الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليـــــم العالـي و البحـث العلمـي
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
جامعة محمد البشير الإبراهيمي بـرج بـوعـريـريـج
Université Mohamed El Bachir El Ibrahimi de Bordj Bou Arreridj

Faculté des Mathématiques et d'Informatique
Département d'Informatique

Mémoire

Présenté en vue de l'obtention du diplôme

Master en Informatique

Spécialité : Réseau et Multimédia

THÈME

Architecture Design and implementation of the startup Farmy.ai's data pipeline.

Présenté par: **Ouaret Sami**

Soutenue le 27/09/2021 devant les jurys:

Président:	***Mr.Karim Lamraoui***	Université de BBA
Examinateur:	***Mdm.Bensefia Hassina***	Université de BBA
Encadrant:	***Mr.Brahimi Mohamed***	Université de BBA

2020/2021

Abstract

In this data-driven world, companies need to work on their data strategy to survive and stay competitive. Most existing data strategies depend on manual labor and minimize human creativity in problem solving and value creation. Because of this, it takes a long time to utilize data resources, making the insights gained obsolete. At **Farmy.ai** Startup, after a year and a half in production, we understood that implementing an automated data pipeline is imperative to accelerate the use of our data resources while scaling to new use cases.

This project aims to design a data architecture and implement a scalable data pipeline for the startup Farmy. This data pipeline is intended to enable image-based diagnosis of plant diseases. The implemented pipeline starts by regularly retrieving images and their metadata from social media and other sources. It then stores and catalogs the collected data in a cloud data lake. Then, it enriches the stored data with annotations from agriculture experts. Finally, we orchestrate all the operations of this data pipeline to avoid repetitive manual work.

ملخص

مع التطور التكنولوجي السريع يشكل الذكاء الإصطناعي والبيانات الضخمة القاطرة التي تقودنا لثورة من الحلول المبتكرة, لذلك تسعى المؤسسات جاهدة على نطاق واسع لدمج البيانات في أعمالها لتحسين منتجاتها وتجاوز منافسيها. تعد الزراعة تحديا جذابا للذكاء الإصطناعي و البيانات الضخمة, فارمي شركة ناشئة جزائرية تهدف لاستعمال الذكاء الإصناعي لتمكين الفلاحين من الحصول على تشخيصا سريعا و موثوقا لأمراض المحاصيل التي تقضي على جزء كبير منها سنويا.

نظرا لكون البيانات المحرك الأساسي للذكاء الإصطناعي, فارمي واجهت تحديات لدمج البيانات من مصادرها المختلفة حتى تتمكن من تطبيق الذكاء الاصطناعي, ولكون الكثير من البيانات تكون تالفة و بلا صلة لما تحتاجه فارمي, يجب تطهير وتصفية تلك البيانات, بعد ذلك حتى تكون البيانات ذات صلة جاهزة للإستعمال والتطبيق يجب إثرائها بمعلومات إضافية من طرف خبراء زراعيين.

من الواضح أن القيام بكل هذه العمليات يتطلب جهدا, لهذا قررت فارمي بناء نظام سلس يسمح لها بإدارة هاته العمليات دون عناء. يهدف هذا العمل إلى عرض تصميم يستفيد من بنية بحيرة البيانات والحوسبة السحابية لبناء خط أنابيب قوي وموثوق لتجميع ومعالجة بيانات فارمي لتمكينها من تطبيق الذكاء الإصطناعي والقيام بأعمالها.

Résumé

Dans ce monde axé sur les données, les entreprises doivent travailler sur leur stratégie de données pour survivre et rester compétitives. Néanmoins, la plupart des stratégies de données existantes requiert un effort manuel considérable et ne crée pas la valeur attendue. La startup Farmy, proposant des solutions basées sur l'intelligence, a rapidement réalisé l'importance d'une architecture de donnée robuste. Un pipeline de données automatisé permet également d'accélérer le développement de nouvelles solutions.

Le cadre de ce projet concerne la conception d'une architecture de données et l'implémentation d'un pipeline de données destinées à la startup Farmy. Ce pipeline de données est appliqué au diagnostic automatisé de maladies des plantes. La première étape consiste à récupérer périodiquement des images et leurs métadonnées à depuis les réseaux sociaux et d'autres sources. Par la suite, ces données collectées sont stockées et cataloguées dans un lac de données dans le cloud. Enfin, ces données sont enrichies par des annotations réalisées par des experts agricoles.

Dedication

I would like to dedicate this work to my family, first of all to my mother, the source of tenderness and love who has been by my side all my life.

Dear friends, teachers and all people who have contributed to the development of this work from near and far.

May Allah grant them health and happiness.

Acknowledgment

I would like to express my gratitude with these few lines

First, I would like to thank my thesis advisor **Dr. Mohamed Brahimi** for the patient guidance, encouragement, and advice he has given me. Those who were present and have always motivated me to complete this project with their support and availability.

Finally, we cannot complete this project without thanking the **Farmy.ai** team, especially Project Manager **Tarik Zedazi**, for the opportunity and passionate work environment that contributed to the success of this work.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context

The massive growth of data has increasingly affected organization worldwide. It has become a fundamental part of any successful business. Data is the cornerstone of decision-making processes and a key enabler for modern fields such as Artificial Intelligence, including machine learning, Deep Learning, which are widely used in many organizations. The use of these technologies leads to better results and higher efficiency.

The availability of data is not the only factor that leads to the success of these companies. It is the intelligent and efficient use of data that makes it valuable. Companies that seek alternative innovative approaches, solutions and technologies to deal with data in order to strengthen their competitive advantages are one step ahead of those that still use unplanned and traditional approaches.

With the emerging of IoT devices, 5G, drones and various new technologies, the potential of data generated from these materials has grown even further. A systematic method is required for such a revolution.

All fields are concerned with this leap of Big Data. Agriculture is no exception. It has gone through many phases over the centuries and has always been a critical area for people. Currently, agriculture is one of the hottest areas that is getting a lot of attention in the current technological revolution.

The term Smart Farming is usually used to talk about the technological impact on agricul-

ture. Smart Farming simply refers to the integration and management of various technologies and data to replace traditional farming approaches [1], such as real-time alerts, alarms, automatic intervention, automation of repetitive manual tasks in farming, automatic data collection, and assistance systems.

Agriculture is at the centre of a new era. We can have a revolution that would change the way we farm.

1.2 Pain points and challenges

One of the major problems that farmers usually face is plant diseases, which greatly affect the production of farms. Farmers find it difficult to get a reliable and immediate diagnosis that allows for quick intervention to treat and save crops.

Given the effort and cost required to reliably diagnose plant diseases, **Farmy.ai** is a startup registered in Algeria that seeks to develop AI and smart farming solutions to provide farmers with high-quality and rapid diagnoses. The goal is to provide farmers with the right information at the right time so they can make the right decisions.The business essentially relies on the integration of Big Data and Deep Learning to run its business.

For startups, innovative solutions are key, but efficient resource management makes them grow and persist. **Farmy.ai** is no exception, **Farmy.ai** is a data-driven startup, so building efficient, cost-effective data integration is a critical task.

No coincidence, a solid integration of data in business gives more leading competition and helps shipping successful products, for **Farmy.ai** startup to be up to deliver its solutions to its end users it faced one major problem which building a robust data integration platform that collects data from its data sources, to enable its Team to process data, do Machine Learning, Deep learning, and Business analysis, to get smart decision, and deliver reliable diagnosis to farmers.

For **Farmy.ai** Startup, providing reliable and quality diagnosis is a crucial thing. But it suffers from the slow exploitation of its data living in different locations (Data silos). regarding the slow process and manual collection of data, introducing additional data sources was a bottleneck, with time it became extensively hard and track data and get fast outcomes.

As a result, **Farmy.ai** Startup needs to design and implement a data integration system

to enable its business. As **Farmy.ai** depends on data to do Deep Learning, it must efficiently collect data from its different applications and sources, process and enrich that data, before it can enable data scientists to do Deep learning.

The primary step for Farmy.ai before doing Deep learning is data Annotation, to annotates data it needs to clean irrelevant data, then make it accessible to its data scientists. robustness and scalability of the process is a key, **Farmy.ai** needs to build a reliable system with painless infrastructure management to make data scientists focus more on the business.

1.3 Objective and contribution

To integrate its data from various sources, **Farmy.ai** ends up in need of building a data pipeline, to do automated data collection, easy data cleansing, and data annotation, to enables its Deep learning business.

As **Farmy.ai** depends mainly on images, we need to build a data pipeline that ingests Images with their related data from its data sources, manage the flow of those data to pass through data cleansing, data annotations, and make them available for data scientists and analysts.

Clearly, this data pipeline represents the building block of **Farmy.ai** business, but what is more important is to build a robust data management system that makes running and managing this pipeline painless, Therefore the major objective of this work is to use a Data lake architecture to design and implement **Farmy.ai** data pipeline.

Particularly, what makes really the target architecture practical is the ability to serve business objectives by providing robust and flexible data governance, and painless data processing. in addition we must:

- Ensure an efficient flow of data.
- Strive for cost effective system.
- Construct and run more data pipelines in the future.
- Enable quick and easy access to data respecting distinct types of users.
- Ensure high availability, and accessibility of data, and scalability of the system.
- Integrate well the data pipeline with the data lake architecture.

- Reduce human intervention with robust automation.
- The system should be future-proof by answering new questions about data.

1.4 Project Planning

This work describes our efforts to come up with a design and architecture to build a robust data pipeline for the startup **Farmy.ai**.

This work contains the first chapter that is dedicated to the state-of-art, the second chapter includes data architecture details, and the third chapter contains implementation, experiments, and major deployment details.

1. **Chapter 01**: The first chapter presents the current state of the art in Big Data and data processing. It aims to introduce some concepts of current architectures and existing solutions used in the project, such as data storage, data lake, data collection, data annotation and, more importantly, the concept of data pipeline, and describes the impact of cloud computing on data solutions.
2. **Chapter 02**: In this chapter, we present a detailed study and analysis of **Farmy.ai** requirements. It shows how we integrate the data lake architecture and data pipeline to collect, cleanse and annotate data.
3. **Chapter 03**: In this chapter we address the practical aspect of our project. We show the environment, infrastructure, software, tools, and techniques we use for the project, and how we use Amazon Web Services (AWS) as the infrastructure for our work. It includes a full end-to-end experiment of the Farmy.ai data pipeline with presentation of the results.

Chapter 2

Big Data Ecosystem

2.1 Introduction

Today, Data Science and Machine Learning are ubiquitous and have become a trending approach in the IT world as they generate tremendous business value. Companies are racing to outperform their competitors and stay ahead of the game.

Unfortunately, acquiring, managing and processing a huge amount of data, which we often refer to as Big Data, is not an easy process. With the advent of modern approaches, technologies and solutions, it has never been easier to build a reliable Big Data system.

Of course, there are various technical and business aspects to consider when developing a robust data integration system, as processing many data sources with different data types and formats is an additional challenge.

2.2 What is really Big Data

Governing relatively large data is not new, many big companies have been doing that for years so far, but the speed of generating data in the last decade is the most regarded point, this evolution has pushed researchers and engineers to develop and build new tools, and techniques to manage that massive volume of data.

Surely, hearing the concept Big data makes us question what characteristics and properties that truly define big data, the term big data is one of the most arguable terms in both

academic and industry fields, one short and expressive definition:

> Big data can mean big volume, big velocity, or big variety[2].

Another definition by The Publications Office of the European Union is:

> Large amounts of different types of data are produced from various types of sources, such as people, machines, or sensors. This data includes climate information, satellite imagery, digital pictures and videos, transition records, or GPS signals. Big Data may involve personal data: that is, any information relating to an individual, and can be anything from a name, a photo, an email address, bank details, posts on social networking websites, medical information, or a computer IP address[3].

Big data interpretations can vary according to the treated problem. Commonly, they define big data as the necessity to control and operate effectively large and various sorts of data generated at high speed to get better business value[4]. In addition, people usually refer to it as the complexity to manage and monitor the massive amount of data that traditional databases and systems cannot handle.

The crucial point, above all, is what we can do with this data. What value can we extract from it? [5] all this data should be well worth the effort we put in.

2.2.1 Big data V's dimensions

Regardless of the different interpretations and attempts to define Big Data by Researchers and Engineers, there are general properties that describe Big Data:

1. **Volume**: It is the most popular dimension that is closely associated with Big Data. It just defines the size of the data. How much data do we have[6]. The amount of data is gradually increasing and we expect it to grow massively in the coming years.
2. **Variety**: Data comes in different types and shapes. Variety describes the need to work with many kinds of data that come in various formats and from different sources.
3. **Velocity**: Velocity refers to the frequency or the speed of generating, processing, and analyzing data[6].
4. **Value**: We consider value as the most essential dimension in big data. It refers to the usefulness of data and how actionable data are.

5. **Veracity**: Unluckily, an extensive amount of data means we need to work with more distrusted or corrupted data. Veracity defines the level of trustworthiness and accuracy of the data[6].

2.3 Data Types and structures

Data comes in various structures and forms. To simplify the interaction with these types of data, we can identify the following types:

1. **Structured data**: A consistent model describes this type of data. It usually comes in an organized format as a tabular data (rows and columns) like the relational model, and it has a standard and deterministic way to define, query, and search.
2. **Semi-Structured data**: Semi-structured data is relatively organized data like structured data, but offers more flexibility. It gives some level of search and aggregation operations, but with some overhead and more complexity than structured data. Some examples of semi-structured data are XML files, JSON files, YAML files, and NoSQL databases.
3. **Unstructured data**: Unstructured data has no structure or model, making it the most difficult type of data to search and organize. However, it is considered the most valuable data. Some examples of unstructured data are videos and images.

2.4 Data Sources

The various data sources contribute a lot to the aggressive data growth. Data sources are places where an organisation can collect data. Every day, we generate nearly 2.5 quintillion bytes of data daily[7] from various data streams, such as IoT devices, search engines, social media, enterprise data, financial systems, and healthcare systems.

We can classify data sources into three different types:

1. **Machine generated data**: Machines and devices such as IoT devices, servers, smartphones, satellites, and sensors generate this type of data without the intervention of humans. This type of data is usually in a semi-structured format and is often available in real time.

2. **Human generated data**: Humans generate vast amounts of data in their various activities. Human intervention generates this data in a variety of ways, including speech, text, emotions, document files, spreadsheets, documents, writing, and speaking.
3. **Organization generated data**: There is no doubt that companies produce an enormous amount of operational and analytical data. Typically, organizations keep their data in separate locations as data silos. Data silos are a collection of data used by different applications and software and stored in isolated local areas [8].

2.5 Data collection

In a data-driven world, ensuring a reliable and smooth way to collect data efficiently with minimal information loss could be challenging, because of the heterogeneity of sources, and the diversity of data.

