UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

# MASTER 1 THESIS

By

**DANG THAI SON (2440045)**

*Information and Communication Technology (ICT)*

Title

# ETL Microservice for Data Ingestion into a Data Lake

---

Supervisor: **Assoc. Prof. Trần Giang Sơn**

*ICT Laboratory - USTH*

**Hanoi, August 2025**

# DECLARATION

I, hereby, Dang Thai Son, declare that all the work and result in this thesis are entirely my own and are not plagiarized from any source. This thesis was written based on my research which was carried out at the Information and Communication Technology laboratory at University of Science and Technology of Hanoi, under the guidance of Assoc. Prof. Tran Giang Son.

Any scientific result, method, comment, and statistics inherited during the research from other authors has been cited thoroughly. In case there is any plagiarism in my thesis, I understand that this work will not be evaluated and I will take full responsibility for penalties from the thesis defense committee and my university.

# ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who offered me the chances to be capable of carrying out my project and then completing this thesis.

First and foremost, I cannot begin to express my thanks to Assoc. Prof. Tran Giang Son, my supervisor, for his invaluable guidance and relentless support throughout my internship. With his expert advice along with immerse patience, and considerable encouragement, I could have a thorough look of the project plan and then define proper methods to execute it. Had it not been for him, I would not have been able to have a grasp of the topic. Working with Assoc. Prof. Tran Giang Son was a tremendous opportunity for me to strengthen not only my knowledge but also my professional experience, which shall carry me a long way in the future.

In addition, I would like to extend my sincere gratitude to all the members of my thesis committee for their time and effort on reviewing my work. Their insightful comments and constructive criticism shall greatly improve the quality of this project.

Last but not least, I very much appreciate all staff and lecturers in the Information and Communication Technology Department at University of Science and Technology of Hanoi for their overwhelming support throughout foundational courses in the master program which serves as a decisive part in this research. Besides, I would like to thank Mr. Le Nhu Chu Hiep specifically, for his active cooperation and additional critical information about the ULake during the internship. The completion of my dissertation would not have been possible without these enormous helps.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| IDC | International Data Corporation |
| AI | Artificial Intelligence |
| ETL | Extract, Transform, Load |
| GA4 | Google Analytics 4 |
| ICT | Information and Communication Technology |
| CDSE | Copernicus Data Space Ecosystem |
| I/O | Input/Output |
| OAuth | Open Authorization |
| REST | Representational State Transfer |
| JSON | JavaScript Object Notation |
| RDBMS | Relational Database Management System |
| SQL | Structured Query Language |
| GDAL | Geospatial Data Abstraction Library |
| TIFF | Tagged Image File Format |
| NASA | National Aeronautics and Space Administration |
| USGS | United States Geological Survey |
| ESA | European Space Agency |
| ALOS | Advanced Land Observing Satellite |
| JAXA | Japan Aerospace Exploration Agency |
| MODIS | Moderate Resolution Imaging Spectroradiometer |
| STAC | SpatioTemporal Asset Catalog |
| SAR | Synthetic Aperture Radar |
| UV | Ultraviolet |

| | |
|---|---|
| S2-L2A | Sentinel-2 Level-2A |
| NDVI | Normalized Difference Vegetation Index |
| CI | Continuous Integration |
| DAO | Data Access Objects |
| DTO | Data Transfer Objects |
| CSV | Comma Separated Values |
| XLSX | Excel Open XML Spreadsheet |
| HDFS | Hadoop Distributed File System |
| MB | Megabyte |
| ID | Identification |
| PNG | Portable Network Graphic |

# LIST OF FIGURES

# ABSTRACT

In recent years, the growing volume of satellite imagery has raised new challenges for traditional data management systems. To deal with the issue, this project presents the design and implementation of an ETL (Extract, Transform, Load) microservice for satellite data integration into ULake, a on-premises data lake infrastructure at the ICT Lab. Built with the Quarkus framework, the microservice enables secure authentication with the Copernicus Data Space Ecosystem (CDSE) to retrieve metadata and imagery over a defined geographic region and time interval, preprocess raw data to meet system conditions, and upload the result to ULake.

The system was tested on Sentinel-2 L2A dataset with the Red River Delta in Vietnam as the region of interest from January 2023 to January 2024. It successfully extracted hundreds of metadata records and corresponding images, transform raw TIFF images with gdal_translate using the nearest-neighbor resampling algorithm to reduce their resolution while preserving spectral information, and finally performed data uploading to ULake. Despite limitations such as token expiration during extraction and lack of full automation, the microservice showed efficiency and reliability in handling Earth observation datasets.

This project not only contributes to the enhancement of ULake's data ingestion capabilities but also offers a reusable and extensible ETL pipeline supporting scientific researches and analysis.

**Keywords**: *ETL, Microservice, Quarkus, Data lake, Satellite imagery, CDSE, ULake, Sentinel-2 L2A, GDAL translate, Nearest-neighbor resampling*

# INTRODUCTION

## I/ Overview & Purpose

In today's modern world, data grows exponentially and continuously in numerous domains. According to a report by International Data Corporation (IDC) and Seagate, the global datasphere which is the total amount of data created and duplicated all over the world is estimated to reach 163 zettabytes ($1.63 \times 10^{14}$ gigabytes) by 2025 [1]. This statistic is approximately ten times bigger than that of a decade ago. This data deluge results in a new terminology called "big data" which generally involves enormous, fast-moving, and diverse datasets. Thanks to it, various information-based applications have been developed to serve critical needs such as analytical models in agriculture, or forecast frameworks in weather and climate fields [2]. Some popular big data systems supporting scientific research include Copernicus Global Land Services, Sentinel-2, Landsat Program, ALOS World 3D, and ERA5 that provide valuable environmental information. However, coming with major advantages, big data has raised 3 V challenges (Volume, Variety, and Velocity) to traditional data systems, such as relational databases, which struggle with issues like scalability, handling unstructured formats, and real-time processing. Specifically, traditional systems may face performance bottlenecks and storage constraints when processing petabytes of data and managing the velocity of streaming inputs [3]. These limitations have urged for the development of new solutions, one of which is data lake.

In order to conveniently store, manage, and extract value from big data where sources are massive and complex, a new type of data storage architecture which is data lake has emerged. A data lake is defined as a highly scalable centralized repository that allows storing structured, semi-structured, and unstructured data in its raw format without the need to define a schema at the time of ingestion. In other words, this schema-on-read mechanism provides a flexible way to store and access heterogenous and high-volume data coming from sources like satellites, sensors, or user logs, without forcing early transformation. From an architectural viewpoint, data lakes can be broadly classified into two categories based on the deployment method: cloud-based and on-premises [4]. A cloud-based data lake leverages remote storage and services from public cloud providers, offering rapid scalability and minimal infrastructure management. In contrast, an on-premises data lake is hosted within the organization's internal infrastructure where hardware, storage, and networking resources are controlled locally. Without such scalable storage architectures, traditional systems would face challenges when working with modern data streams, which would drastically reduce the ability to implement cutting-edge AI models or to fulfil analytical work. To seal that gap, several large-scale data lakes have been developed and widely adopted for either scientific or industrial use such as Amazon S3-based data lake, Google Cloud Storage, and Microsoft Azure data lake.

