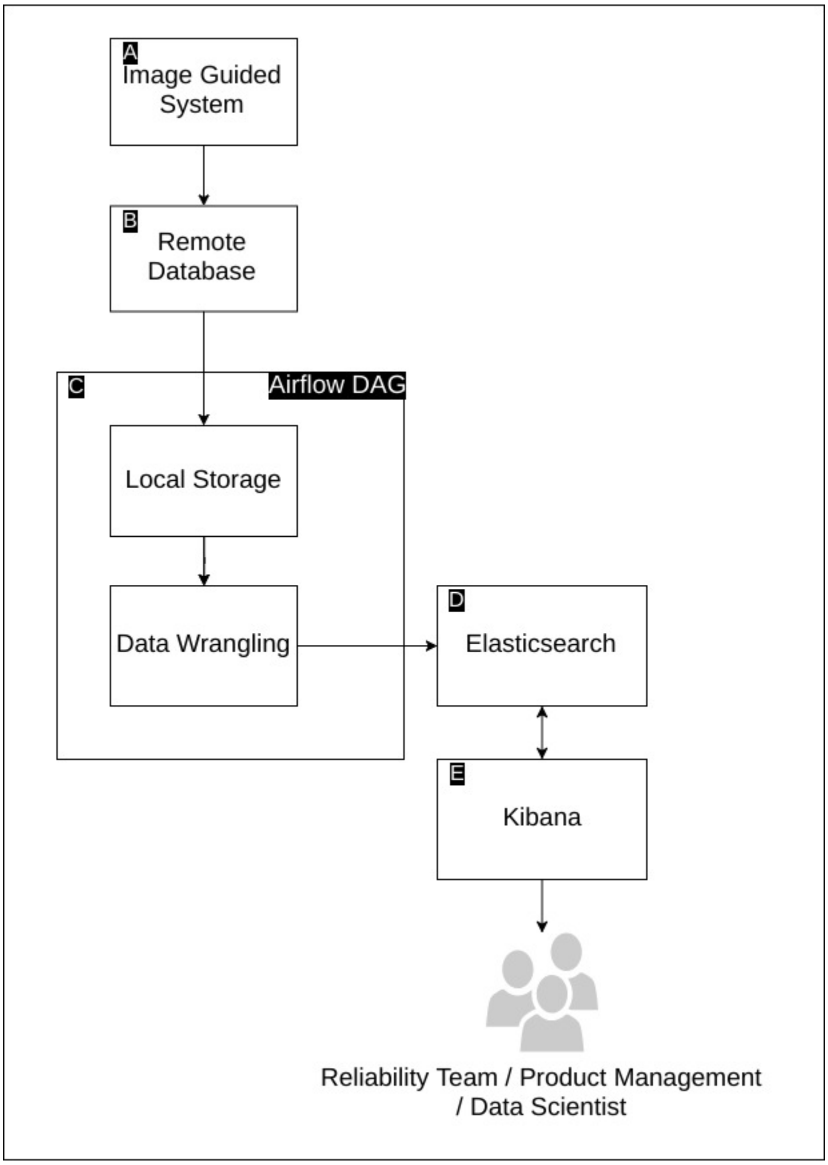


Figure 3.6: System Architecture

Fetched data is cleaned and then injected into Elasticsearch. Kibana contains dashboards which constitute of various graphs and metric build on the data. These dashboards are available to product management, reliability team, and data scientists.

3.5.2 Part II: Working of DAG

In this sub-section working of the developed DAG is explained. Refer figure 3.7 and 3.8.