Data collection is the process by which we get data from sources to process it in real-time or store it for future processing. Data has influenced different aspects of business operations, growth, decisions, and strategies. Finding efficient ways and techniques to collect data is inevitable to build a good data strategy.

According to the source of data, used methods and techniques to collect data may vary. Some companies ask for data from users, others track user activities, others use API provided by social media. We cam see Web scraping as a data collection method, regardless of the controversial debate about the ethical issues with it.

2.5.1 Web scraping

Social media, websites, and blogs represent human-generated data sources. Millions of transactions take place every day, people can initiate transactions by transferring money, buying goods online, paying for services, etc. According to a report by Statista, in 2020, about 3.6 billion users[9] use social media 145 minutes per day[10]. Usually, companies try to get data from these platforms to help them gain business insights for their various activities.

Web scraping is a process by which we can collect and store data from websites in order to process it and extract valuable information from it. Typically, we scrap data that changes frequently, is not provided through an API, is only publicly available, and for which we have access permission, otherwise illegal and ethical issues may arise.

Data Scraping can improve data collection. It is a systematic approach to collect large amounts of organized and frequently updated data. A good investment in data scraping can save a lot of money and resources.

2.6 Data storage

One significant component of any Big Data system is a reliable storage. Data storage has gone through several phases over the past few decades. Flat files have played an important role in various businesses. Unfortunately, depending on them for data storage is not a reliable method. Databases, especially relational databases, have been the salvation from the chaos caused by files.

Databases, warehouses, and No-SQL

Relational databases have been the first choice for storing and managing data for decades. We typically use them to store relational, operational, and transactional data. However, the advent of data warehouses has been a game changer for enterprises to store analytical data and perform analytics at scale.

A data warehouse is a central repository that merges data from many sources. It stores columnar rather than transactional data, which gives it powerful and fast query and analysis capabilities.

Data warehouses are optimized for data-driven organizations to analyze and extract useful information optimally [11]. It enables the storage of historical data for reporting and analysis to increase business value.

NoSQL databases are another alternative flexible way to store data in relational databases. It allows for dynamic data schema by allowing access and management of data using a range of data models [12]. Notably, it can handle large workloads, large volumes of data, and low latency service requirements.

Hadoop Distributed File System(HDFS)

Databases and data warehouses in particular are designed for a specific purpose only, to store operational and analytical data with limited processing capabilities. Hadoop Distributed File

System (HDFS) has been a new generation of data storage and processing and has enabled a new way of handling data at large scale.

HDFS is a highly fault-tolerant and distributed file system that provides fast access, low cost, and high throughput, and enables the storage of a large amount of data sets of different types on different platforms [13].

A major drawback of HDFS is the need to manage and provision servers, also the processing capabilities are tightly coupled with storage.

Object based storage

Unfortunately, despite all the many capabilities of HDFS, but it is tightly couple processing with storage which adds extra overheads to span and manage clusters. To overcome this problem, object storage appeared. Object storage is an architecture to store an extensive amount of unstructured data as objects accessed via API, like HTTP API.

Distinctly, each object is independent, self-contained, and holds its own meta-data. Object-based storage offers high availability, and it scales independently from the processing tier.

Cloud providers offer this kind of object storage[14] as SaaS with multiple tiers that are optimized for different cases like backup, and archive.

2.7 Data Cataloging

Data cataloging defines a process to centralize all meta-data related to an organization's data within accessible storage called Data Catalog, and Combine it with search capabilities and quality management [15].

Meta-data is the data that describes and adds context to actual data. The major role of a data catalog is automating data finding and searching by storing and tracking its Meta-data. It permits consistent data usage and prevents wasting efforts when working on the same data by separate teams within an organization.

Obviously, a data catalog adds more trust to organization data. It allows for a self-service model. A self-service is the ability for organization stakeholders to find, and process data without having to manually search of the data.

2.8 Data Processing

Data processing is the operation of converting raw data into actionable and meaningful data. It could be a transaction, a data preparation, data conversion, aggregation, or any kind of transformation or manipulation. The output of this operation should help to generate business value.

Processing large amounts of data is not an easy task. Databases have some built-in processing capabilities that are scalable to a limited extent. Databases are used for storing operational data rather than analytical data. On the other hand, warehouses are used to store and query analytical data.

Hadoop was the first real scalable storage that allows for highly parallel and distributed processing through the map-Reduce model, primarily it allows to define a map function that processes a key-value pair to generate a set of intermediate key-value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key [16].

Hadoop depends on Map-Reduce model to allow for massively parallel processing of vast amounts of data running on large clusters [17].

2.8.1 ETL processing

In data processing design patterns, ETL is the most known method to process and move data. ETL does not depend on specific tools. It contains three primary steps, the first step is extracting data from its data source, the second step is transforming retrieved data into usable data. At last, we load transformed data into a final destination[18].

ETL is widely used in data processing systems, it is a fairly simple process that allows for predictive use of system resources and many tools has built-in support for ETL process.

One drawback of ETL is it permits only a onetime processing over the extracted data, it also introduces some bottlenecks to scale.

2.8.2 Lambda architecture

The Lambda architecture is more a general design of big data system, mainly it offers two primary channel for data processing. A stream processing channel that allows for real-time processing of the data, and the batch layer which ensures the consistency of the data by

doing a full calculation over the received data [19]. This architecture is relatively flexible since it provides both stream and batch processing, it offer speed and reliability, but it has a problem of code redundancy to handle both scenarios (stream and batch) which introduces some bottlenecks during maintenance and migration [19].

2.8.3 Kappa architecture

The Kappa architecture reduces the complexity of Lambda architecture by removing the batch layer and keeps only the stream layer, which increases the performance of the system [20]. Although we should see these architectures as a complementary way for designing a big data system and not as an alternative to Lambda. Kappa is suitable when the scenario it heavily depends on real-time data with no batch processing.

2.9 Data Lake architecture

A Data Lake is a central repository that stores raw data of different types. It leverages the disk's lower cost advantage, and distributed storage to offload capacities from data warehouses and databases[21]. It only stores data without a pre-defined model, so we can access it later in an on-read schema[22].

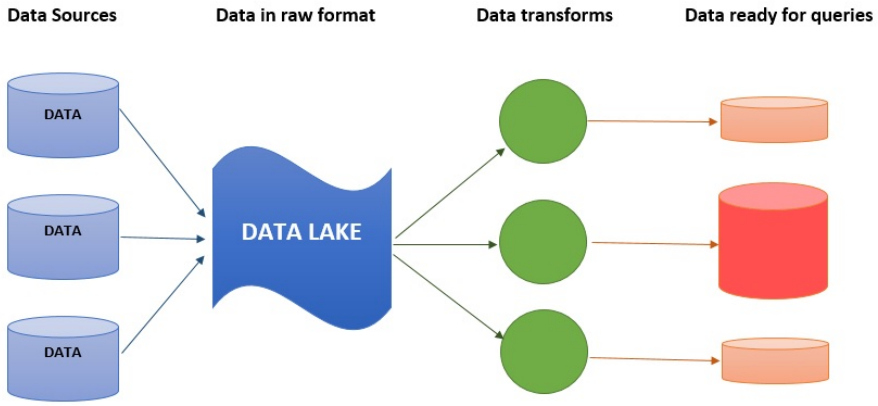

Figure 2.1: Data Lake Architecture
[23]

A Data Lake is not a Data warehouse, Data Lake dynamically allows for on-read schema but data warehouse restricted on-write schema makes to it not flexible compared to the data

lake schema. Table 2.1 shows a comparison between data lake and data-warehouse:

Table 2.1: Data Lake and Data Warehouse Comparison

Data lake	**Data warehouse**
It stores all raw data of all types.	mostly enhanced for relational structured data and columnar data.
We may use it for Machine learning, data discovery.	Enhanced for analytics and visualization.
It offers lower costs with relatively faster queries.	Provides faster queries with higher costs.
Requires up-to-date catalogs for better queries.	It has powerful querying capabilities.

2.10 Data pipelines

ETL is the common approach to integrate and process data, unfortunately, it struggles poorly to scale, an alternative approach is to use a data pipeline which is an architecture to organize data-life cycle including data ingestion, data processing, and data storage[24].

While ETL represents an abstract framework for designing pipelines, a data pipeline is a complete system, it may follow ETL as well as it could just move data with no transformation. A well-architected data pipeline extensively reduces human intervention. Particularly, data pipeline ensures an efficient flow of data.

Data-driven businesses widely adopted data pipelines. Data pipeline is more about the data movement either from one phase to another or between one or more different systems, typically it moves from data sources, passing through various interconnected tasks until it reaches its last destination (data sink)[25].

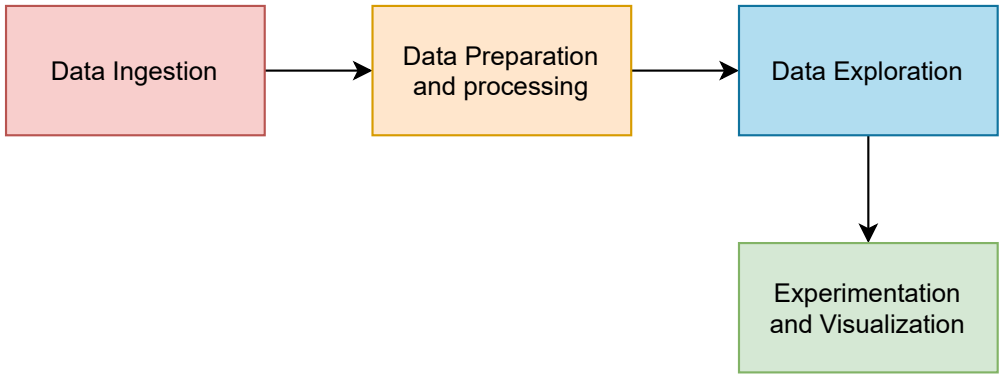

Figure 2.2: General life cycle of data

In particular, practically building and managing a data pipeline is complex, as it requires the administration of well-maintained infrastructure, integration of heterogeneous storage solutions, managing communications, connections, additionally handling various errors and issues raised by processing tasks, and traceability of executed tasks.

Despite of all challenges and bottlenecks to build and develop a robust data pipeline, with wise tools choosing, solid infrastructure management, and leveraging best software engineering practices, building data pipelines could be a moderately manageable task to do.

Ultimately, a well-architected data pipeline should permit for fully automated data movement, including data extracting, validating, transforming, data analysis, and data visualization[26].

2.11 Data Annotation

While ETL and data pipelines describe how to do data processing, data annotation describes what to do with data, Data annotation is adding labels and semantics to data so we can use to train machine learning models [27, p. 193].

The main power of a high-quality data annotation is accurate human intervention, choosing highly skilled experts in the target domain is critical, for instance, labeling data for sentiment analysis needs skilled psychology specialist, although some labeling may need less expertise, it is still a good practice to strive for more skilled people to ensure the quality of data annotation.

Data annotation can vary depending on the type of data. For instance, it could image annotation, text annotation, audio annotation, video annotation, etc.

In addition, we need to prepare and set up a stable environment for annotators with robust labelling tools, techniques, to ensure a quality data annotation, as humans are the keystone of the process, a healthy human resource management is required [27, p. 193].

2.12 Big data on The Cloud

The advent of cloud computing has drastically changed the way Big Data is handled. It has opened up completely new dimensions for different businesses and the way we develop systems and software. Big Data has benefited greatly from this advent.

Cloud computing virtualizes infrastructure as utilities delivered on demand over the Internet instead of setting up complex hardware and software. It provides highly decoupled services that remarkably enable resilient and scalable Big Data solutions with little effort and in a cost-effective manner.

In the era of cloud computing, building data pipelines has never been easier. With a well-architected design developed and deployed in the cloud, cloud providers offer access to a wide range of services for various operations, such as storage, computing, and clusters.

Importantly, they also provide enough flexibility and control over the infrastructure to create a highly customized solution for most business problems. They also offer a wide range of managed services for rapid business development.

2.12.1 Serverless computing

Clearly, setting up infrastructure, running applications and servers is overwhelming, due to the continuous management and patching of servers, operating systems, upgrades, and updates , etc.

The emergence of serverless computing has hardly affected the enterprises in the industry. The serverless model eliminates the need to manage infrastructures and computes, it allows compute to be used only for the amount requested, without having to provision or manage servers [28], and without additional costs or requirements.

Most often, cloud providers offer serverless computing as a service. A particularly common service is Function as a Service (FaaS), which allows a portion of code to be executed on demand and we only pay for execution time[29].

2.13 Conclusion

In this chapter we have looked at Big Data, its controversial definitions and its dimensions. We have talked about the different types of data and some of their examples.

We have looked at data collection and the different sources of data, we have explored different storage solutions and design patterns for integrating and processing data, and we have made a brief comparison between data lake and data warehouse.

We also talked about data pipelines, the process of data annotation, and finally the leap of cloud computing in creating data-centric solutions.

Chapter 3

Farmy.ai Data architecture

3.1 Introduction

In this chapter, we show the analysis and design of the **Farmy.ai** data integration system to set up and run its annotation pipeline to enable its business.

This part covers various components of the system, decisions and concepts that ensure the flow of data within **Farmy.ai**. It also includes a brief description of **Farmy.ai**'s data-centric business and its specifications.

We strive to describe the design and approaches for a cost-effective and efficient data pipeline running on a data lake architecture. We show comprehensive details of the components of **Farmy.ai** Data-Lake.

We cover the process of data annotation that essentially drives the data pipeline, starting with ingesting images from various sources, processing and annotating images, and finally delivering the data to its final destination so that it is available for processing and ready to be used to extract, visualize, and provide needed annotations for Deep Learning.

This type of platform provides an efficient way to enable scalable data flow that allows **Farmy.ai** to manage data efficiently and, in particular, enables Data Scientists to run productive Data Analysis and Machine Learning at scale.

3.2 Data pipeline description

Ensuring efficient and smooth data flow is a critical task that requires a lot of effort and attention. In this part, we would like to present the architecture we used to design and build a reliable, loosely coupled, and service agnostic data integration system.

Farmy.ai needs to ingest and collect images with their metadata from various data sources in order to run its business. This process involves cleaning and annotating images before we can use the results for Deep Learning.

Although the image cleaning and annotation pipeline is the main goal of this project, there are other motivations and requirements for the target system to allow Farmy.ai to create and run other data pipelines for Deep Learning and data analysis.

In summary, we can divide the characteristics of this data integration system into 2 main categories: business requirements and technical requirements.

3.2.1 Business requirements

Business requirements are the first and only motivation for building a new system. They define the features and functions required to make the business run.

Since the system should be extensible to take on new data processing tasks, we specify some functions for the system in advance. Here we list some of the most important ones:

- Eliminates manual intervention as much as possible.
- Is extensible to handle more data sources in the future.
- Help us clean up and label images effortlessly.
- Enable different users to find the data they need quickly and efficiently.
- Improve architecture for robust data processing and efficiently handle more data pipelines in the future.
- Consume data continuously for Deep Learning and ensure the correctness and reproducibility of the data pipeline.