In the context of data lake, ETL process (Extract, Transform, and Load) is widely considered the backbone of the entire architecture since it performs the core operations needed to deliver up-to-date and consistent data for analysis and decision-making. As the name suggested, ETL is a sequential three-phase data integration procedure. Firstly, diverse raw data from multiple sources, for example structured databases, GA4 statistics, or user transaction logs, is extracted. Then using methods like format conversion, deduplication, and resizing, it is transformed into a clean and consistent format before being loaded into a destination system, here is the data lake, for analytical platforms or model training purposes [5]. Its importance lies in the fact that it not only automates data movement from source systems to target repositories but also ensures that data is clean, and compliant with business rules before it is stored. Without a properly implemented ETL process, organizations could have problems with data inconsistencies, schema mismatches, and loss of traceability, especially when working with high-volume or heterogeneous sources [6]. Take major data lake platforms, Amazon S3, Google Cloud, and Microsoft Azure, as examples, they all use ETL pipelines which are AWS Glue, Google Cloud Dataflow, and Azure Data Factory respectively for loading clean and structured data. Overall, ETL is indispensable for ensuring raw and unstructured data becomes reliable and usable for downstream applications.

In recent years, the acceleration of climate change and environmental degradation has boosted the global demand for reliable Earth observation and forecasting systems. To understand and respond to these needs, data scientists and analysts must work with high-resolution satellite imagery and geospatial datasets which are enormous, heterogenous, and frequently updated. These datasets which are considered as a big data source then need a scalable infrastructure to be ingested, stored, and processed effectively. As previously discussed, data lakes and ETL pipelines are fundamental for such workflow. At the ICT Lab, a centralized on-premises data lake infrastructure called ULake has been established, supporting ingestion pipelines for Kaggle and GitHub. However, there is currently no dedicated ETL process for ingesting satellite data into the lake, which limits the lab's ability to fully integrate remote sensing data into its analytical workflows. Therefore, the goal of this project is to fulfil the need by developing an ETL microservice for data ingestion for the ULake that automates the end-to-end process of retrieving, transforming, and storing satellite data.

## II/ Objectives

The objective of this project is to design and implement a microservice-based ETL pipeline used for the ingestion of satellite imagery into the ULake data lake. The microservice supports interactions with a satellite data provider, here specifically is the Copernicus Data Space Ecosystem (CDSE), to extract relevant imagery and metadata. It then performs downsizing for full-band-color images as a preprocessing step before loading the data into appropriate storage locations within ULake. By doing so, this project will not only fill a current technical gap in the laboratory's data infrastructure, but also enhance the efficiency and accessibility of downstream applications such as climate modeling and machine learning pipelines. On top of that, the ETL microservice will contribute to a streamlined, reusable, and scalable data ingestion job for Earth observation datasets in research and practical applications.

# MATERIALS AND METHODS

## I/ Tools

### 1. Microservice

To ensure modularity and asynchronous processing, this project adopted a reactive microservice architecture as the backbone of the ETL system.  To start with, a microservice is a lightweight architectural style in which an application is decomposed into small and independently deployable services, each of which is responsible for a specific business function. The microservice approach aims for parallel development, easier debugging, and more flexible deployment, which is crucial in big data workflows like ETL pipelines to ULake where different services may evolve at difference paces over time. Without microservices, developers could likely resort to a contrast approach which is the monolithic architecture that would bundle all ETL logic intro a single application. This may be easier to start but then becomes difficult to maintain, hard to scale, and more prone to system failure. For example, a bug in one function could crash the entire system, requiring full re-deployment for any change [7]. To be clearer about this, below are simple graphs to compare the deployment of the two approaches:

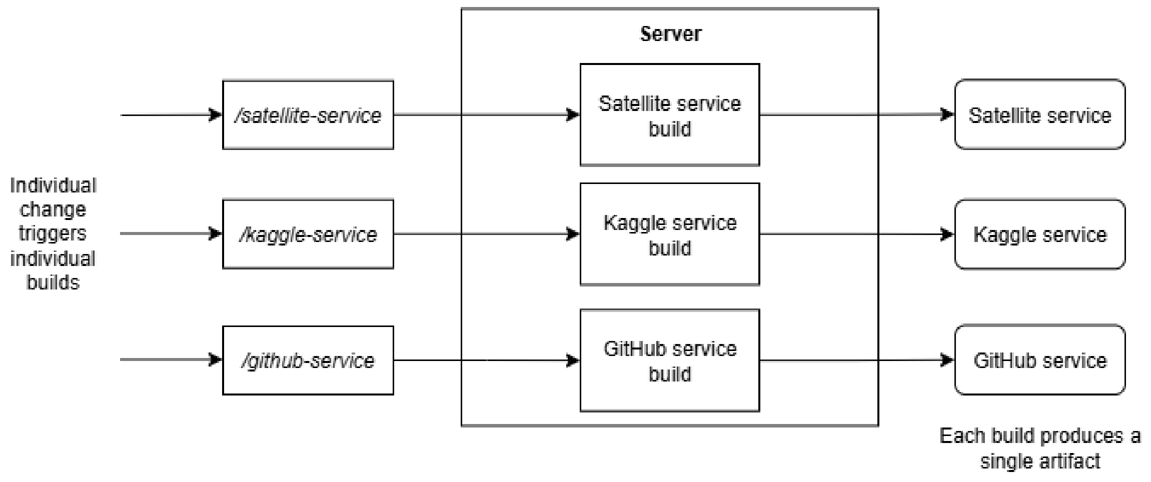


**Figure 1. Microservice architecture with one source code repository and CI build per service [7].**

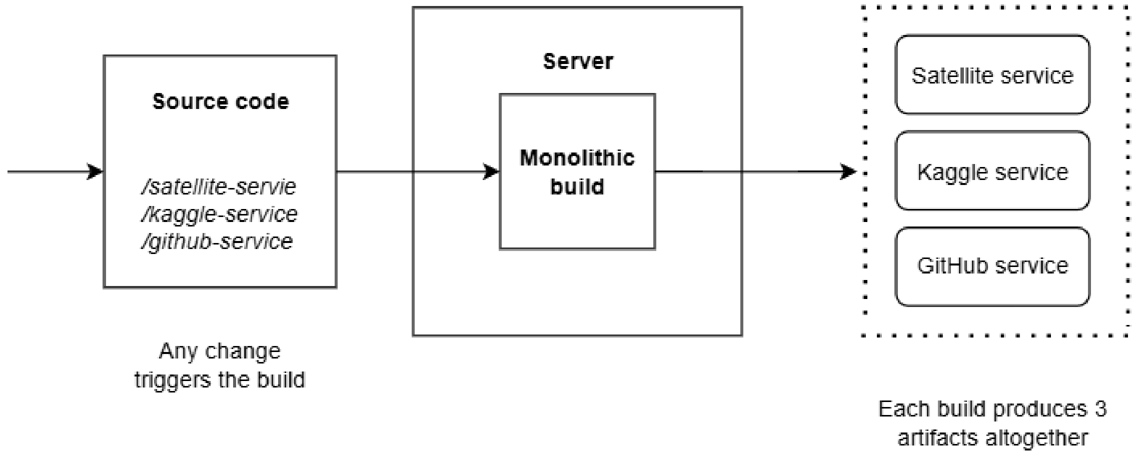


**Figure 2. Monolithic architecture with a single source code repository and CI build for all services [7].**

A reactive microservice further enhances the model by introducing non-blocking, event-driven, and backpressure-aware design patterns, aligning well with the demands of data stream ingestion and transformation. In this project, the reactive microservice was implemented using the Quarkus framework, leveraging its reactive core built on Mutiny and Vertx.