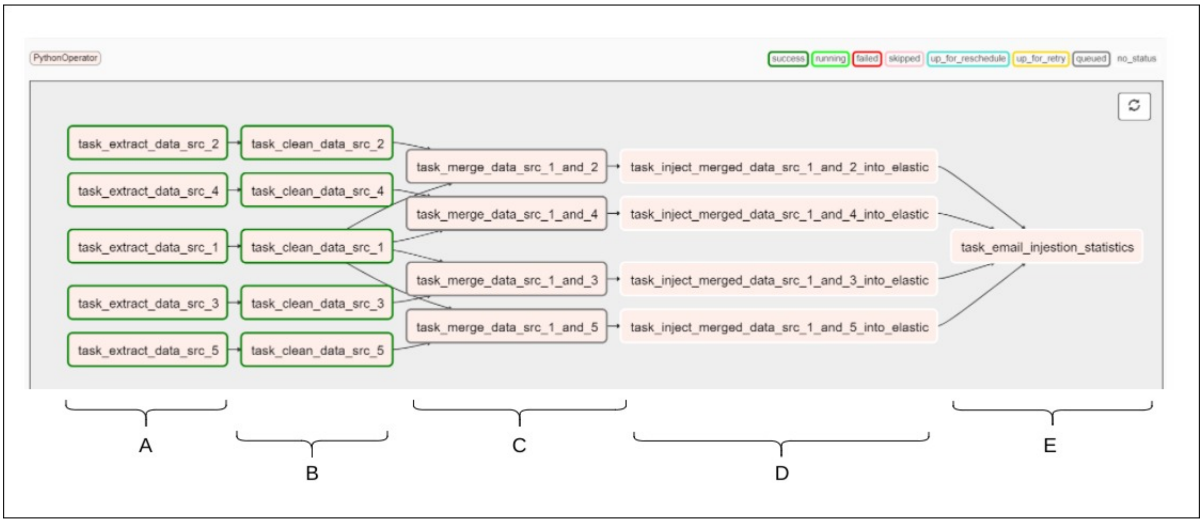
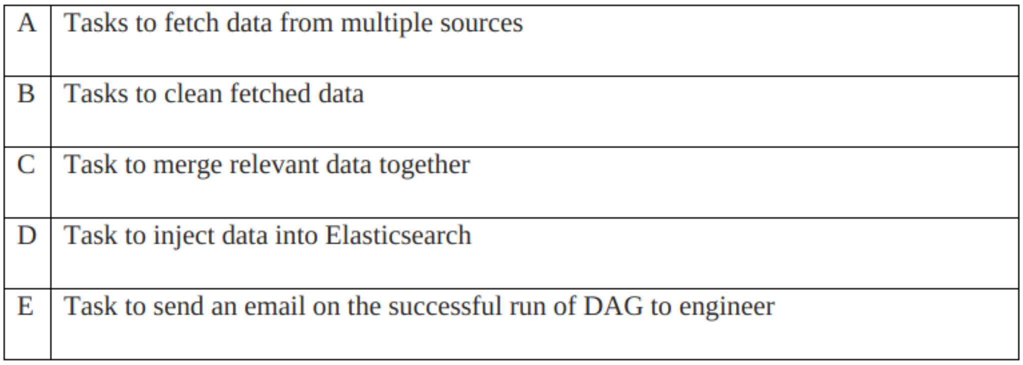

PythonOperator
success
running
failed
skipped
up_for_reschedule
up_for_retry
queued
no_status
task_extract_data_src_2
task_extract_data_src_4
task_extract_data_src_1
task_extract_data_src_3
task_extract_data_src_5
task_clean_data_src_2
task_clean_data_src_4
task_clean_data_src_1
task_clean_data_src_3
task_clean_data_src_5
task_merge_data_src_1_and_2
task_merge_data_src_1_and_4
task_merge_data_src_1_and_3
task_merge_data_src_1_and_5
task_inject_merged_data_src_1_and_2_into_elastic
task_inject_merged_data_src_1_and_4_into_elastic
task_inject_merged_data_src_1_and_3_into_elastic
task_inject_merged_data_src_1_and_5_into_elastic
task_email_injestion_statistics
A
B
C
D
E

Figure 3.7: Developed DAG



A	Tasks to fetch data from multiple sources
B	Tasks to clean fetched data
C	Task to merge relevant data together
D	Task to inject data into Elasticsearch
E	Task to send an email on the successful run of DAG to engineer

Figure 3.8: Description of Tasks of DAG

task_extract_data_src_1 constitutes of a custom python script to fetch data from source 1 and store it in the local server. Similarly, other 4 tasks from A in figure 3.7 constitute of a custom python script to fetch data from the sources 2,3,4, and 5. When the DAG starts execution at the scheduled time, these five tasks run in parallel. task_clean_data_src_1 constitute of a custom code in python, whose objective is to clean the data fetched from source 1.