3.2.2 Technical requirements

Technical requirements are more about properties, principles, and tools used to develop the system. There are certain technical requirements, of which we list some important ones:

- Efficiently ingest in various types of data.
- Enable modular, controllable, and reusable data ingestion.
- Build a maintainable and modular system.
- Debugging and testing the system should be relatively easy.
- Enable strong data governance and ensure high data availability, security and accessibility of all assets.
- Ensure the security and integrity of the data against various processing and operations.

3.3 High-level overview of the system

Before examining the various components of the system, it is worthwhile to gain an overview of the system to capture the various requirements, actors, and components of the **Farmy.ai** platform.

The first thing we should discuss is that the **Farmy.ai** startup contains many services that have full control over their functionalities and are the main data sources. The incoming data comes mainly from the mobile application and from web scrapers that feed Deep Learning with images and some metadata.

The incoming data is usually primarily images accompanied by some metadata in JSON format. We regularly ingest this data into the data platform by retrieving the data in scheduled batches, validating it, and then storing it.

After a successful data ingestion, we clean the ingested data. **Farmy.ai** team members help in this phase. **Farmy.ai** plans to incorporate Machine Learning to automate this phase, but the process still requires some level of human intervention.

The third part represents the first phase, which essentially requires complete human intervention, mainly Agriculture Experts annotate relevant images with various labels. This process may require one or many different annotations depending on the requirements.

After completing the annotation phase, we release the results for consumption and processing, load them for experiments, analyze them, load them on dashboards for visualization, and essentially train the data using Deep Learning.

To ensure the proper flow and execution of these tasks, some synchronization and orchestration is required. This process logically extends horizontally across all phases of the system.

Putting all these phases together, we get the **Farmy.ai** image annotation pipeline.

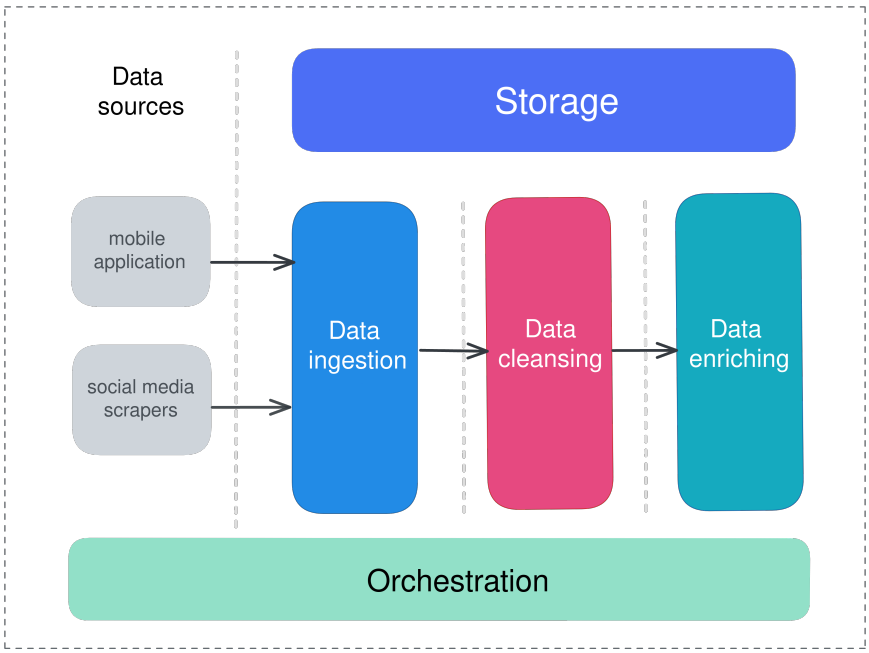

Figure 3.1: High level overview of Farmy Data Integration System

3.4 Farmy.ai Data Sources

Farmy.ai depends primarily on images to do its business. Most importantly, the images have associated metadata that makes them understandable. All incoming data comes primarily from the mobile app and web scrapers.

3.4.1 Farmy mobile Application

The mobile application of **Farmy.ai** is an important source of data. It allows farmers to upload photos of plant diseases and get quick diagnoses in response to their posts. We would

like to discuss this application to understand the context of the data used in the **Farmy.ai** data pipeline.

The application allows farmers to upload images with a description and detailed information about their infected plants. When farmers upload images, they receive a disease diagnosis from an agricultural expert. The data resides in their local database, and the images are stored in their private Object-Storage. So when the data pipeline runs, it consumes data from the application database and Object-Storage.

3.4.2 Social media scrapers

Before we developed this type of application at **Farmy.ai**, this process was done manually. It was a slow and time-consuming task that created a bottleneck in data collection, and to eliminate these problems we developed Web Scraper.

Web scrapers automate the process of image collection. These are primarily scripts that are executed at a specific time or on demand to crawl data and images in batches from social media websites. Web scrapers feed the data pipeline with images and metadata.

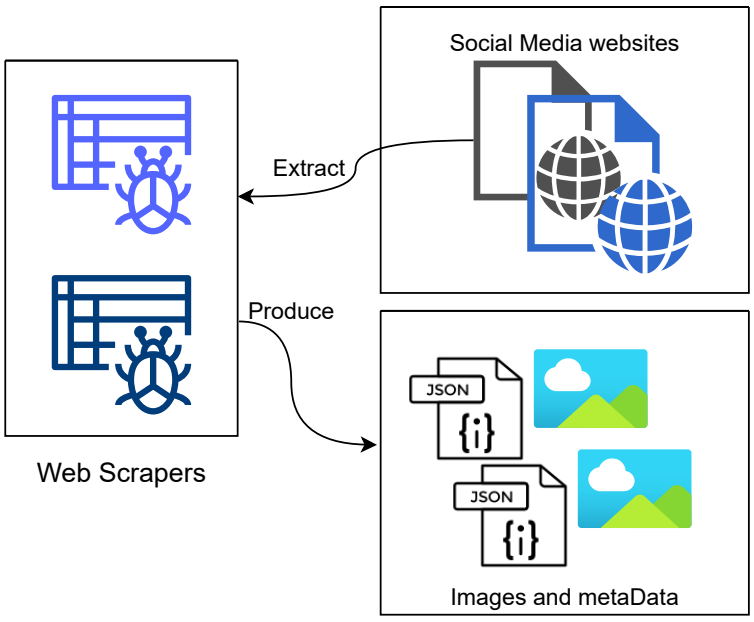

Figure 3.2: Basic workflow of Farmy.ai web scraper

3.5 Farmy.ai Data Lake Architecture

Typically, data pipelines need to move data from one phase to another, interact with databases, communicate, transfer data, or store intermediate results during some processing jobs.

We use a data-lake architecture as the building block for constructing the **Farmy.ai** data pipeline. The data lake represents the central repository where raw data and processing results are stored. We use object-level storage to build an organized and highly available scalable data integration system.

The motivation for choosing a data lake is not only that it is a resilient and persistent storage, but also that it provides flexibility in terms of the data stored and the type of data. When we ingest data from the **Farmy.ai** data source, we can easily store everything in its original format.

Data Lake allows us to essentially not think about the processing required until we need to interact with the data. It allows us to use the on read schema when dealing with the data. This allows us to run our data pipeline multiple times on the same data, reducing the overhead and effort required to process the data.

The Data Lake is at the heart of **Farmy.ai** image annotation pipeline. Developing such a system was not a clear path from the start, given the requirements and concerns of developing a robust, scalable, efficient, and extensible system.

3.5.1 The Data Lake constraints

There are some constraints and rules when building the **Farmy.ai** Data Lake. Our goal was to maintain and ensure a low-coupled data lake with external components:

- The Data Lake should serve as a central point for access, management, auditing, and provisioning.
- Separation of concerns and responsibilities, ingestion of data, use of storage, processing, retrieval of data are all independent of each other, we just aim for few low-coupled layers.
- Ensure accessibility and searchability of datasets for approved users.

Data lake security is an important task. A complete Data Lake must include a security layer that ensures data integrity, and confidentiality, a flexible authorization system, and support for identity access management, data replication, data versioning, data retention policies, and encryption.

3.5.2 Farmy.ai's Data lake zones

Some of the key principles during the architecture process were to divide the data lake into zones. There are three primary zones: the raw data zone, the staging data zone, and the curated data zone. Each zone corresponds to a specific use and has a different level of access. Figure 3.3 shows the Basic Farmy.ai Data lake architecture.

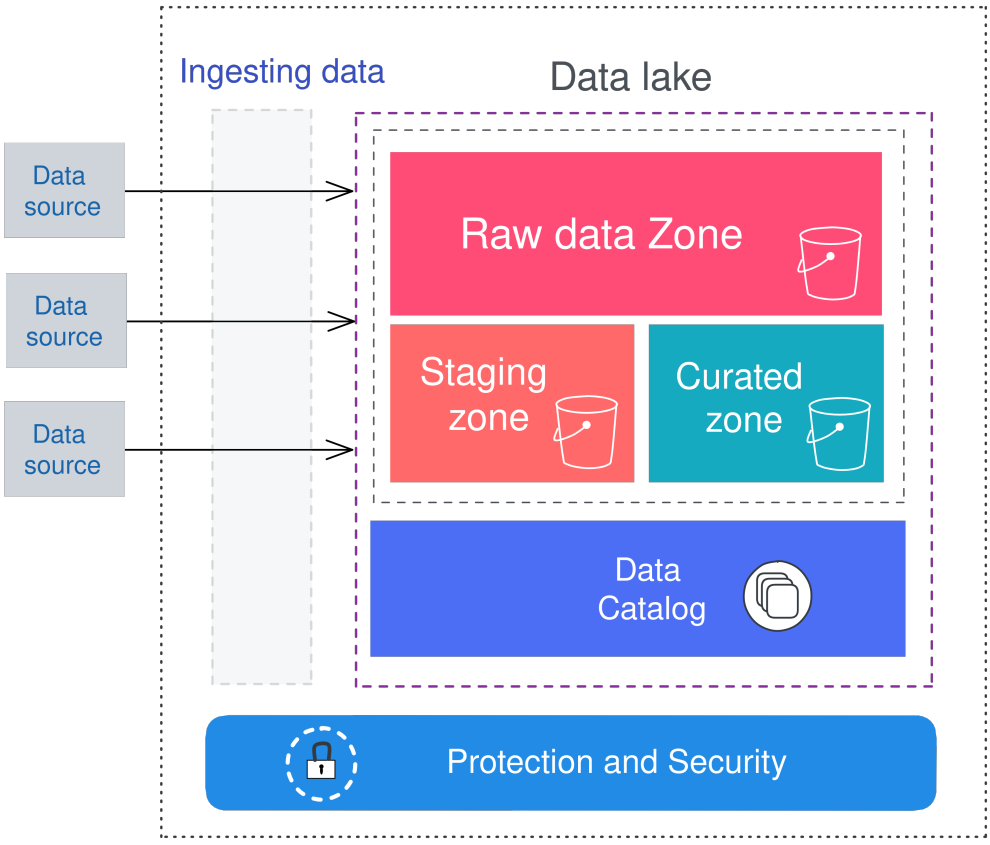

Figure 3.3: Basic Farmy.ai Data lake Architecture

Raw data zone

This zone stores only the incoming raw data from the data sources in their native format. Since few know how to handle this raw data, access to this zone is only accessible by engineers and developers, and with limited access (temporary) for cleansing and data annotation application.

Staging data zone

The staging zone stores data ready for processing, intermediate processing results, and some other data under review. Raw data is usually moved to this area or is associated with data stored in this area.

The data pipeline stores temporary data in this zone. Therefore, the staging zone primarily acts as a transition station from the raw data zone to the curated zone. Processing jobs (data cleansing and labeling application) have access to this zone.

Curated data zone

This area stores the curated data that is ready to consume in experiments, Machine Learning (Deep Learning) training. Hence, this area is accessible for data science, data analysis.

3.5.3 Farmy.ai Data lake storage

Farmy.ai is highly dependent on images as the main data type. Choosing a storage that ensures durability, high availability and security is the foundation of the **Farmy.ai** Data Lake. A local environment like Hadoop Distributed File System (HDFS) with coupled compute capacity is not an ideal choice for a decoupled system and a startup with small resources.

The **Farmy.ai** Data Lake uses AWS Cloud Simple Storage Service (S3) as its main storage, a Object-Level storage that offers powerful features such as consistency, high scalability, simplicity, lower cost, and robustness without managing servers.

The Object storage allows us to divide our data lake into zones, where each zone corresponds to a **Bucket**, which is simply an organized container of data.

3.5.4 Data ingestion Layer

The ingestion layer manages data extraction and uploads from **Farmy.ai** data sources and provides a flexible and convenient way to interact with Data Lake. It also validates the ingested data and data sources.

It acts as a gatekeeper for incoming data. It routes the data to the appropriate location in each zone of the data lake. Whether the data is coming in batches or streams, this layer should be able to handle it. In particular, it should ensure stable loading of the data into the Data Lake.

Of course, the ingestion layer knows how to connect, extract images and their metadata from **Farmy.ai** data sources. The ingestion layer uses object-level storage capabilities and an identity access management system to ensure its functionalities. **Farmy.ai** mainly depends on cloud services.

To ensure clean and maintainable ingestion and avoid the complexity of dealing with mobile app databases, storage, and web scrapers on a regular basis, we delegate the ingestion task to connectors that act as proxies to our data lake.

Connectors perform the communication between Data Lake and the data sources. Connectors are mainly reusable libraries that provide an interface for common data ingesting functions. Figure 3.4 depicts the ingesting process.

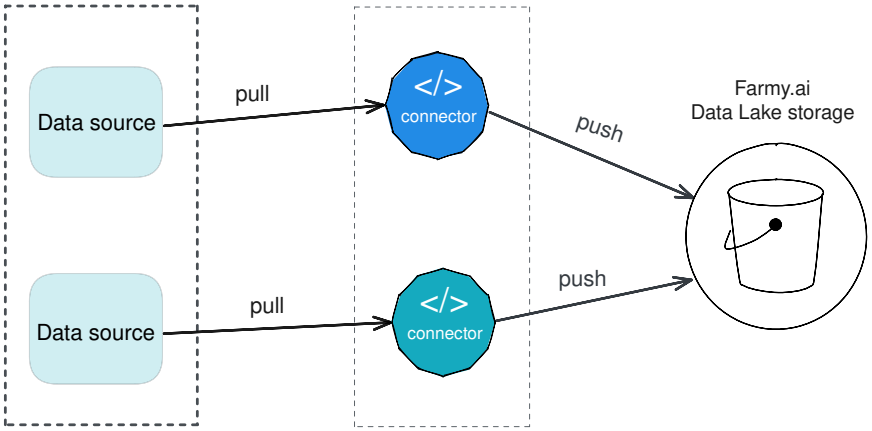

Figure 3.4: Farmy.ai Data Lake's ingestion layer

In particular, a connector exposes its functionality to its users as an API encapsulated in a library, abstracting from the details of the ingestion layer. It uses the storage API and

the data source API to provide us with a painless and effortless data ingestion. Figure 3.5 shows the Basic Class diagram of **Farmy.ai** connectors.