In brief, using a reactive microservice structure not only improves the isolation, scalability but also ensures the system can ingest and process satellite imagery in a timely manner, aligning with the goals of the modern data lake environment [7].

## 2. Quarkus framework

The microservice developed in this project was built using Quarkus, a modern full-stack Java framework optimized for building cloud-native, container-first, and reactive applications. Quarkus, which supports both imperative and reactive programming models, provides high performance and low memory footprint, making it well-suited for event-driven architectures and data-intensive workflows [8]. Therefore, one main reason why Quarkus was selected is its efficiency in handling data streams and non-blocking I/O operations, which is especially suitable when working with large volumes of satellite imagery and metadata from external APIs. Additionally, this choice ensured compatibility with the existing ULake ecosystem where other microservices were developed using Quarkus, which allows consistency in deployment and configuration across the data lake.

Several Quarkus extensions utilised to support the system's functionality include: quarkus-oidc-client for secure OAuth2-based communication with external APIs, quarkus-hibernate-reactive-panache to simplify entity-repository mappings using reactive Hibernate, and quarkus-rest-jackson for handling RESTful endpoints and JSON serialization. The ETL pipeline was made following a modular structure comprising entity, model, proxy, repository, resource, and service packages with reactive programming principles which leverage Mutiny to ensure non-blocking behaviour across service layers.

### 3. PostgreSQL

In this project, PostgreSQL was used as a local relational database to store structured metadata extracted during the first phase of the ETL process. PostgreSQL is an open-source relational database management system (RDBMS) known for its stability, strong community support, and adherence to SQL standards [9]. It can handle structured tabular data which is an appropriate choice for storing metadata fields such as image identifiers, capture timestamps, source URLs, coordinates, and file paths related to downloaded satellite imagery. The primary reason for choosing PostgreSQL was to save metadata locally during the development and testing of the ETL microservice, which enabled intermediate storage between extraction and transformation phases, and supported data traceability and debugging. Without such tool, metadata shall need to be stored directly in flat files which are inconsistent, hard to query, and prone to loss if the system crashes.

### 4. GDAL translate utility

To support the transformation phase of the ETL process, this project took advantage of the gdal_translate utility, which is a part of the Geospatial Data Abstraction Library (GDAL), a widely used open-source toolkit for reading, writing, and converting raster geospatial data formats. GDAL supports over 150 raster formats and capabilities for re-projection, resampling, and data compression. Specifically, gdal_translate was employed in this project to downsize large multi-band satellite images by reducing their spatial resolution while preserving all spectral bands. This step is critical to ensure the data lake input file size remains below the upload threshold, allowing successful ingestion into ULake. Without this tool or any similar one in the preprocessing step, raw satellite imagery, especially full-band TIFF would exceed storage or transfer limitations set by the data lake's ingestion interface, which may result in failure or performance degradation blocking the data ingestion workflow.

## II/ Dataset

### 1. Satellite imagery

Satellite imagery refers to images of Earth or other planets captured by satellites operated by governments or commercial providers. These images are typically obtained using onboard sensors that record electromagnetic radiation reflected or emitted from the Earth's surface. Depending on sensor specifications, satellite images can include multiple spectral bands, enabling advanced analysis of land cover, vegetation health, water bodies, and atmospheric conditions [10]. This type of data plays a vital role in a wide range of scientific and environmental applications, including climate monitoring, agricultural forecasting, and disaster management.

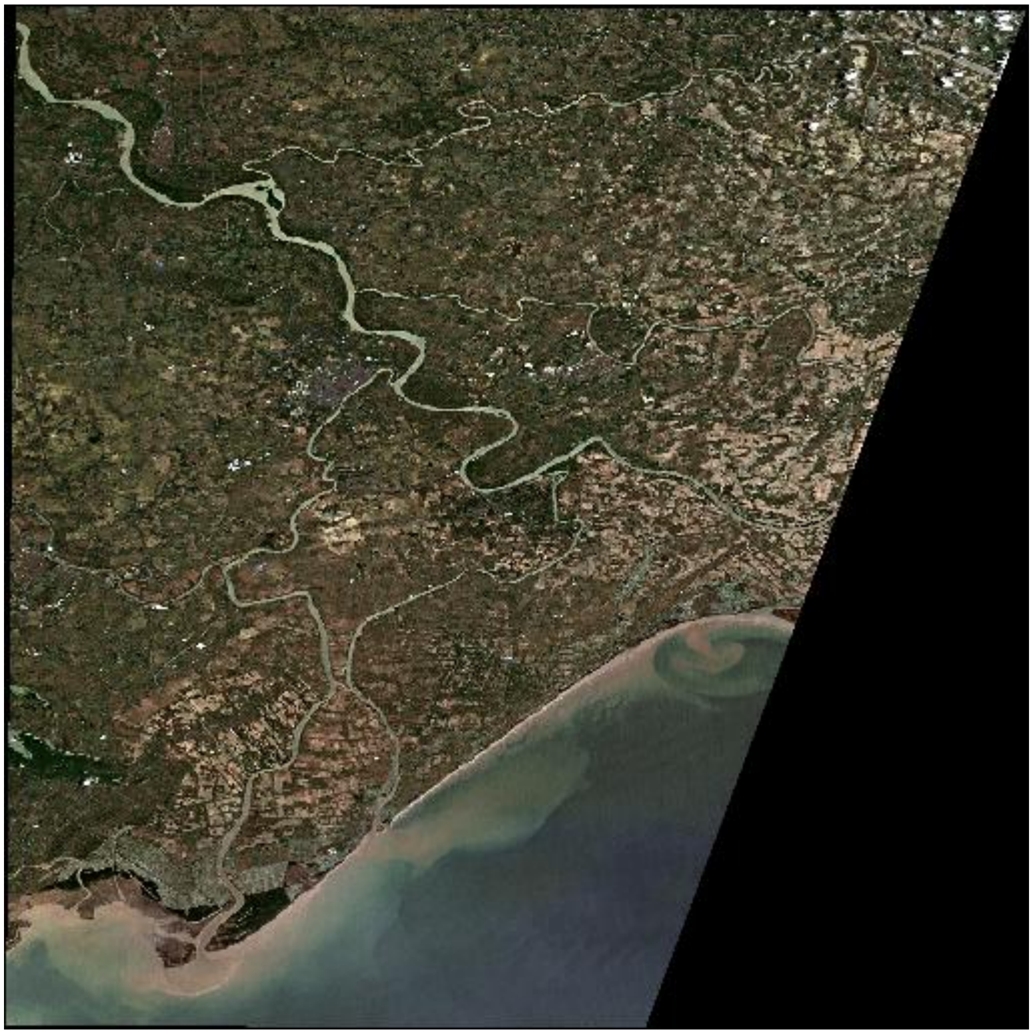


Figure 3. Example of a satellite image on Red River in Vietnam

In this project, satellite imagery serves as the primary raw data source extracted and processed through the ETL pipeline. These images are essential for enabling downstream geospatial analysis and model training tasks. Without access to satellite data, there would be serious limitations in observing environmental patterns, particularly in regions lacking ground sensors. The ability to capture large-scale and repeatable observations makes satellite imagery irreplaceable for Earth observation and global-scale analysis.