The arrow in the DAG represents the dependencies of a task over others. In the developed DAG subsequent tasks will run only when all connected preceding tasks are successfully completed. It is intuitive to understand that tasks to clean data will run only after when data is fetched from the source. In the case of failure of a task, Airflow provides feature to only rerun the failed tasks and not the entire DAG, this rerun of only failed task and not the entire DAG saves execution time in case of failure of any task. This concept is explained through an example, as shown in figure 3.9

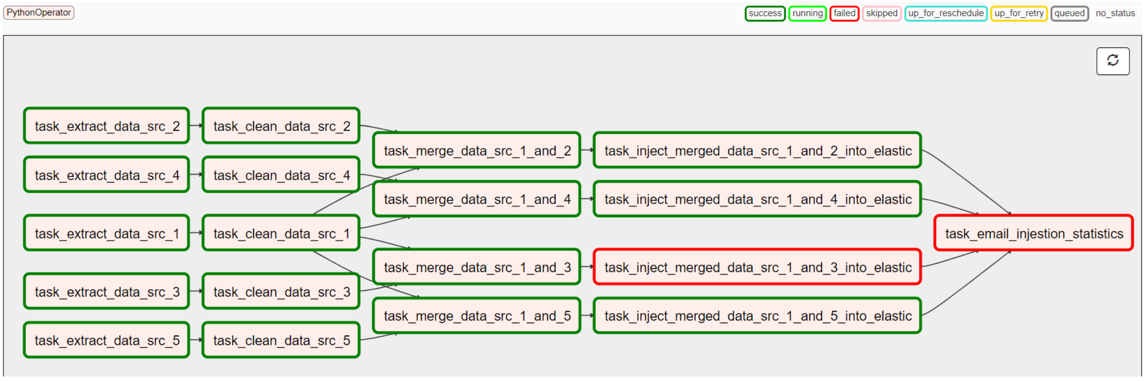

PythonOperator
success
running
failed
skipped
up_for_reschedule
up_for_retry
queued
no_status
task_extract_data_src_2
task_clean_data_src_2
task_extract_data_src_4
task_clean_data_src_4
task_extract_data_src_1
task_clean_data_src_1
task_extract_data_src_3
task_clean_data_src_3
task_extract_data_src_5
task_clean_data_src_5
task_merge_data_src_1_and_2
task_merge_data_src_1_and_4
task_merge_data_src_1_and_3
task_merge_data_src_1_and_5
task_inject_merged_data_src_1_and_2_into_elastic
task_inject_merged_data_src_1_and_4_into_elastic
task_inject_merged_data_src_1_and_3_into_elastic
task_inject_merged_data_src_1_and_5_into_elastic
task_email_injestion_statistics

Figure 3.9: DAG with failed Task

If task_inject_merged_data_src_1_and_3_into_elastic failed which also results in failure of the dependent task, in this example which is task_email_injestion_statistics, this failure may be due to network connection failure to Elasticsearch or Elasticsearch server is not running. DAG is developed to accommodate failure cases and the scheduler will rerun only the failed task after a specified time interval, if it failed after specified multiple reruns, Airflow will send an email to engineer with email containing a description of failure.
Figure 3.10 shows DAG execution statistics. Our developed DAG is set to run daily. Airflow provides color code to display the status of each task. It shows the status of each task of DAG execution for each day. This DAG execution statistics is very helpful for the engineer to analyze the working of each task.