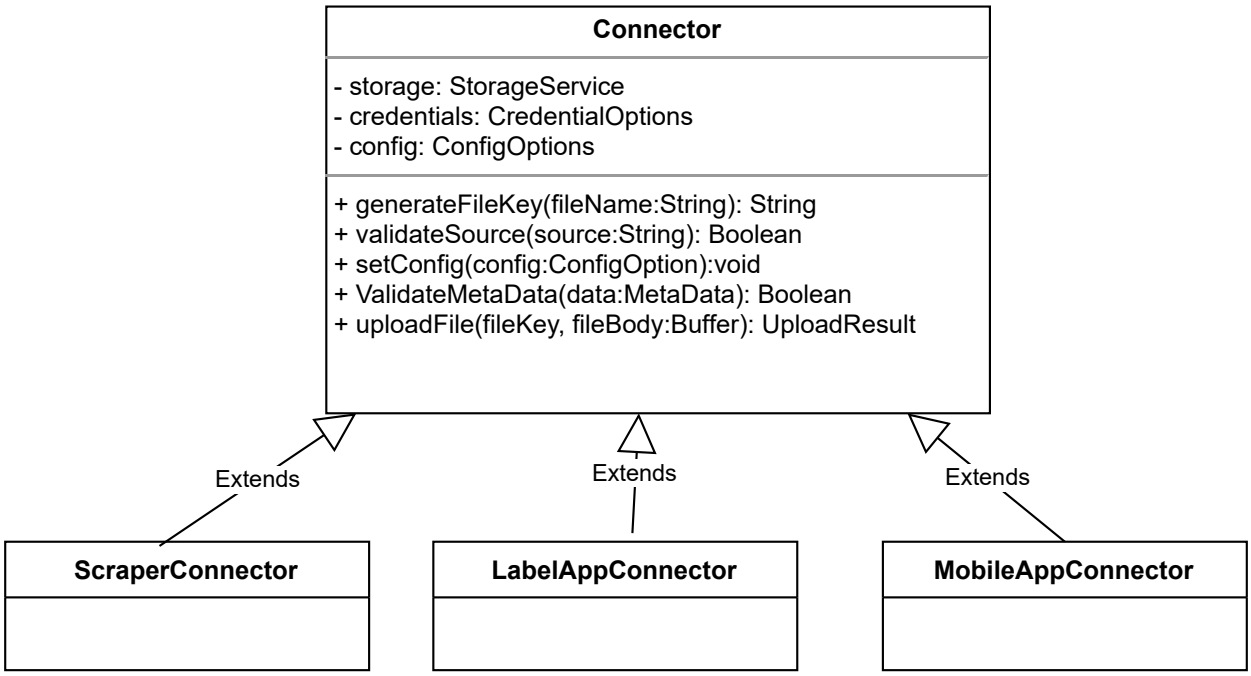

Figure 3.5: UML class diagram of connectors

Other advantages of using data connectors include the ability to add more connectors, refactor connectors and maintain compatibility without refactoring code in other parts of the data pipeline, and a modular approach that leads to code isolation, ease of debugging, and most importantly, testability.

3.5.5 Farmy.ai data lake catalog

Over time, as we add more data to the data lake, and since most **Farmy.ai** data is unstructured, like images, the data lake lacks a defined way for searching and retrieval, and especially after the growth of the stored data, it becomes really difficult to keep track of all the data stored, we call this problem Data Swamp[30].

We need a mechanism to overcome this dilemma. **Farmy.ai** data lake maintains a data catalog. The data lake's data catalog is the source of truth that allows us to track all of our data lake's assets and compensate for the lack of structure in the Farmy.ai data.

The **Farmy.ai** data catalog is a database that tracks and stores information about all

ingested data. It has answers to the actual data stored by storing important metadata about all assets such as data source, data type, and owner of the data.

The data catalog makes the data lake storage queryable and accessible, it gives **Farmy.ai** a broad overview of the stored data. Usually we call a data catalog within a Data Lake a comprehensive or index data catalog[31].

Constructing the catalog

The object storage propagates some events during the life cycle of its resident objects (uploaded files). Setting up a handler that listens for triggered events when a file is uploaded or updated allows us to get the metadata of the uploaded objects and add them to the comprehensive catalog, keeping it up to date. Figure 3.6 shows the process of maintaining the **Farmy.ai** comprehensive data lake catalog.

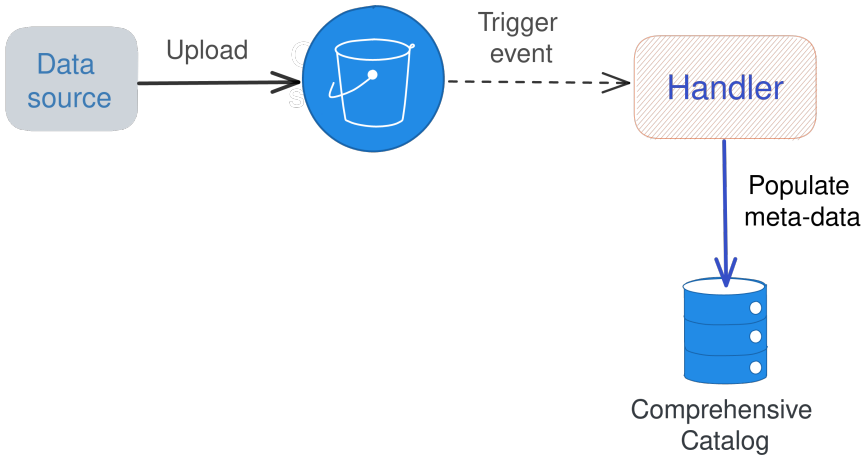

Figure 3.6: Constructing The Comprehensive Data Catalog

In **Farmy.ai**, the compute handler is a Function as a Service (FaaS), which is simply a piece of code that executes when a Ingested Object event is triggered to update the catalog, and then shuts down, there are no servers to manage which reduces considerably efforts and cost.

The data Catalog data

The comprehensive data catalog contains all the necessary fields and information for easy and quick access to data-sets, it has enough information to analyze and monitor the data-

sets, Figure 3.7 represents an experimental schema we developed while building the data lake for the data catalog **Farmy.ai**.

The main field is the primary key *file_type* which represents the subject of the file, a file identifier field *file_Key* represents a unique generated key for the file, *timestamp* represents the timestamp when we index the file in the catalog.

More about the fields in the catalog: The *source* field indicates where the file came from. Occasionally, when new changes are made to the schema or content of files, *version_id* contains the version of the file to keep track of the different variants of the file.

Table	
PK	**File_type**
FK	**RelatedTo**
	file_key: string
	timestamp: Date
	source: string
	location: string
	ContentType: string
	owner_id: string
	user_id: string
	version_id: string
	meta_info: map

Figure 3.7: The Data Lake catalog schema

We also store some other fields like the *location* field which describes the location of the file (bucket), the *owner_id* is the owner of the file, *user_id* is the user who caused the events.

3.5.6 Farmy.ai Data lake Query layer

As the data is available, stored securely in our data lake, and the data catalog is up to date, we naturally want to query or process our data sets. The query layer serves these purposes and allows users of varying technical levels of **Farmy.ai** data lake to query and explore the data.

The query layer depends on query engines, which function in some ways like connectors but serve a different purpose. They decouple direct programmatic query and processing access to our data lake. Similarly, we can add query and search capabilities to our Data Lake by connecting compatible search services to the Data Lake storage.

The query layer also allows us to enforce and guarantee read-only queries to users or, masking some parts of the data. We can also integrate search and processing tools as needed. Figure 3.8 shows a Basic UML class diagram of query engines.

Since we insist that there is no direct interaction with our data lake, we add pre-configured libraries to the data lake storage and data catalog that act as query engines for common queries and help us build and run the data pipeline.

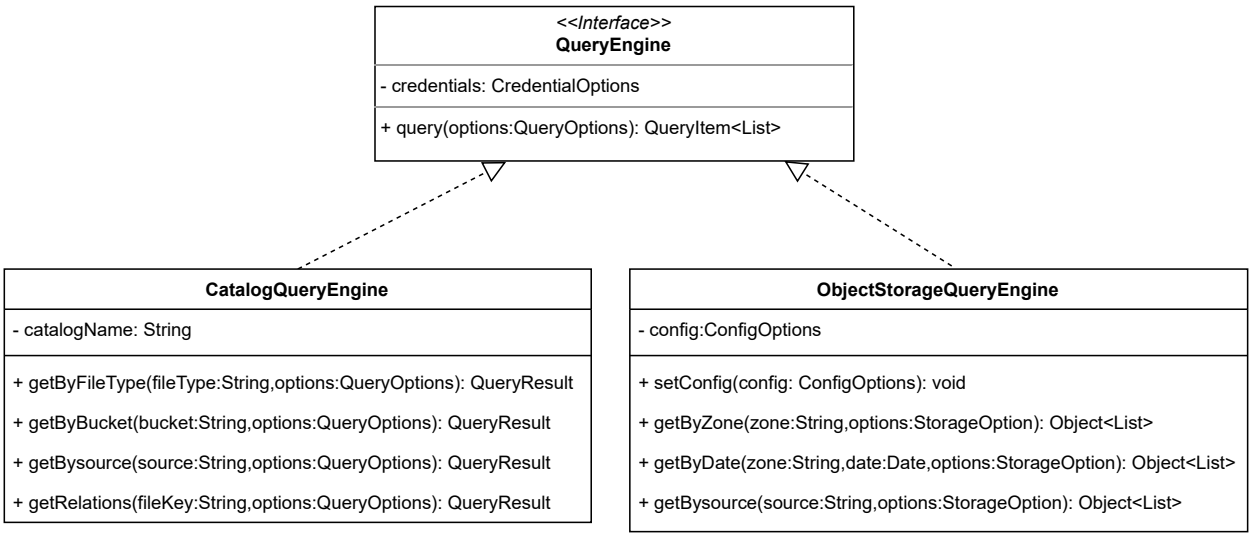

Figure 3.8: Query Engines UML Class Diagram

Query engines, whether libraries or search services, enable **Farmy.ai** data pipeline phases such as image cleaning and annotation, **Farmy.ai**'s Deep Learning's training, analysis, and other experiments to gain insights and retrieve data as needed with minimal effort.

3.5.7 Packing the data lake

When we combine all the Data Lake components, we get a complete picture of the fully functional version. Figure 3.9 depicts the **Farmy.ai** data lake architecture, including the ingestion layer with connectors, the query layer with query engines, Object-Level storage, and the comprehensive data catalog.

For completeness, we show the security layer, which includes various tools, policies, and best practices to enable a well-functioning data lake.

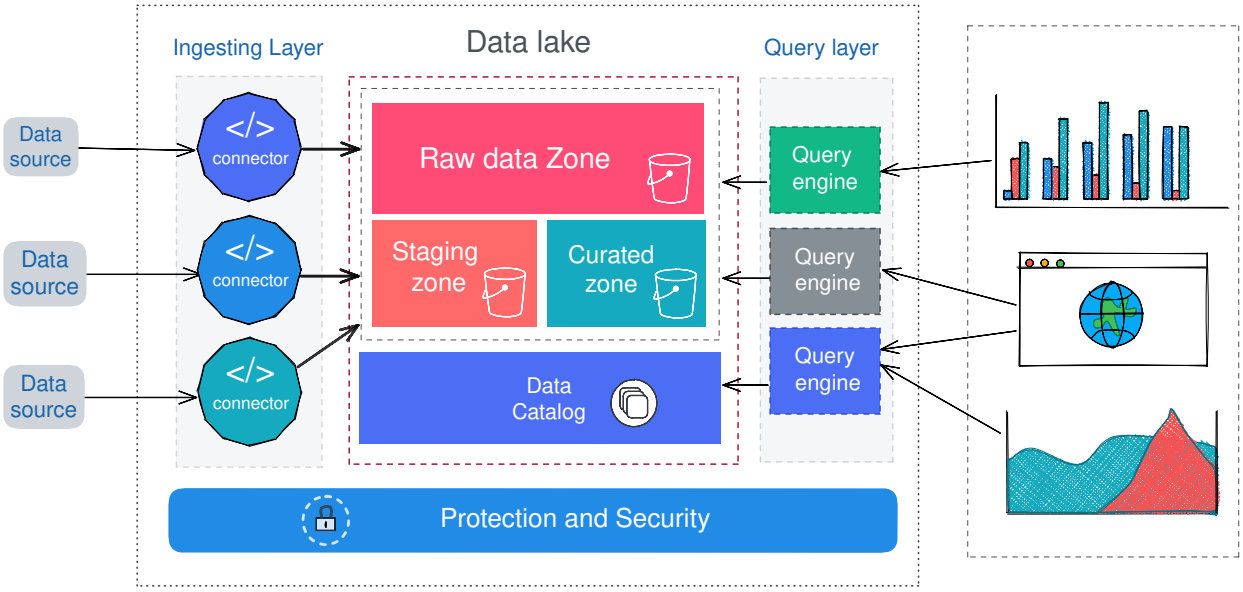

Figure 3.9: The complete Farmy.ai Data Lake architecture

3.6 Farmy.ai Annotation Data pipeline

3.6.1 Annotation Process

The **Farmy.ai** annotation process begins by selecting ingested images from the Data Lake storage. Usually, the ingested data contains irrelevant images for plant diseases. To filter out these irrelevant images, we need a mechanism that examines newly received raw data and removes these irrelevant images.

Non-expert people examine selected images to clean up irrelevant images. We clean the data by accepting or rejecting the examined data through a web application. In this process,

tags and labels are added as metadata to each task related to an image.

Next, two different agricultural experts (agronomists) examine the relevant images and annotate them. This step may take some time. We assign annotation tasks to each of them in a completely separate project. An annotation application essentially visualizes the relevant images for them.

Figure 3.10 represents a UML use case for images cleansing and annotating images.