Some of the most widely used open-access satellite data programs include the Landsat program operated by NASA and USGS which has provided continuous Earth observation since the 1970s, and the Sentinel constellation operated by European Space Agency (ESA) under the Copernicus program which offers high-resolution multispectral data. Other valuable sources include the ALOS World 3D dataset by JAXA and MODIS imagery by NASA.

## 2. Project dataset

Among popular satellite data programs, the CDSE was specifically chosen as the main data provider for this project. As mentioned previously, the CDSE is an open-access Earth observation platform developed under the Copernicus Program and operated by the ESA. It is designed to provide free, large-scale, and near real-time access to satellite imagery, primarily from the Sentinel imaging mission, along with modern features such as RESTful APIs and scalable interfaces for developers and researchers. Compared to other satellite data providers, this platform offers several key advantages including detailed API documents for efficient automation in data discovery and retrieval, no token limitation for programmatic use, and standardized metadata format following SpatioTemporal Asset Catalog (STAC) for simplified filtering. In terms of the platform's features utilized in the ETL microservice, the data extraction process was driven mainly through two of its standardized APIs: Catalog API and Process API. While the former is used to search and filter metadata of available satellite scenes based on coordination and time range, the latter is for generating and retrieving imagery over a selected area of interest. Together, these APIs enable an efficient workflow to extract necessary data types for ingestion into the ULake environment for further analysis.

To be more specific about the data content, CDSE distributes big data from several Sentinel missions, which are a series of satellites developed as a part of the Copernicus Earth Observation Program. Each Sentinel satellite is designed for a specific purpose:

- Sentinel-1 provides Synthetic Aperture Radar (SAR) for land and ocean monitoring.
- Sentinel-2 supports high-resolution optical imagery for land use, vegetation, and agriculture.
- Sentinel-3 studies sea surface topography, ocean and land color and surface temperature.
- Sentienl-5P includes atmospheric composition like ozone and UV radiation.
- Sentinel-4, Sentinel-5, and Sentinel-6 are dedicated to atmospheric, air quality, and sea-level monitoring respectively.

Among these, CDSE currently offers access to Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5P datasets. The Sentinel program plays a critical role in modern Earth observation due to its high-resolution, multi-spectral sensors and short global revisit capabilities, which provides consistent and reliable imagery essential for climate change tracking, deforestation monitoring, and disaster management applications [11]. Without open-access data sources like Sentinel, many global environmental analysis tasks and machine leaning models would rely heavily on commercial satellite data which is limited in coverage and accessibility.

In this project, Sentinel-2 Level-2A (S2-L2A) imagery was used as the primary dataset for stimulation and system testing. This product contains atmospherically corrected surface reflectance data, making it suitable for vegetation indices such as NDVI (Normalized Difference Vegetation Index) or agricultural monitoring applications. While only S2-L2A was applied in this implementation, the architecture is designed to support ingestion of other Sentinel data in the same ETL pipeline with just some small modifications.

Due to the vast size of satellite archives, instead of querying the entire globe, the system focuses on one target region only: the Red River Delta in Vietnam. This area is known for intensive rice cultivation and it has been used as an instance for NDVI tracking and crop phenology studies [12]. The extracted data can support downstream tasks like agricultural monitoring, model training, and vegetation pattern analysis.

## III/ Methods

### 1. Project microservice architecture

The project codebase was organized into separated functional packages under a root namespace *"usth.m1"* including:

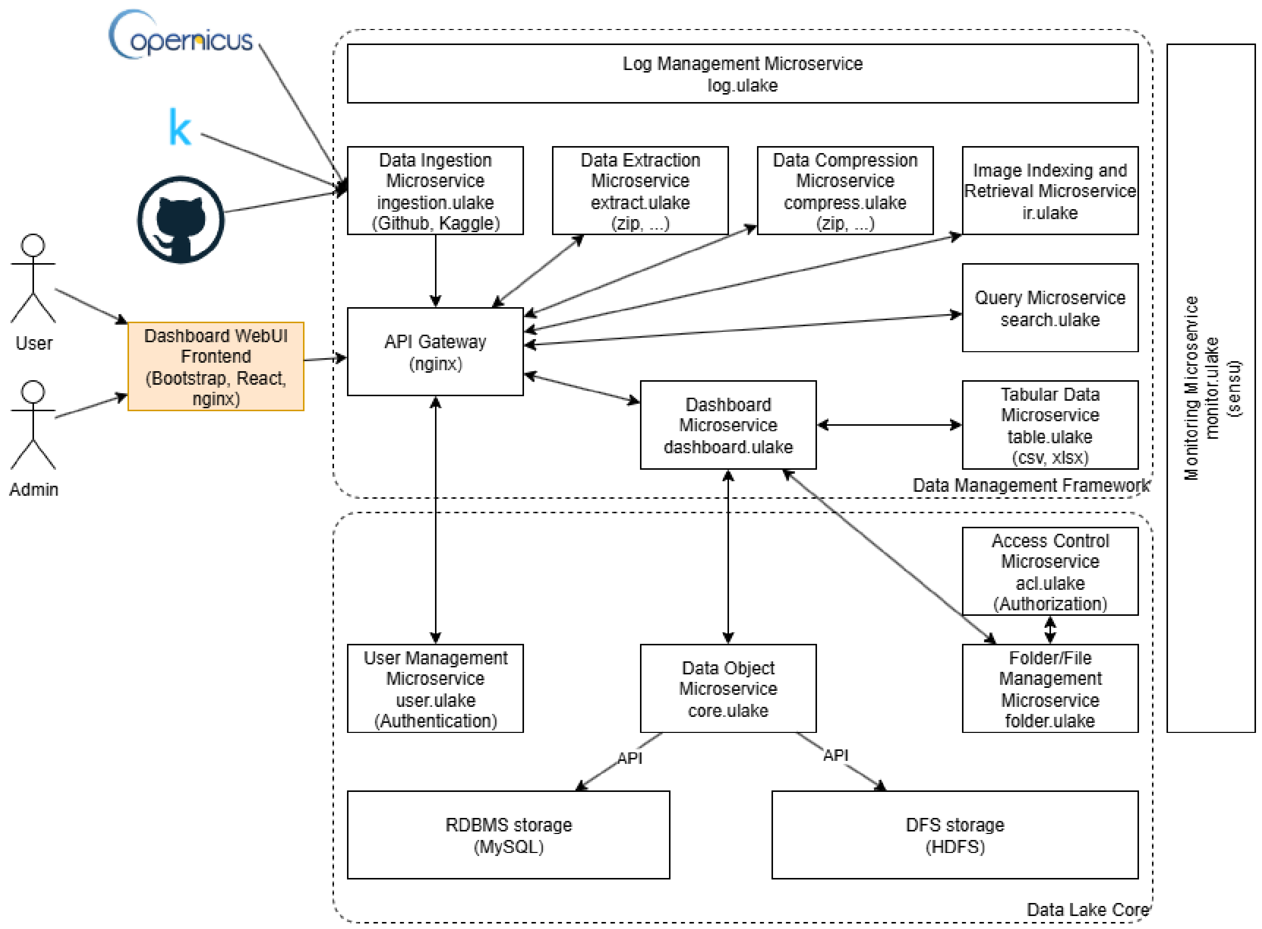
- *entity*: Data Access Objects (DAO) contain data structures that map to PostgreSQL tables, particularly for storing satellite metadata.
- *model*: Data Transfer Objects (DTO) represents request/response bodies used in REST interactions.
- *proxy*: Defines REST client interfaces for external services such as the Copernicus APIs and ULake platform.
- *repository*: Manages database operations by utilizing Panache and PostgreSQL to store and query metadata records.
- *resource*: Acts as the REST entry point to the microservice, exposing API endpoints to trigger metadata extracting, image downloading, and data uploading.
- *service*: Business logic to chain catalog queries, process API calls, GDAL transformations, and file uploads to ULake.