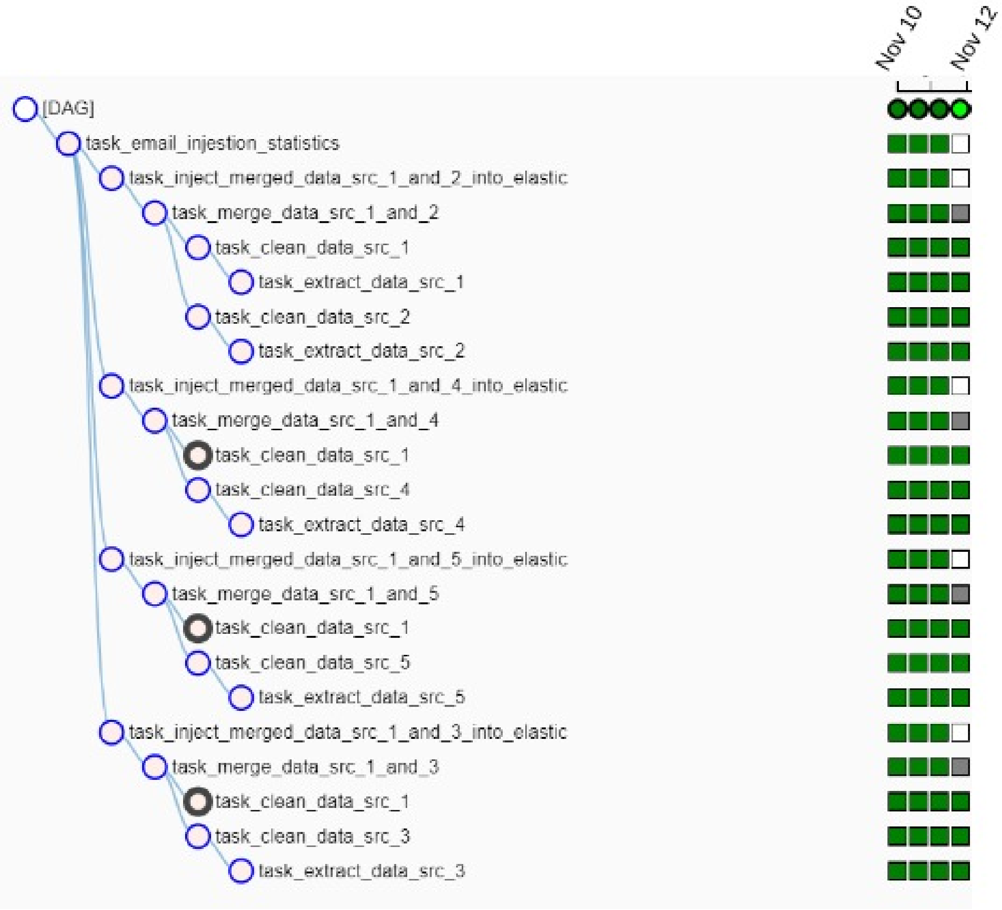

Nov 10
Nov 12
[DAG]
task_email_injestion_statistics
task_inject_merged_data_src_1_and_2_into_elastic
task_merge_data_src_1_and_2
task_clean_data_src_1
task_extract_data_src_1
task_clean_data_src_2
task_extract_data_src_2
task_inject_merged_data_src_1_and_4_into_elastic
task_merge_data_src_1_and_4
task_clean_data_src_1
task_clean_data_src_4
task_extract_data_src_4
task_inject_merged_data_src_1_and_5_into_elastic
task_merge_data_src_1_and_5
task_clean_data_src_1
task_clean_data_src_5
task_extract_data_src_5
task_inject_merged_data_src_1_and_3_into_elastic
task_merge_data_src_1_and_3
task_clean_data_src_1
task_clean_data_src_3
task_extract_data_src_3

Figure 3.10: Screenshot of Developed DAG Monitoring using Airflow UI

Chapter 4

Future Work and Conclusion

This chapter discusses the future work of the project. In this project, the developed platform provides near real time analytics and access to clean, structured data. Developed platform opens many possibilities of work that can be done in the domain of predictive analytics. The next stage of this project is to build predictive analytics on top of this platform using machine learning techniques. Multiple learning algorithms can be implemented using data from the platform for training. The developed platform can be extended to other critical equipment.
In conclusion, the scalable platform build on open source technologies provides organization access to structured, clean data and analytics on this data in near real time which significantly helps engineers in improving reliability and quality of the equipment.

Bibliography

[1] GE Healthcare, Interventional Image Guided System - https://www.gehealthcare.com/products/interventional-image-guided-systems

[2] Apache Airflow, open source workflow scheduler - https://airflow.apache.org/

[3] Elasticsearch, open source search engine - https://www.elastic.co/products/elasticsearch

[4] Kibana, open source data visualization tool - https://www.elastic.co/products/kibana

[5] Extract, Tranform, Load - https://www.sas.com/en_us/insights/data-management/what-is-etl.html