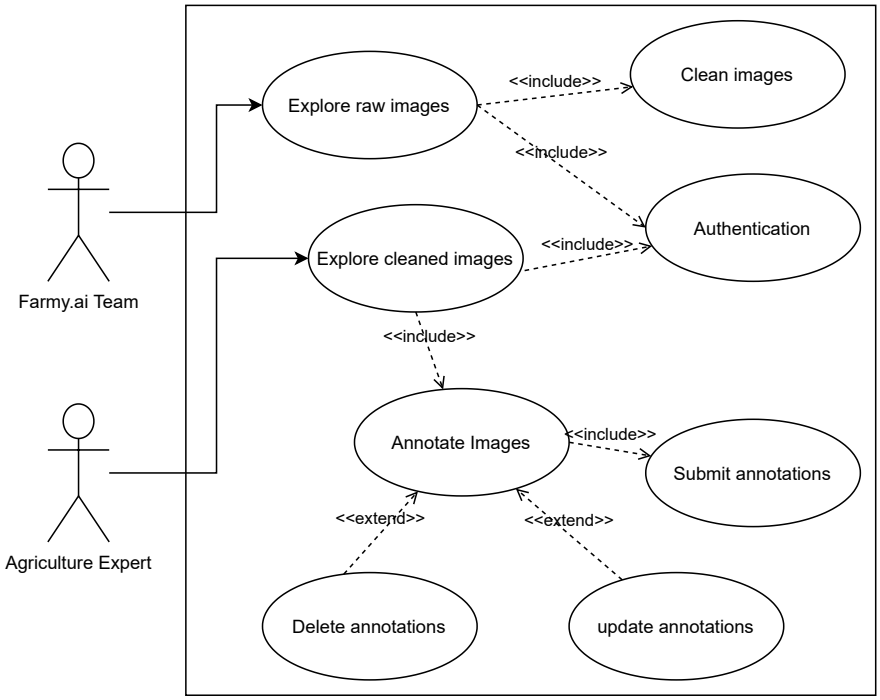

Figure 3.10: UML use case for cleaning and annotating images

The annotation process is quite simple. An authorized Agriculture Expert navigates through the accepted images via the annotation web application and then continuously annotates them with labels, classifications, and other types of comments.

Since two different experts do the image annotation, there is a possibility that different annotations will be created for the same image. We need to resolve the conflict, a third experienced agronomist resolves this conflict. Ultimately, annotation should be ready for use in Deep Learning for plant disease detection.

The annotation application allows us to visualize our data and images by allowing us to assign tasks for annotators to work on. It supports various types of annotations, including

image annotations.

We call any complete annotation process an annotation round. It starts with loading the ingested images and ends with exporting all created annotations. Figure 3.11 depicts the annotation process.

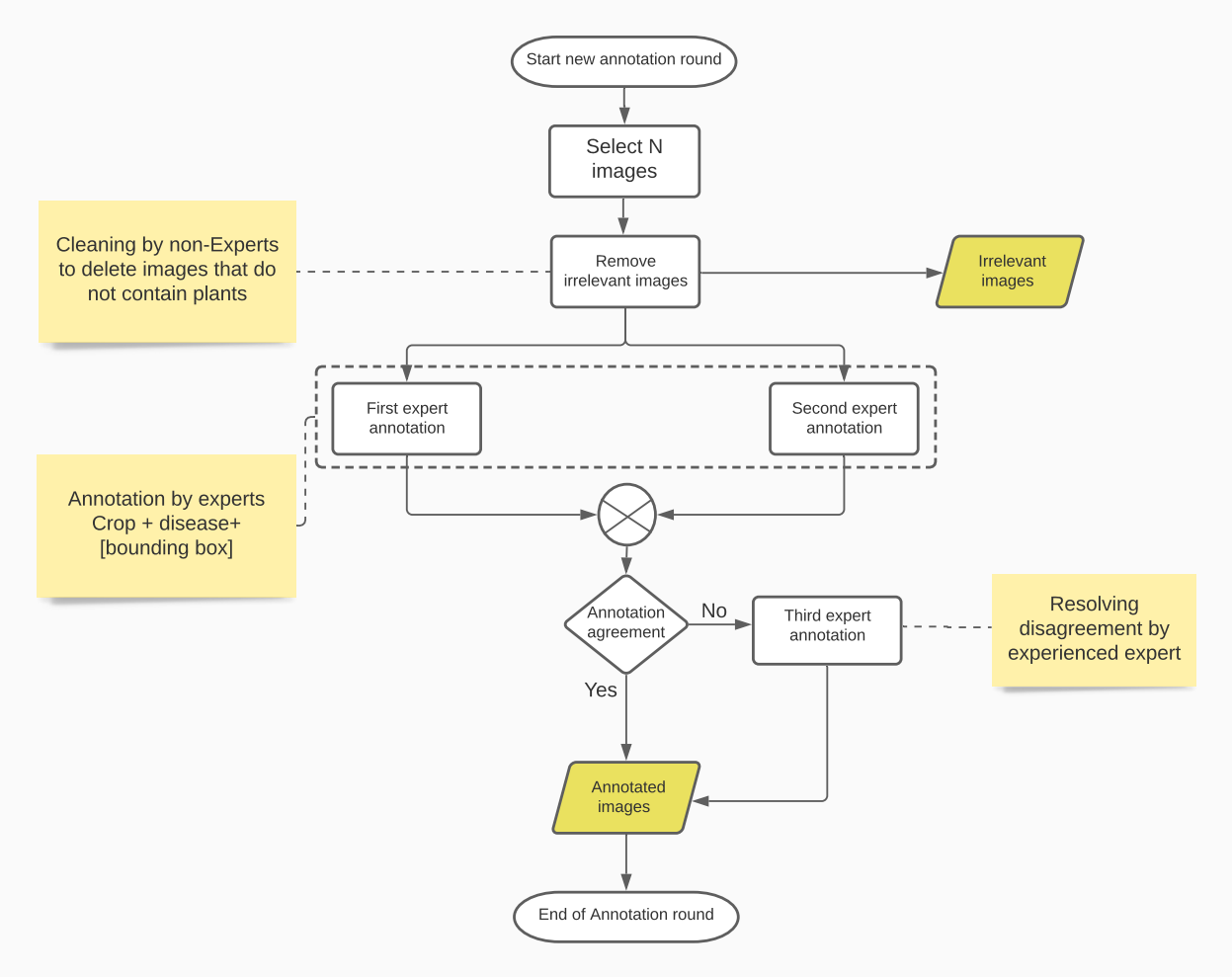

Figure 3.11: Farmy.ai data pipeline process

3.6.2 Farmy.ai Data Pipeline Details

The data pipeline of **Farmy.ai** manages the stages of the annotation process. It helps us reduce the overhead of manual tasks, such as moving images between phases, assigning tasks to annotators, checking completed phases, and finding conflicts in image annotations when exporting them.

The data pipeline ingests images with associated metadata into the Data Lake and then loads them for cleanup, starting a new round of image annotation.

The building block of the data pipeline is the Data Lake. The pipeline is well integrated with the data lake, but it is unaware of the Data-Lake architecture and infrastructure.

Since we perform data cleansing and annotation in an independent application, the data pipeline connects between the data lake and the annotation application by acting as a producer or consumer of the data lake through its ingestion and query layers.

Figure 3.12: Farmy.ai data pipeline phases

3.6.3 Farmy.ai Data pipeline phases

Images Ingestion phase

The data ingestion represents the first phase of the pipeline. The data pipeline extracts images by running Web Scraper to obtain images and their associated metadata, and by retrieving images and their metadata from the mobile application to load them into Data Lake storage.

Images cleansing

This phase only works with raw data. Since we are cleaning images manually, the data pipeline manages this phase by creating cleanup tasks for images and then loading them into the data cleansing application.

Once we complete all cleansing tasks, the data pipeline exports the cleansing results and stores them in the data lake. After it successfully exports the tasks, the data pipeline removes them from cleansing application.

Images annotation

The data pipeline manages the cleaned images by creating annotation tasks for them and loading them into the annotation project. To keep track of the annotation round, it loads new annotation tasks only when there are no annotation tasks in progress.

Finally, it exports the annotation results to the Data Lake as soon as they are ready, and after each successful export of the results, the data pipeline removes all completed tasks.

Resolving annotation conflicts

This is a conditional phase that usually occurs after annotation results have been exported from the annotation application. The data pipeline examines the annotation results from both experts to find conflicts. If it finds a conflict between annotations of same images, it generates annotation tasks for a third annotation performed by a more experienced expert.

The data pipeline is responsible for finding conflicts in annotations. It processes all exported annotation results from both annotation projects. If it does not find any conflict, it exports the annotations to the data lake. Otherwise, it generates new annotation tasks with these annotations and loads them to resolve the conflict.

3.6.4 Data Pipeline workflow

When translating the data pipeline of **Farmy.ai** to a workflow (see figure 3.13), more work is added to ensure the correctness of the phases, e.g. checking for completed annotation, completed cleanup tasks, generation of annotation rounds, but the main phases are still the same.

The phases of the data pipeline have common functions, such as importing tasks, exporting results, and checking complete phases. Each phase of the pipeline includes 3 tasks:

1. Generate work tasks and load the tasks to process them (cleansing, annotation, resolving annotations conflict).
2. Each phase has a task to export the result of the whole phase.
3. All phases have a completion task that deletes completed tasks after each data export.

Searching for conflicts is a standalone task that works with exported annotations to find mismatched annotations and load them into third annotations.

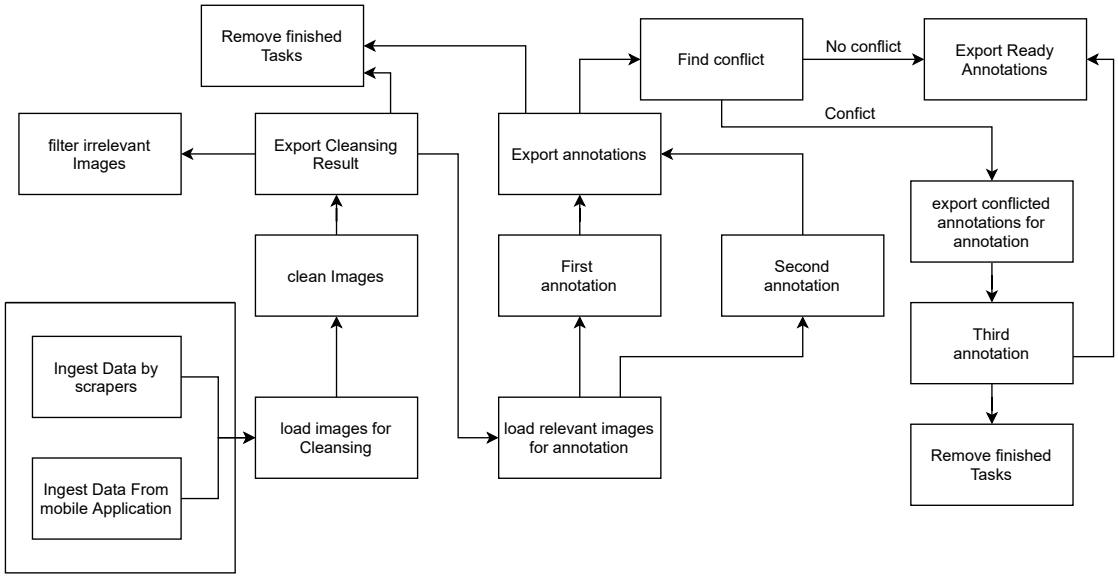

Figure 3.13: Farmy.ai data pipeline workflow

The data pipeline checks to see if tasks are ready, empty projects to ensure that operations are running correctly. A complete, successful round of the annotation data pipeline produces ready annotations. Images and annotations should be ready to be trained by data scientists with images for Deep Learning.

3.6.5 Integration of Data Pipeline with the Data Lake

We describe how we integrate the data pipeline into the **Farmy.ai** Data Lake and how it interacts with its components. The data pipeline uses query engines to find target images and their metadata, it queries the data catalog, the data lake storage when it runs.

Each time the data pipeline generates tasks for cleansing, annotation, and resolving conflict tasks, it stores the generated tasks in the staging area using connectors, which indexes all stored tasks in the data catalog passing through the entire ingestion life-cycle.

Figure 3.14 represents the complete integration of the **Farmy.ai** image Annotation pipeline with the Data Lake.

Here we wrap up all interactions of the data pipeline with the data lake architecture:

- The data pipeline uses data lake storage as its main storage.
- The data pipeline uses connectors to load processing data (tasks) into the Data Lake

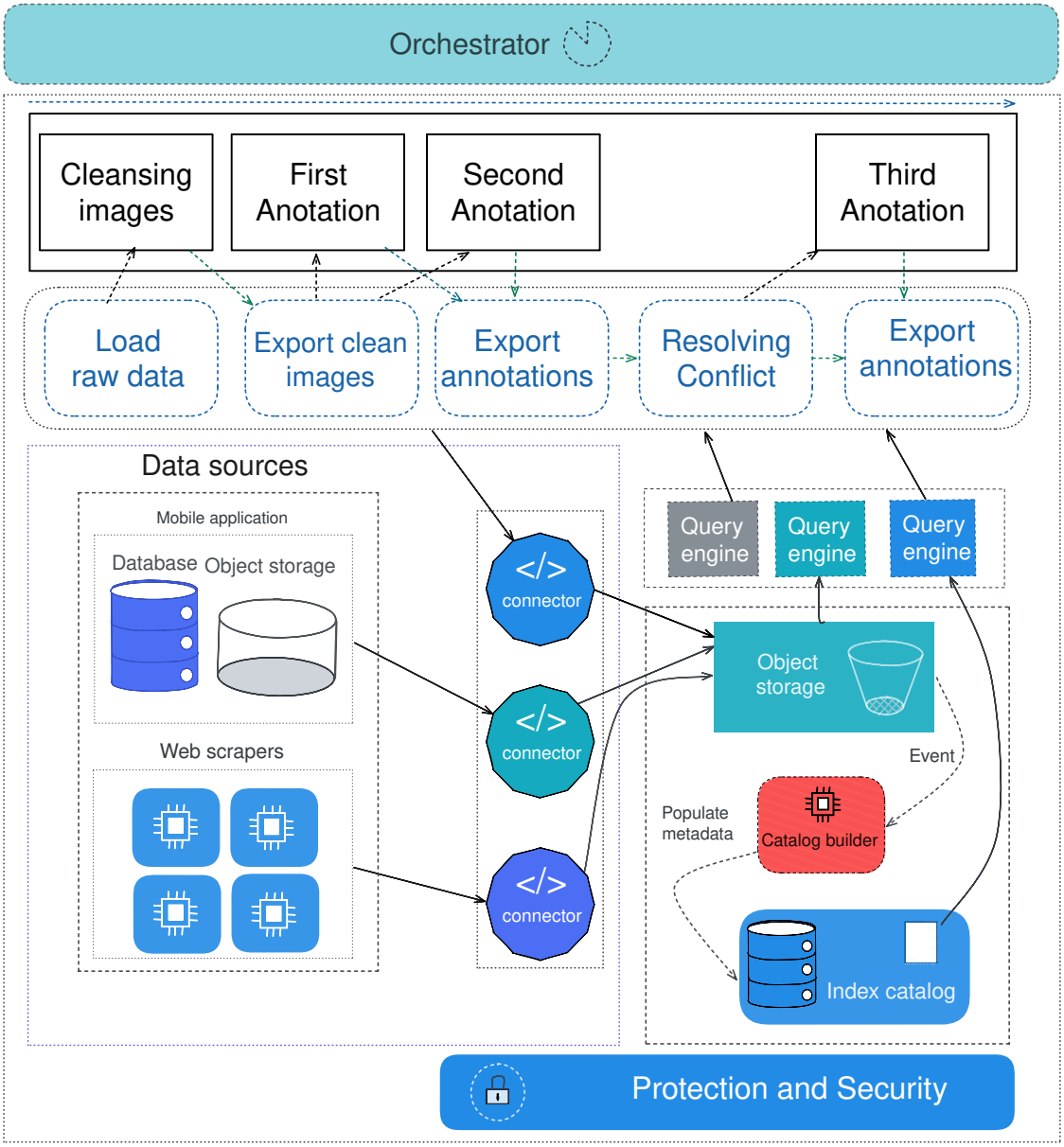

Figure 3.14: integration of annotation data pipeline with the data lake

staging zone.