This separation follows the single responsibility principle which makes the system easier to test, maintain, and scale within the boarder ULake platform.

### 2. Data Lake Architecture

As mentioned in the introduction, the ULake system, developed and maintained by the ICT Lab, is an example of an on-premises data lake designed to provide local, secure, and cost-efficient storage for high-volume scientific datasets. The primary motivation for adopting an on-premises architecture is to ensure data locality, access control, and integration with local compute infrastructure, which is often required in academic and governmental research environments where sensitive data workflows must be managed without depending on external networks.

The overall structure of the on-premises ULake used in this project follows a modular microservice-based architecture as illustrated in the diagram below:



opernicus
k
User
Admin
Dashboard WebUI Frontend (Bootstrap, React, nginx)
Log Management Microservice log.ulake
Data Ingestion Microservice ingestion.ulake (Github, Kaggle)
Data Extraction Microservice extract.ulake (zip, ...)
Data Compression Microservice compress.ulake (zip, ...)
Image Indexing and Retrieval Microservice ir.ulake
API Gateway (nginx)
Query Microservice search.ulake
Dashboard Microservice dashboard.ulake
Tabular Data Microservice table.ulake (csv, xlsx)
Data Management Framework
Monitoring Microservice monitor.ulake (sensu)
Access Control Microservice acl.ulake (Authorization)
User Management Microservice user.ulake (Authentication)
Data Object Microservice core.ulake
Folder/File Management Microservice folder.ulake
API
API
RDBMS storage (MySQL)
DFS storage (HDFS)
Data Lake Core

Figure 4. ULake architecture.

The system is composed of several interacting layers grouped into two primary blocks including Data Management Framework and Data Lake Core:

Data Management Framework contains:

- Log management microservice (package *log.ulake*) tracks the whole system logging details.
- Data ingestion microservice (package *ingestion.ulake*) connects to external data sources including Copernicus (the result from this project), Kaggle, and GitHub.
- Supporting data ingestion are data extraction (package *extract.ulake*) and data compression (package *compress.ulake*) microservices. These are responsible for data extraction and compression tasks respectively.
- Once data is ingested, it is further indexed and prepared for retrieval through the image indexing and retrieval microservice (package *ir.ulake*).
- The query microservice (package *search.ulake*) is used to search data after processing.
- All tabular data is handled via the tabular data microservice (package *table.ulake*) including CSV and XLSX files.

- The API gateway (based on nginx) acts as the central access point through which both users and admins interact with the system, typically via a dashboard interface served by package *dashboard.ulake*.

Data Lake Core includes:

- User management microservice (package *user.ulake*) manages account authentication to the ULake.
- Data object microservice (package *core.ulake*) connects directly to the underlying storage systems: MySQL for structured RDBMS storage and HDFS for distributed file system storage.
- Folder/File management microservice (package *folder.ulake*) used to manage the logical structure of folders and files is closely attached with access control microservice (package *acl.ulake*) which is in charge of enforcing user permission within the ULake.

Lastly, overall system health is handled by the monitoring microservice (package *monitor.ulake*).

This layered and modular design not only enables cleat separation of responsibilities across services but also ensures extensibility, making the architecture highly adaptable to evolving data needs. This creates a strong foundation for extending the platform through specialized components. One such extension is the focus of this project: an ETL microservice for data ingestion into the data lake which adds a mean of support for satellite image processing and ingestion from CDSE, significantly enhancing ULake's ability to serve scientific and analytical work.

## 3. ETL microservice flow

### *a) Use-case diagram*

Generally, the ETL microservice pipeline developed in this project consists of three main functional components: data extraction, data transformation, and data loading, which operates on satellite imagery provided by the CDSE aggregating data from multiple Sentinel missions namely Sentinel-1, Sentinel-2, Sentinel-3, and Sentinel-5P. The process begins with data extraction which includes two sub-processes: source authentication to acquire access tokens for interacting with CDSE APIs, and local save used to download metadata and images. The extracted data is then moved to the data transformation phase where preprocessing steps are applied. Finally, data loading is responsible for ingesting the processed data into the ULake. This last phase includes data lake authentication obtaining tokens to access to the ULake. Below is the use-case diagram which the project based on:

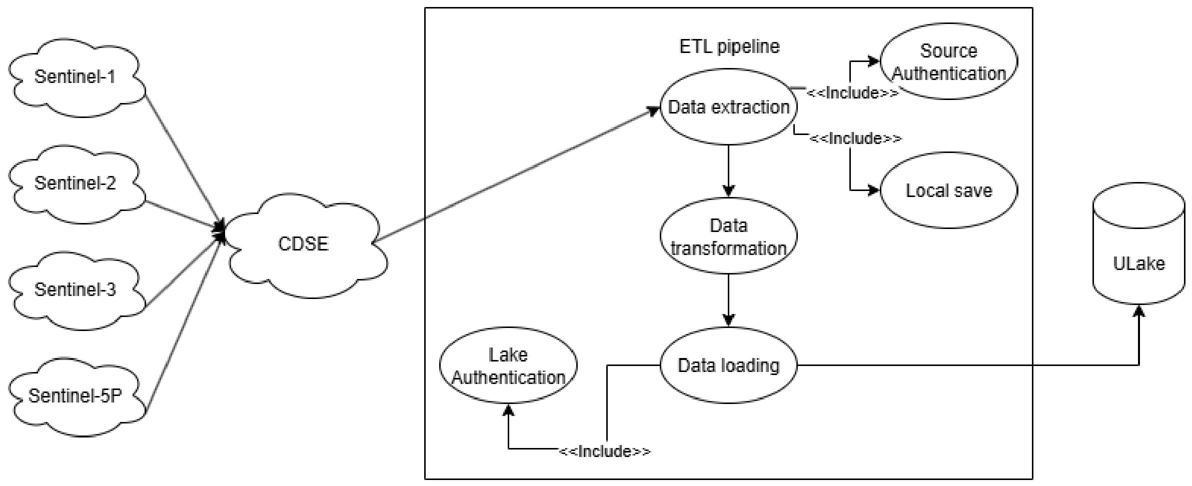


Figure 5. ETL use-case diagram

To be more specific about each component of the graph, five sequence diagrams along with detailed description are shown thoroughly in the next part.

### *b) Authentication process*

In order to secure the communication between the data source and any external application, CDSE and ULake both require authentication for mostly all requests. Therefore, proper authentication processes must be implemented. In general, when a user or a scheduled process initiates an access request to the auth resource component which is a REST API endpoint, the request is forwarded to the auth service and then the auth proxy to encapsulate the business project before communicating with the server. After that, a POST request is sent to the authentication endpoint of CDSE or ULake including user credentials in the request body. The server validates the credentials and if successful, an access token is passed back to the user or the invoking process via the auth service and the auth resource sequentially through the response body.

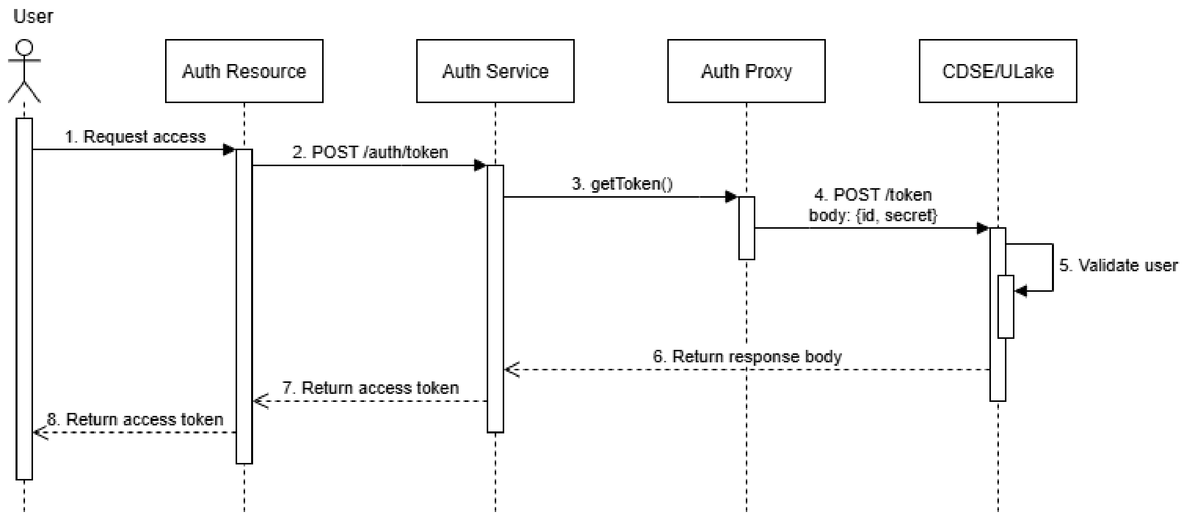


**Figure 6. Authentication process**

### *c) Extract process*

Once the access token is successfully retrieved through the authentication process, the ETL microservice can proceed with data extraction which is divided into two sub-processes including metadata extraction and image retrieval.

With regard to extracting metadata, this operation is carried out using CDSE's Catalog APIs. When a user or a scheduled job sends a search request, it is forwarded by the catalog resource as a POST request to the catalog service where the *searchCatalog()* method is invoked. The catalog proxy as a REST client then sends the request to the corresponding CDSE endpoint with the access token and search parameters. After receiving, the server queries its catalogs for matching satellite scenes and returns a list of corresponding metadata which is passed back through the proxy and service layers to the user. Additionally, in the catalog service after getting the result metadata, the layer supports functions to save the information to the local PostgreSQL database as an outcome tracking and backup method. To do this, the metadata service is used as an intermediate layer to forward the insertion request to the local storage. Once the data is written, a success message is propagated back to the user through service and resource components.

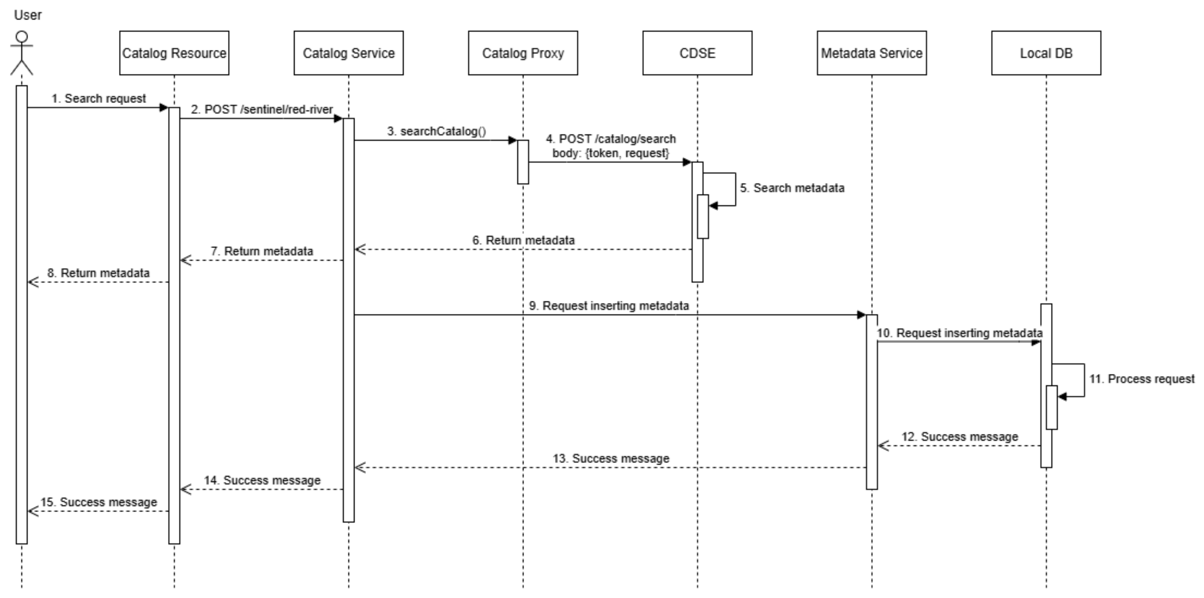


**Figure 7. Metadata extraction process using Catalog APIs**

Following the metadata extraction and using Process APIs from CDSE, the imagery retrieval phase can proceed by iterating through the metadata list previously extracted. To be more precise, after getting the metadata records through Catalog APIs, a user or a job can send a download request by the process resource which is received and handled by the process service and the process proxy before reaching the CDSE server with a POST request containing the required token and query parameters. Upon a successful response, a required image is returned and saved to the local machine by the service layer. Subsequently, the process service calls the metadata service again to update each metadata record by appending the local file path to the corresponding image, which ensures the traceability for each image item. Once this operation is finished, a final success message is returned to the user to wrap up the extraction phase of the ETL pipeline.