- We query results and raw data with query engines.
- The data pipeline stores all processing tasks in the staging zone of the data lake, including intermediate cleansing and annotation results.
- It exports its final annotation results and stores them in the curated zone using connectors.

3.6.6 Orchestrating the data pipeline

Managing and provisioning the data pipeline workflow is particularly challenging because orchestration involves coordinating the various tasks.

The keystone of orchestrating is automation. Orchestrating of a data pipeline includes scheduling of tasks, resource, execution environment, retries, error handling, logs.

Clearly, a critical task like orchestration needs a robust system that can do it independently and reliably. There are many open-source tools that can reduce the overhead of orchestration, and let us focus more on logic.

In addition to the tool for orchestrating the data pipeline, we still need to model the data pipeline and represent it as a workflow. In particular, we can represent almost any data pipeline as DAG (direct acyclic graphs), which simply represent tasks as nodes, while the dependencies between them are the edges.

3.6.7 Data lineage in the data pipeline

Running the data pipeline and tracking the flow of data through the various tasks, from source to final destination, is a non-trivial task. Data lineage is about following the path of data from its original source to its final destination [32]. Therefore, data lineage is the key to ensure traceability of various data transformations and data movements in the pipeline.

Data linage involves more metadata management. when we consider the architecture of the data lake, There is a single source of truth for metadata management, which is the data catalog.

Using the data catalog could support our purpose in data lineage, but we can do more. A complementary approach is to rely on the Data Lake storage zones. A more detailed way is

to divide the data storage into a hierarchical structure that reflects the context of the data it contains.

Achieving a storage hierarchy has added a prominent data lineage process, Figure 3.15 represents a hierarchical structure of the **Farmy.ai** Data Lake storage for the Annotation Data Pipeline workflow, helping to track the data flow from the first step to the last.

Adding inputs and outputs for each processing task in the staging area forces us to ensure data immutability, which allows for flawless data movement.

Processing tasks should generate entirely new data. Reducing transformation significantly increases data lineage and reduces extensive processing errors and accidental data corruption.

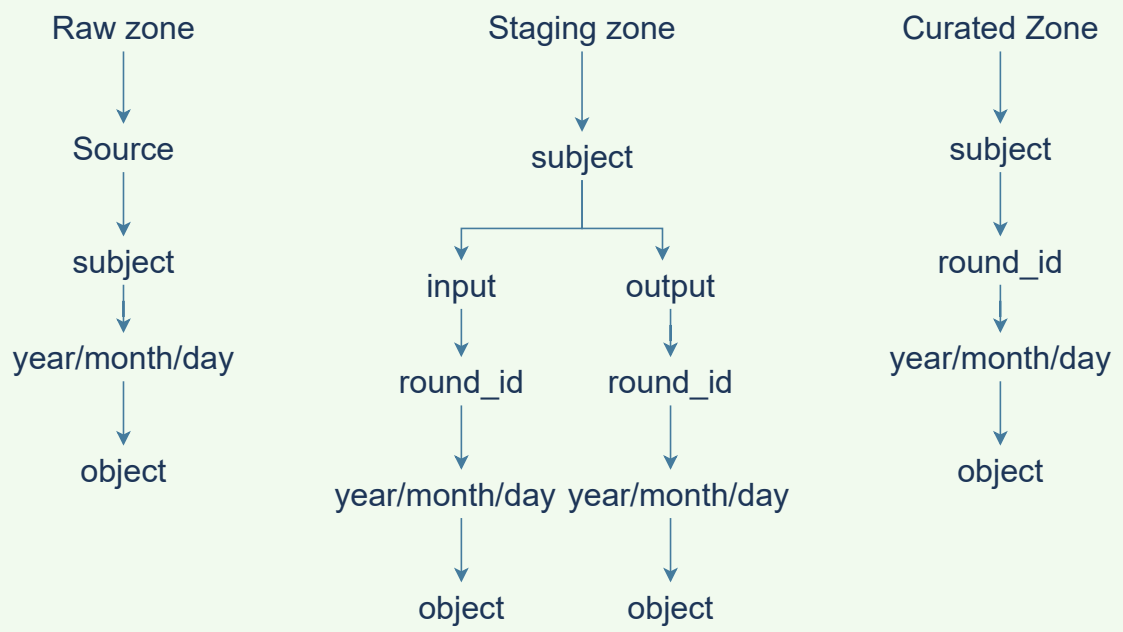

Figure 3.15: Hierarchical structure of Data Lake's object storage

In addition, logging provides great capabilities to track data movement, errors, and get feedback on various tasks in progress to intervene or improve them for future processing.

The combination of the data catalog, hierarchical structure of the data lake, and extensive logging helps us to sufficiently track and feel confident about the datasets generated during the data pipeline.

3.7 Conclusion

In this chapter, we have shown how we build the **Farmy.ai** data pipeline using the data lake architecture to offload capacity from databases and achieve a low-cost architecture. We showed the importance of a data catalog to track data lake assets, how to maintain and build the data catalog, and what to choose as storage for the data lake.

We have shown how we can create low-coupled and reusable ingestion components that ease the burden of dealing with multiple data sources.

We showed how query engines enable reusability and add search capabilities to our data lake. We then showed how to integrate the data pipeline with the data lake using connectors and query engines.

We illustrate the critical role of orchestration to execute and monitor the data pipeline. Finally, we described how we perform data lineage with data catalog, hierarchical structure, and logging.

Chapter 4

Implementation and experiments

4.1 Introduction

This chapter presents the key implementation details of Farmy's data integration solution to build the image annotation pipeline using the Data Lake architecture and data annotation application.

We describe various tools, services, environments, infrastructure, tools used, and how we use Amazon Web Services (AWS) as the infrastructure for our work.

At the end, we describe a complete end-to-end experiment of the data pipeline with the demonstration and presentation of the results. This includes a concrete visualization of the tools and the results obtained.

4.2 Infrastructure and environment

Undoubtedly, choosing an environment and the tools required can be challenging and require some time of investigation, comparison and actual testing of tools to decide.

For the development and deployment of the **Farmy.ai** data pipeline, it took some time to get an idea of the combination of tools to be used for the system. However, given the limited resources and short time it took to build the business, the easiest route was to choose what was essentially a cloud-native solution.

In this section, we present the main parts and services used to build and deploy an experimental integration system of **Farmy.ai**, including the data lake, the data pipeline, and the data annotation tool.

4.3 Choosing a cloud provider

Numerous organizations across different industries are using the cloud for various use cases. The cloud is an easy option for any startup, Farmy.ai is no exception. We have chosen many Amazon Web Services (AWS) cloud services and solutions to implement Farmy data pipeline along with some other open source solutions.

We have chosen AWS as our main cloud provider. AWS has a global presence due to its features and offers great flexibility and different levels of control over infrastructure, management and monitoring, and provides a wide range of services.

AWS offers a robust Object-Storage, the Simple Storage Service (S3), wide compute options, a wide range of databases. All these services provide convenient SDKs and APIs for use and easy integration with various solutions.

4.4 AWS services used

There are various AWS services used to effectively build our data lake with the image annotation pipeline. Here we list major services and how we integrate them with each other to ensure an efficient flow of data.

4.4.1 AWS Identity and Access Management (IAM)

IAM is a free web service from AWS that allows us to securely control access to AWS resources, authentication, authorization, permissions, and set up restrictions over resources and users, we can use it through the AWS console, SDK, or CLI. The three basic concepts of IAM are users, groups, and roles. It allows us to create multiple users under the control of a root account or enable temporary access through identity federation.

It enables programmatic access through secure tokens. It allows users to be included in groups. We can associate a user or group with one or more roles. It allows us to grant

different permissions to users and groups. Policies are documents that grant access rights to AWS services to specific users and groups [33].

Figure 4.1: AWS IAM icon
[34]

4.4.2 AWS simple storage service(S3)

S3 is a highly available, scalable, and durable object-level storage built with a minimal set of features for simplicity and robustness. It can securely store an unlimited number of files with a SLA of 99.99% [35], it provides strong read-after-write consistency so that data is not corrupted across different operations, data is accessible over the web via HTTP REST API or with the AWS SDK, objects reside in named, unique containers across regions, S3 is highly cost-effective, it provides different tiers of storage based on access level to further reduce costs.

Figure 4.2: AWS S3 icon
[36]

4.4.3 AWS Dynamo

Amazon DynamoDB is a cost-effective, fully managed NoSQL database service designed primarily for storing key-value data. It is a highly scalable distributed database that supports nested JSON format and offloads the management overhead of running and scaling the database. It eliminates hardware provisioning, setup and configuration, replication, software patching, or cluster scaling[37].

Figure 4.3: AWS DynamoDB icon
[36]

4.4.4 AWS Lambda

AWS Lambda is a Serverless Compute service, more specifically a Function as a Service (FaaS) that eliminates the overhead of managing, provisioning, and monitoring servers. It has built-in auto-scaling, code monitoring, and logging, and supports a variety of programming languages [29], The defined FaaS runs on demand only for the time needed with no upfront commitment (we only pay for what we use), it scales automatically, to handle thousands of requests per second, we can call the Lambda function directly through the Lambda API or in response to various events generated by other AWS services, it is well suited for the event-driven approach to building and maintaining the data catalog of the Farmy's data lake.

Figure 4.4: AWS Lambda icon
[36]

4.4.5 Amazon EC2

Amazon Elastic Compute (EC2) is a compute capacity distributed through Amazon Web Services (AWS) with robust control over size and security, it can scale to handle massive workloads, it offers a wide and diverse selection of operating systems, storage, processor, networks, GPU, and purchasing models[38],

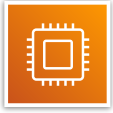

Figure 4.5: AWS EC2 icon
[36]

it provides the building block to build and deploy various applications such as web applications and APIs, Machine Learning, mobile backend services.

4.4.6 Amazon Athena

Amazon Athena is an interactive query service that allows us to easily use standard SQL to evaluate data directly in Amazon Simple Storage Service (Amazon S3)[39]. It allows us to run and query Amazon S3 with a simple setup, it allows us to query the desired data with conventional and rich SQL support.

Figure 4.6: AWS Athena icon
[36]

4.5 Programming languages And Tools

4.5.1 Python

Python is a popular open-source general-purpose programming language that is widely used in various fields, especially in data science, AI, machine learning, automation, web programming, data engineering, etc. It is a dynamic Object-Oriented and functional language that offers rich and powerful features, standard and community libraries[40]. it is a solid choice for fast scripting and simple data integration systems.

Figure 4.7: Python Logo
[41]

4.5.2 Node.js

Node.js is an open source and cross-platform JavaScript runtime built on Chrome's V8 JavaScript engine[42], it offers asynchronous I/O and event-driven functionality to create

efficient, non-blocking, event-driven programs and servers. It is a great choice for real-time applications and for building web APIs to create scalable network applications. Although it uses a single-thread model, its non-blocking mode makes it an ideal choice for high-throughput IO operations. In addition, we can launch Node.js processes to use all the cores of the machine.

Figure 4.8: NodeJS Logo
[43]

4.5.3 Apache Airflow

Apache Airflow is a highly extensible, easy-to-use open source orchestrator that programmatically authors, schedules, and monitors workflows and can execute almost any command.Apache Airflow supports modelling data pipelines of different workflows with different complexity as DAGs. It enables provisioning, monitoring, and execution of workflows that are triggered periodically or sometimes by specific events. We define pipelines in Python, which enables dynamic generation of pipelines. It also provides a web user interface for monitoring, scheduling, and managing workflows and integrates well with most cloud providers [44].

Figure 4.9: Apache Airflow Logo
[45]

4.5.4 Docker

Docker is an open source platform for building, deploying, and managing containerized applications. It provides more isolation and flexibility than virtual machines, completely separates infrastructure from running applications, and enables fast, consistent application deployment, as well as efficient application management, testing, and rapid deployment [46]. As enterprises migrate to a cloud-native model, Docker is becoming more popular. It essentially provides a set of features that allow developers to create, deploy, and control containers with a great deal of automation in hand.

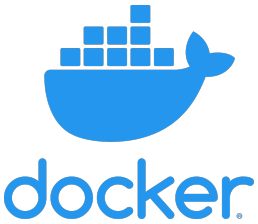

Figure 4.10: Docker Logo
[47]

4.5.5 Label Studio

Label Studio is an open source data labeling tool that allows us to label and explore different types of data. We use it for Image Classification, Semantic Segmentation, object detection on images including B-boxes, polygons. It easily integrates with Machine Learning by adding pre-labels and predictions to models to greatly improve and streamline the process. Label Studio is also available in Enterprise and Cloud editions[48].

Figure 4.11: Label Studio Logo
[49]

4.6 Experiments and results

In this section, we show an experimental end-to-end run of the data cleaning and annotation pipeline, including the tool used for image annotation (Label Studio) and the data pipeline orchestrator Apache Airflow.

4.6.1 Mapping the architecture to AWS

In addition to the AWS infrastructure, we use a few other open source tools and solutions to map the final architecture and get our data pipeline up and running. Figure 4.12 shows the core services we used to run this experiment.