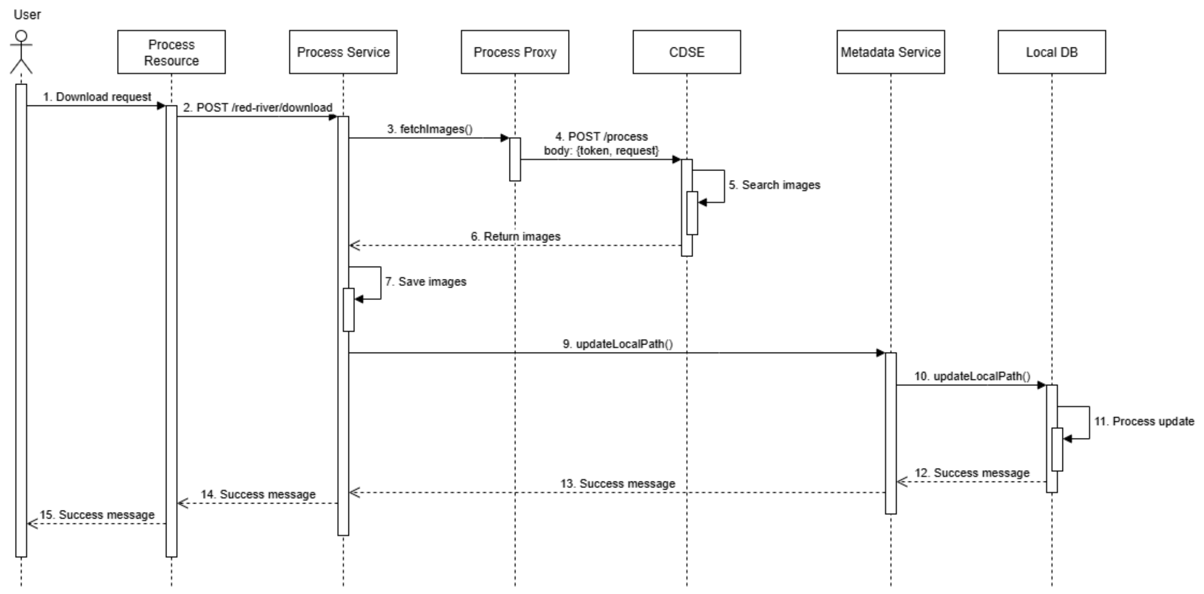


Figure 8. Imagery extraction process using Process APIs

### *d) Transform process*

After the extraction process, the ETL pipeline proceeds to the transformation phase for those locally downloaded images before the loading step. This phase deals with the challenge of raw image size exceeding the current upload threshold set by the ULake which is 1MB per file. In order to resolve the issue, the system implements a down-sampling technique using the gdal_translate tool to reduce the spatial resolution of input images to 30% compared to the original dimensions. The algorithm behind this data-preprocessing technique bases on a default resampling method which is the nearest neighbor resampling. For each pixel in the down-sampled output image, the algorithm identifies the closest corresponding pixel in the original image to assign its value directly which is computationally efficient while still preserving original spectral value [13]. This transformation process ensures that raw TIFF images are efficiently stored and compatible with the file size limit from ULake.

### *e) Load process*

Finally, the ETL microservice ends with the loading process which is responsible for uploading transformed images to the ULake. Similar to the extract phase, this step can only happen with the access token received from the data lake through the authentication process. This stage is divided into two sub-processes: folder creation and file uploading.

In terms of the file creation step which is used to organize the logical folder structure, when a user or an automated workflow triggers a creation request, the folder resource receives and forward it to the folder service and folder proxy sequentially. Then, the proxy communicates with the ULake folder endpoint via a POST request including the access token and the folder parameters. If the folder is created successfully, the server returns its identifier to the user through the service and the resource layer. This folder ID is later used as a reference to assign the destination path for the file uploading in the next step.

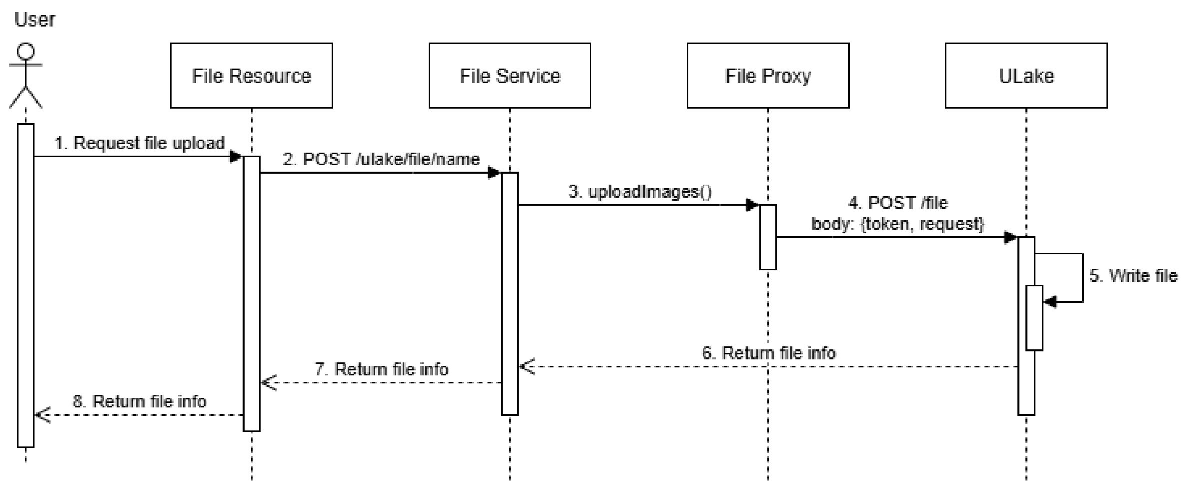


**Figure 9. Load process for creating ULake folders**

Following folder creation, the second sub-process in the loading phase focuses on transferring locally downloaded files to the data lake. Due to current limitations of ULake where the data object microservice is not working properly with the RDBMS storage, only binary file upload is supported, which means this operation is currently limited to true-color and raw TIFF images. Generally, in this implementation, true-color images are uploaded directly due to their relatively small file sizes (under 1MB) while raw images must be transformed by down-sampling to reduce their size below the threshold before entering this phase.