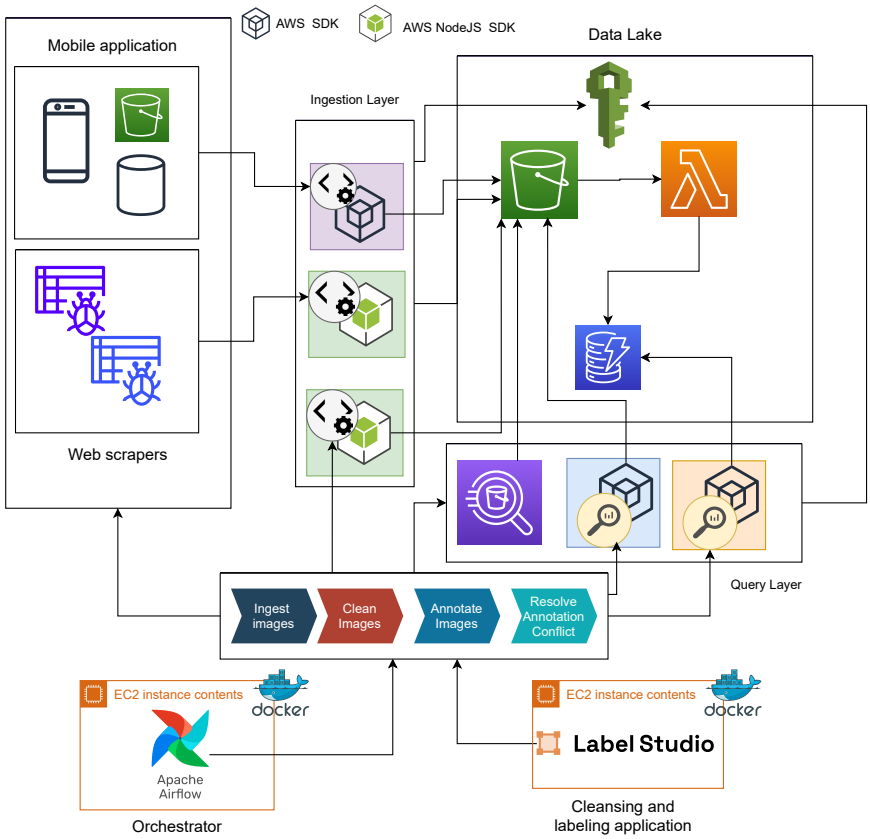

Figure 4.12: Farmy data pipeline on AWS

The Figure 4.12 shows the full architecture of the Farmy Data Lake and data pipeline deployed in the AWS infrastructure. The ingesting layer ingests data from the mobile application, and web scrapers and stores it in AWS S3. A Lambda function listens for events triggered by AWS S3 to update the data catalog, which uses AWS DynamoDB to store the metadata.

The query layer provides preconfigured libraries and AWS Athena for searching AWS DynamoDB and AWS S3. The data pipeline is driven by Apache Airflow deployed on an AWS EC2 instance. It uses Data Lake and Label Studio, also deployed on an EC2 instance, to achieve the desired behavior.

4.6.2 Starting a New Annotation Round

In this part, we describe the starting round of the Farmy.ai image pipeline. Each run of the pipeline performs a full image ingestion. For this experiment, we capture over 160 images using the Web Scraper to clean and annotate them.

On the first run, the data pipeline starts a new round of annotation each time it detects that all projects are empty or it has finished resolving annotation conflicts. Figure 4.13 shows Apache Airflow's DAG representation of the first run of the data pipeline.

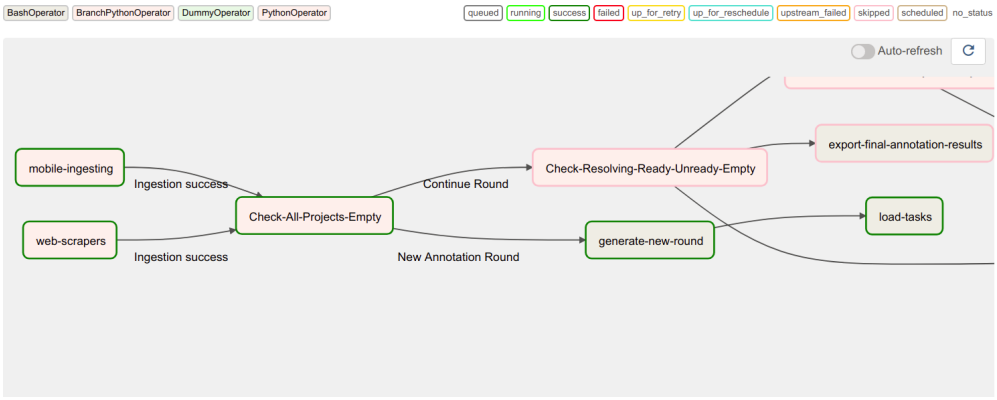

Figure 4.13: Initial loading of tasks from a new annotation round

Figure 4.14 shows the loaded tasks to Label Studio for cleaning.

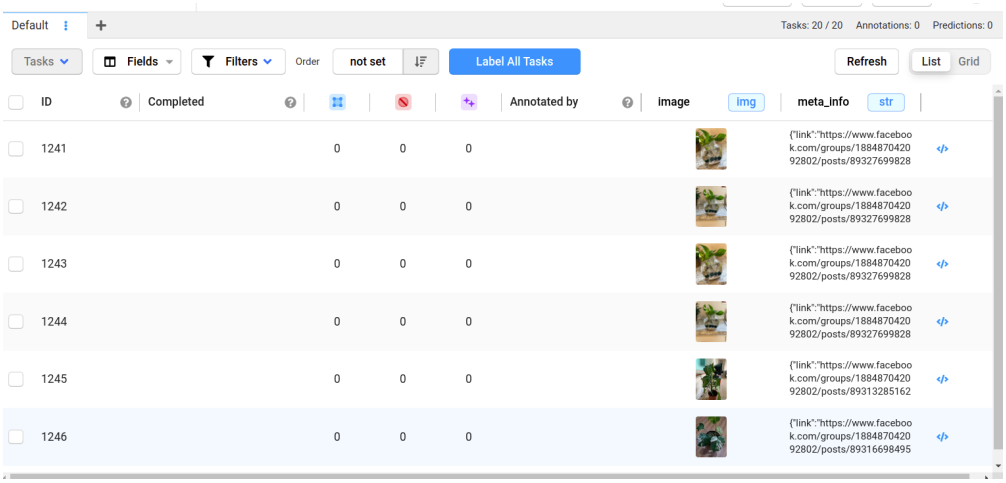

Figure 4.14: Initial loading of annotation tasks within a new round

After the first successful run, the data pipeline loads images, generates tasks, and then imports them into the annotation application for cleansing by **Farmy.ai** team.

4.6.3 Preview the annotation phase

Once we complete image cleansing, the next successful run of the pipeline exports the cleansing results and then generates tasks with relevant images. Last, it imports them for annotation by Agriculture Experts, working on two different annotation projects. Figure 4.15 shows DAG representation of this run.

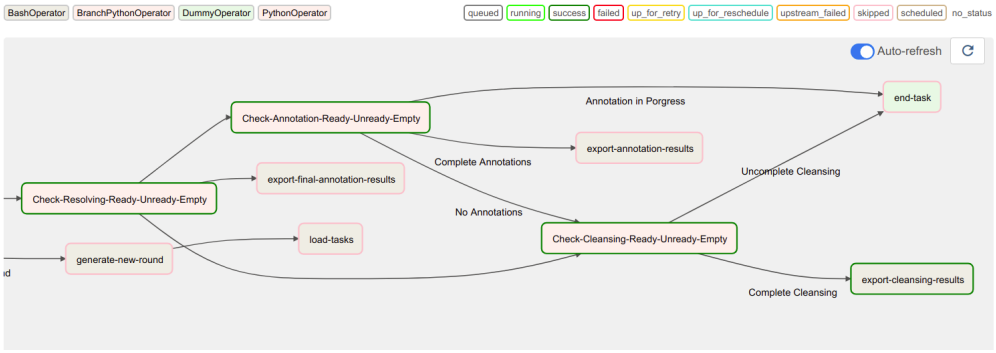

Figure 4.15: Successful export of cleansing results

Figure 4.16 shows an assigned annotation task accessible to the agricultural expert in one of the two projects.

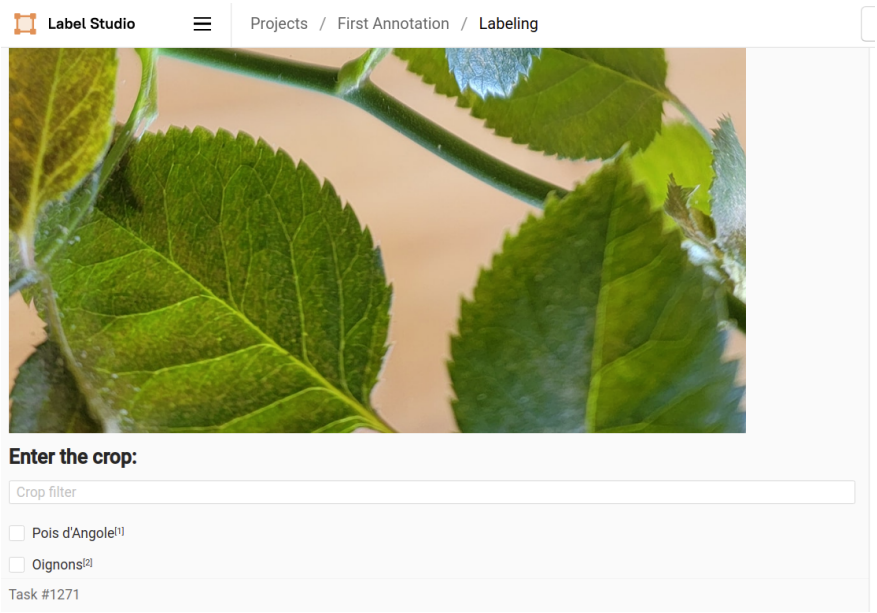

Figure 4.16: Working Annotation task

4.6.4 Resolving annotation phase

After we finish annotating images, our data pipeline exports the annotation result to work with. If we find a conflict in some image annotations, we load them into a third annotation project. Figure 4.17 shows an annotation task that resolves a conflict between two annotations.

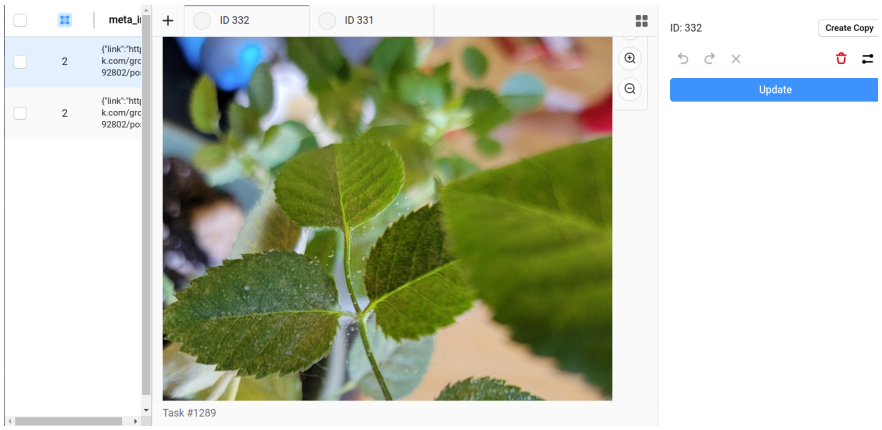

Figure 4.17: Resolving Annotation task

4.6.5 Exporting the Resolved Annotation

Ultimately, our data pipeline exports the results of the resolved annotation conflicts, which declare a complete and successful round of cleansing and annotation data pipeline. Figure 4.18 shows the DAG state of this run.

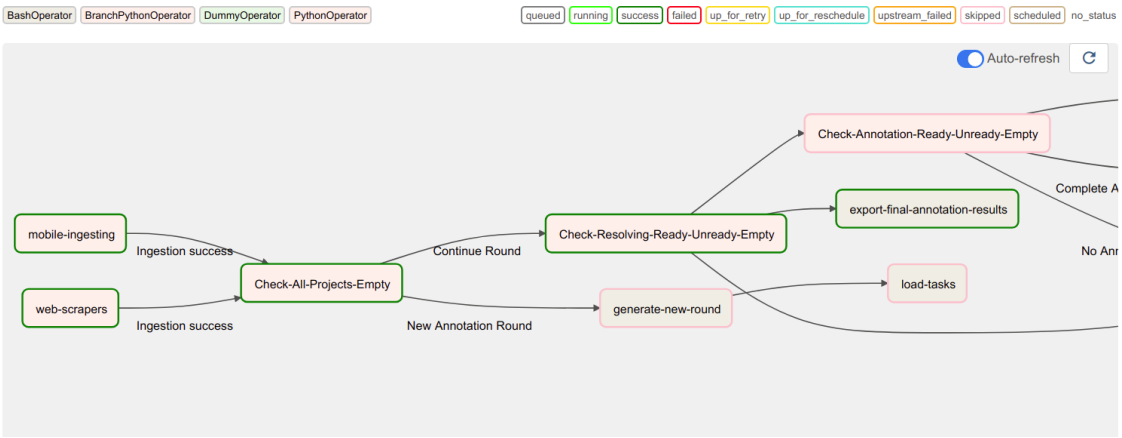

Figure 4.18: Export resolved annotations

4.6.6 Apache Airflow Orchestration

Apache Airflow orchestrates our data pipeline. It manages communication, transition between tasks, and allows us to schedule our data pipeline or respond to external events to trigger execution of DAG.

Figure 4.19 represents tree view that includes our data pipelines runs during this experiment, beside some other old runs.

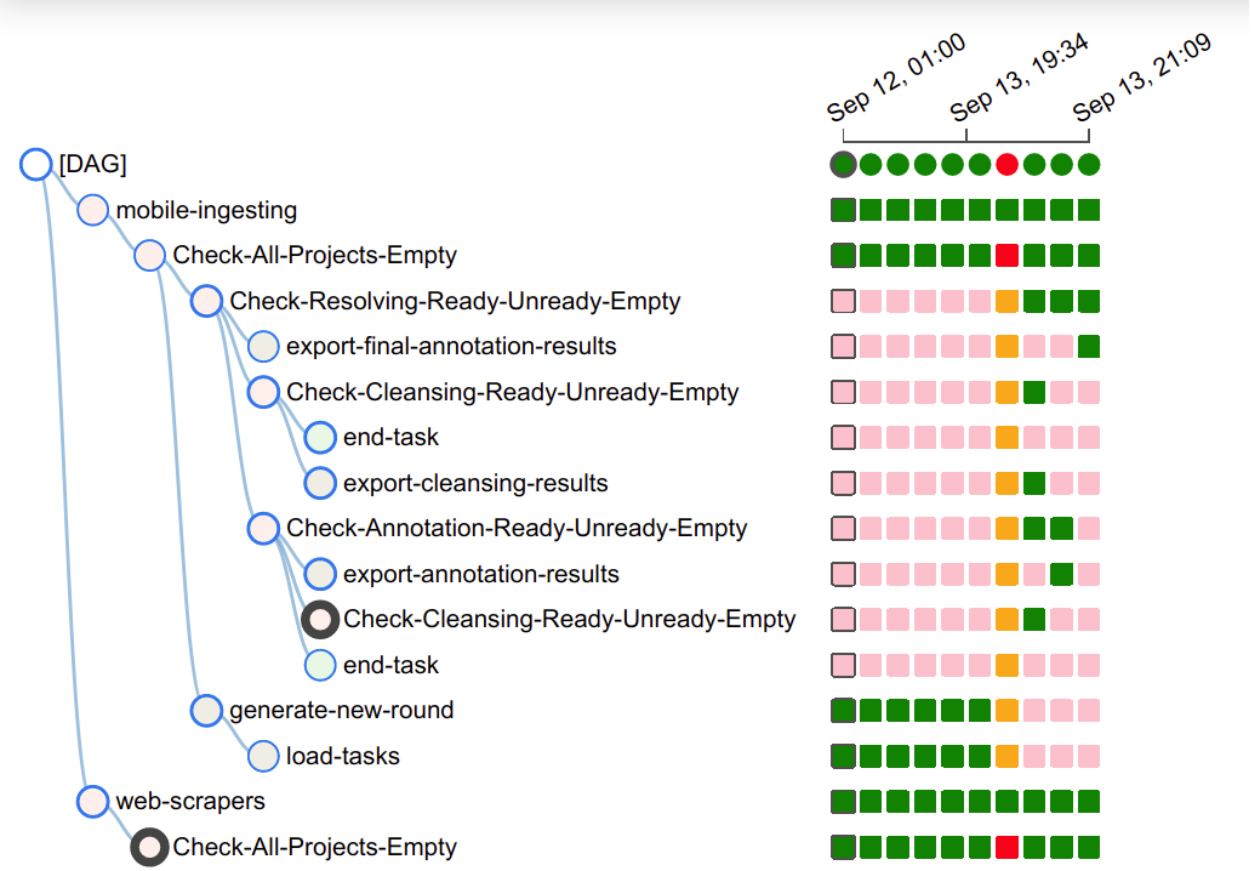

Figure 4.19: Tree view of the data pipeline

Figure 4.20 represents the Apache Airflow web user interface for configuring, scheduling, and testing the annotation data pipeline DAGs. It allows us to manage users and many other functions.

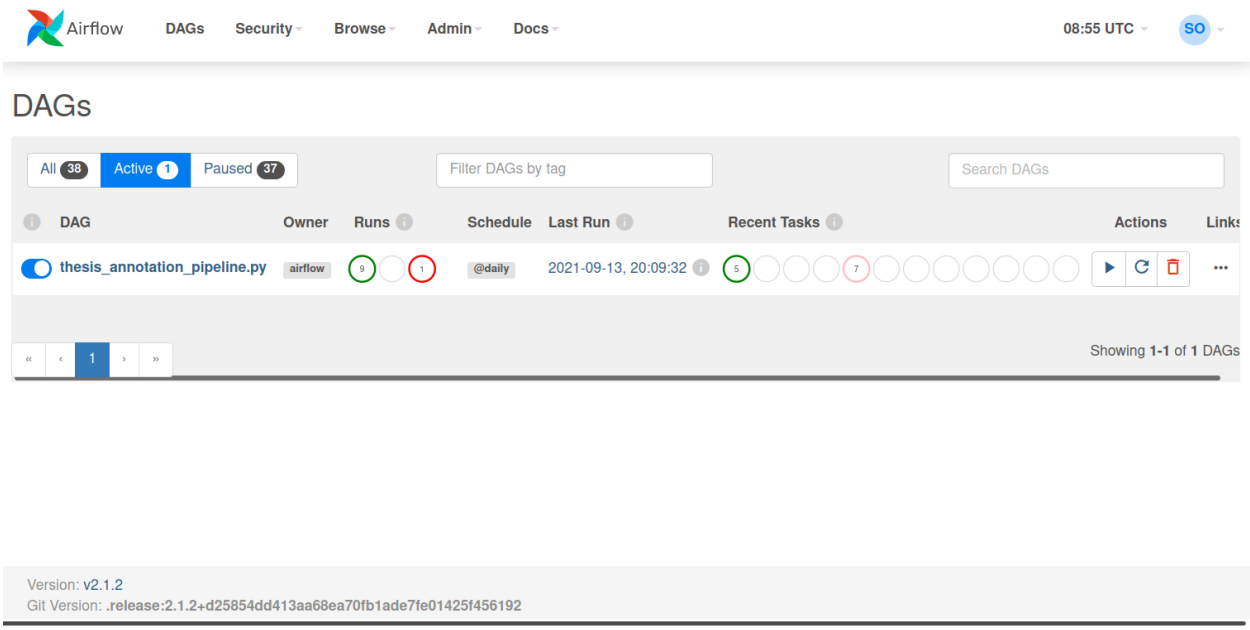

Figure 4.20: Apache Airflow Web Server User Interface

4.6.7 Query Annotations in S3 with AWS Athena

With AWS Athena, we can use SQL syntax to directly query the under-processing data such as annotation tasks assigned to the two agricultural experts, accepted or rejected images from an annotation round, assigned cleansing tasks, and, more importantly, the final annotations created.

AWS Athena, easily and reliably allows us to search the contents of working files, such as annotations tasks, by providing the schema defined in a table to read files (schema on-read).

Figure 4.21 shows the final result of the annotation pipeline after we complete the annotation round.

AWS Athena helps us find conflicting annotations in the Data Lake storage. Figure 4.22 describes an SQL query to find annotations that have gone through a third annotation to resolve a conflict.

Results

See results in full screen

	project	data
1	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241879062_10223547724893598_7352552614677129640_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
2	6	{image=s3://samyouaret-data-lake/facebook/images/1627187873837_176840171_1484773158541945_1573554244932056075_n.jpg, meta_info={round_id=2021-07-10-2021-09-08}}
3	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241858200_10223720623597388_4560996916315970884_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
4	6	{image=s3://samyouaret-data-lake/facebook/images/1627190095788_220100818_2875977825999234_6364000308600295977_n.jpg, meta_info={round_id=2021-07-10-2021-09-08}}
5	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/242030503_10222523761855263_6821614172680689069_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
6	6	{image=s3://samyouaret-data-lake/facebook/images/1627187873837_176840171_1484773158541945_1573554244932056075_n.jpg, meta_info={round_id=2021-07-12-2021-09-10}}
7	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241811213_10222523762095269_2935769640293941187_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
8	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/242030503_10222523761855263_6821614172680689069_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
9	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241795671_10223720623237379_4636312297443900256_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
10	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241770826_1831094667096159_6773636913632997991_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
11	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241744473_906752566864846_7801533486752836658_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
12	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241858200_10223720623597388_4560996916315970884_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}
13	8	{image=s3://samyouaret-data-lake/facebook/images/1627187869501_220100818_2875977825999234_6364000308600295977_n.jpg, meta_info={round_id=2021-07-10-2021-09-08}}
14	8	{image=s3://samyouaret-data-lake/facebook/images/1627190095788_220100818_2875977825999234_6364000308600295977_n.jpg, meta_info={round_id=2021-07-12-2021-09-10}}
15	6	{image=s3://farmy-dl-raw-zone/facebook/images/2021/9/13/241811213_10222523762095269_2935769640293941187_n.jpg, meta_info={round_id=2021-07-15-2021-09-13}}

Figure 4.21: Query Annotation results

Figure 4.22: Find Conflict Query

4.7 Conclusion

In this chapter, we have shown the concrete data flow in Farmy.ai, we have shown the deployment architecture in AWS, including Data Lake components such as AWS S3, AWS DynamoDB, the Lambda compute function, Apache Airflow and the Annotation web application, we have run a complete experiment with various tools and services mentioned.

Chapter 5

General Conclusion

Data is gradually changing the way we do business. Collecting and processing data has always been a challenge for engineers and developers. An efficient flow of data is the most critical operations in today's data-driven businesses.

A valuable business can start only when we guarantee that the data is available in a ready and secure state. Cloud computing greatly helps us to develop reliable and cost-effective solutions. However, a well-designed architecture is still required.

5.1 Findings

In this project, we have presented an effective architecture for managing data throughout its life-cycle. We have shown how Big Data can be integrated into the real world business, we also showed the significant impact that good data collection has on the business.

Building a reliable data pipeline is challenging, a reusable and reproducible data pipeline is even more challenging as it involves various movements, processing steps and dependencies between processing tasks. We have developed a design that leverages data cataloging and object-level storage to build a reliable data lake architecture for Farmy.ai. Developing an efficient and reusable low-coupled data ingestion layer using components such as connectors was a critical step that significantly reduces data ingestion overhead.

Avoiding a data swamp is key to data lake architecture. Creating a comprehensive data catalog was the cornerstone for fast data discovery and traceability. We demonstrated how

to achieve an event-driven architecture to build and maintain an up-to-date data catalog.

To track the path of data in the data lake, we showed the hierarchical structure of data lake storage that leads us to a modular and clean schema to track different data movements or produced assets during different processing operations. Combining such a structure with the data catalog, consistent logging, and robust orchestration helped us efficiently manage and execute the image annotation pipeline of **Farmy.ai**.

5.2 Final thoughts

Developing data pipelines based on Data Lakes does not mean we have to throw away other data tools and solutions like data warehouses. In fact, data warehouses integrate well with data lakes because they offer great search and processing capabilities, while data lakes can significantly reduce the cost of data warehouses by offloading their capacities.

Building Data pipelines with Data Lakes provides great flexibility, but data lineage is not always easy and depends heavily on the type of data processing and data movement in the system, there are a variety of tools and solutions that can reliably perform this task.

Bibliography

[1] Robert Finger Achim Walter, Robert Huber, and Nina Buchmann. *Opinion: Smart farming is key to developing sustainable agriculture*. URL: https://www.pnas.org/content/114/24/6148.

[2] Edward CurryWolfgang Wahlster José María Cavanillas. *New Horizons for a Data-Driven Economy. A Roadmap for Usage and Exploitation of Big Data in Europe*. SpringerLink.com, 2016. ISBN: ISBN 978-3-319-21569-3.

[3] The Publications Office of the European Union. *The EU data protection reform and big data*. URL: https://op.europa.eu/en/publication-detail/-/publication/51fc3ba6-e601-11e7-9749-01aa75ed71a1.

[4] Gartner Glossary. *Big Data*. URL: https://www.gartner.com/en/information-technology/glossary/big-data.

[5] Professor Gary king. *Big Data is Not About the Data*. URL: https://gking.harvard.edu/presentations/big-data-not-about-data-19.

[6] FACP Dr.Anil Jain MD. *The 5 V's of big data*. URL: https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/.

[7] www.domo.com. *Data Never Sleeps 5.0*. URL: https://www.domo.com/learn/data-never-sleeps-5.

[8] IEEE Jayesh Patel Senior Member. *OVERCOMING DATA SILOS THROUGH BIG DATA INTEGRATION*. URL: https://www.researchgate.net/publication/339088516_OVERCOMING_DATA_SILOS_THROUGH_BIG_DATA_INTEGRATION.

[9] H.Tankovska. *Most popular social networks worldwide as of January 2021, ranked by number of active users*. URL: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

[10] H.Tankovska. *Daily time spent on social networking by internet users worldwide from 2012 to 2020.* URL: https://www.statista.com/statistics/433871/daily-social-media-usage-worldwide/.

[11] Talend. *What is a Data Warehouse and Why Does It Matter To Your Business?* URL: https://www.talend.com/resources/what-is-data-warehouse/.

[12] AWS. *What is NoSQL?* URL: https://aws.amazon.com/nosql/.

[13] Apache Hadoop. *HDFS Architecture.* URL: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Introduction.

[14] AWS. *What is Cloud Object Storage?* URL: https://aws.amazon.com/what-is-cloud-object-storage/.

[15] Dave Wells. *What Is a Data Catalog?* URL: https://www.alation.com/blog/what-is-a-data-catalog/.

[16] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters.* URL: https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf.

[17] Apache Hadoop. *MapReduce Tutorial.* URL: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Purpose.

[18] Microsoft docs. *Extract, transform, and load (ETL).* URL: https://docs.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl.

[19] Ph.D Iman Samizadeh. *A brief introduction to two data processing architectures — Lambda and Kappa for Big Data.* URL: https://towardsdatascience.com/a-brief-introduction-to-two-data-processing-architectures-lambda-and-kappa-for-big-data-4f35c28005bb.

[20] Milinda Pathirage. *kappa-architecture.com.* URL: https://milinda.pathirage.org/kappa-architecture.com/.

[21] AWS. *What is a data lake?* URL: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake.

[22] Sultan Ahmad Ajit Singh. *Architecture of Data Lake.* URL: https://www.researchgate.net/publication/331890045_Architecture_of_Data_Lake.

[23] https://databricks.com. *Data Lake*. URL: https://databricks.com/glossary/data-lake.

[24] Inc. Glossary Hazelcast. *What Is a Data Pipeline?* URL: https://hazelcast.com/glossary/data-pipeline.

[25] Helena Homström Olsson Aiswarya Raj Munappy Jan Bosch. *Data Pipeline Management in Practice: Challenges and Opportunities*. URL: https://research.chalmers.se/publication/523476.

[26] Garrett Alley. *What is a Data Pipeline?* URL: https://www.alooma.com/blog/what-is-a-data-pipeline.

[27] ROBERT (MUNRO) MONARCH. *Human-in-the-Loop Machine Learning*. Manning Publications Co, 2021. ISBN: 9781617296741.

[28] AWS. *Serverless on AWS*. URL: https://aws.amazon.com/serverless/.

[29] AWS Docs. *What is AWS Lambda?* URL: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[30] Talend Tech Team. *Don't Let your Data Lake become a Data Swamp*. URL: https://www.talend.com/blog/2018/06/28/dont-let-your-data-lake-become-a-data-swamp/.

[31] AWS Whitepaper. *Data Cataloging*. URL: https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/data-cataloging.html.

[32] www.imperva.com. *What is Data Lineage?* URL: https://www.imperva.com/learn/data-security/data-lineage/.

[33] AWS Docs. *What is IAM?* URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html.

[34] www.iconfinder.com. *Iam, management, copy, deployment icon*. URL: https://www.iconfinder.com/icons/259295/iam_management_copy_deployment_icon.

[35] AWS Docs. *What is Amazon S3?* URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[36] aws.amazon.com. *AWS Architecture Icons*. URL: https://aws.amazon.com/architecture/icons/.

[37] AWS Docs. *What Is Amazon DynamoDB?* URL: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[38] aws.amazon.com. *Amazon EC2*. URL: https://aws.amazon.com/ec2.

[39] aws.amazon.com. *Amazon Athena*. URL: https://aws.amazon.com/athena/.

[40] docs.python.org. *The Python Tutorial*. URL: https://docs.python.org/3/tutorial/index.html.

[41] python.org. *The Python Logo*. URL: https://www.python.org/community/logos/.

[42] https://nodejs.dev. *Introduction to Node.js*. URL: https://nodejs.dev/learn/introduction-to-nodejs.

[43] nodejs.org. *Resources*. URL: https://nodejs.org/en/about/resources/.

[44] Apache Airflow. *Apache Airflow*. URL: https://airflow.apache.org/.

[45] Jarek Potiuk. *Airflow logos*. URL: https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+logos.

[46] docs.docker.com. *Docker overview*. URL: https://docs.docker.com/get-started/overview/.

[47] www.docker.com. *Docker Logos*. URL: https://www.docker.com/company/newsroom/media-resources.

[48] https://labelstud.io. *Get started with Label Studio*. URL: https://labelstud.io/guide.

[49] https://labelstud.io. URL: https://labelstud.io/.