To be more specific about the file uploading flow, when user or a job sends a request using the file resource, the file service carries out business logics before invoking the file proxy to communicate with the ULake API endpoint through a POST request containing the access token, the binary file, the parent folder ID along with other parameters. After successfully creating the file entry in the system, ULake sends a response to the user notifying about the detailed file information just created in the lake This final step wraps up the ETL pipeline.

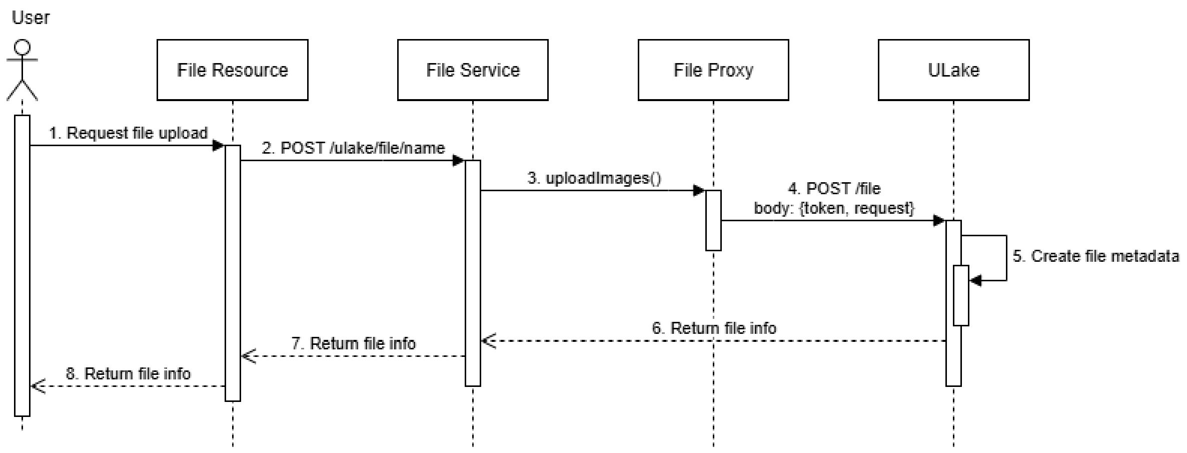


Figure 10. Load process for uploading files

# RESULT AND DISCUSSIONS

## 1. Result

After a three-month internship at the ICT lab, this project has achieved numerous progressive results. First and foremost, the implementation of the ETL microservice was successful using Quarkus framework. The microservice supports end-to-end data ingestion into a data lake, from CDSE as the data source to ULake at ICT lab as the destination data storage platform. The interaction between the two endpoints is consistent via a secure authentication process which authorizes continuous data retrieval and uploading through external REST APIs. Generally, the project offers a way to transfer data from the source, through a preprocessing step, to the destination repository.  In this project, the specific region of interest is the Red River Delta of Vietnam during a one-year interval from the first of January, 2023 to the first of January, 2024, which is tracked by the Sentinel-2 L2A satellite from CDSE.

In the first step of this ETL pipeline, the extraction process is divided into two sub-components including the metadata retrieval and the image downloading. Using the Catalog API from CDSE, the system successfully fetched metadata records from the source and saved them into a local PostgreSQL database for tracking purposes. Then, based on the retrieved metadata list, the Process API was utilized to download corresponding imagery to a local machine. This process worked with two types of images namely true-color PNG images for visualization and raw TIFF images containing twelve spectral bands for further scientific use. The one-year data download time of the former was recorded at more than 6.5 minutes for 456 files of 139MB in one API call. For raw TIFF images, since each file was a few MB in size, the download progress was interrupted at ten-minute run time with an unauthorized 401 issue implementing the process API token expired at some point due to the long response time. Therefore, to retrieve all one-year data, two API calls were executed. In the first run, 1.03GB of 225 files was downloaded for approximately 7 minutes while the second run got 1.05GB of 231 images in 8 minutes, making a total of 16 minutes for 456 images weighting 2.08GB.

After the first phase, the extracted data is pre-processed in the transformation step. Due to current file upload constraints of the ULake, specifically 1MB upload limit per file, a down-sampling method was applied to raw TIFF images which usually exceed the size limit. This phase made use of the gdal_translate tool with the nearest-neighbor resampling algorithm to support in reducing image resolution by 70%, from 512x512 pixels to 153x153 pixels while still preserving all twelve spectral bands. In terms of the response time, the complete transformation of 2.08GB images was achieved within a minute. Below is an example to compare an image before and after the transformation:

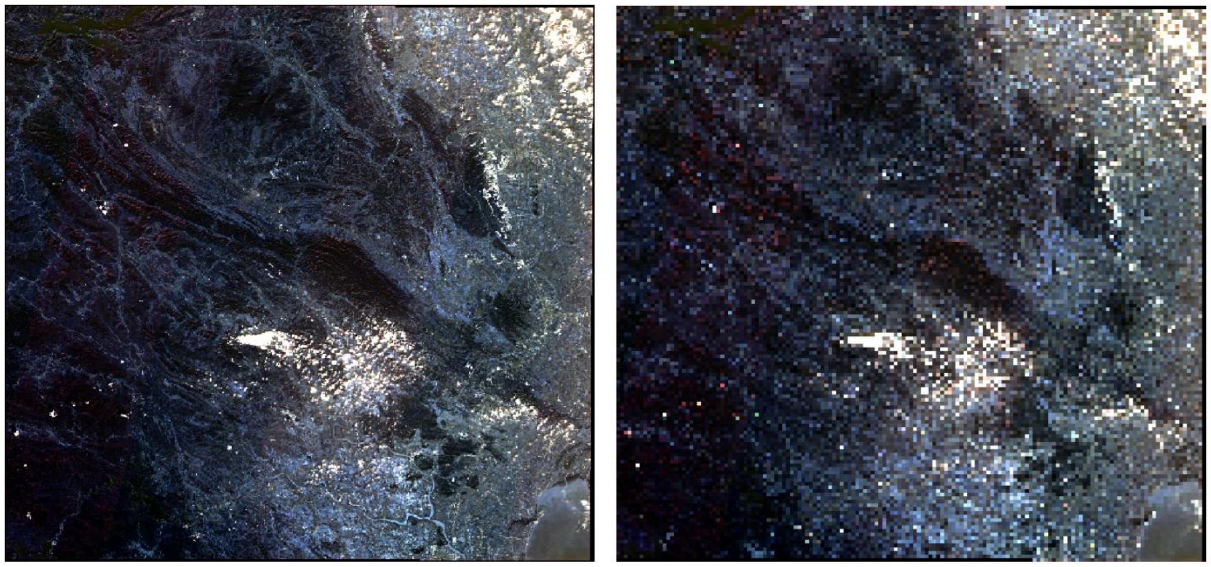


**Figure 11. A TIFF image before and after the transformation process**

The final phase of the pipeline is the loading process which was responsible for working with folder and file microservices on ULake to manage logical organization and to upload relevant data to designated folders. Due to some current issues with the ULake's RDBMS metadata storage, only binary file upload is supported. The time required to upload images to ULake was quite similar to the downloading time, which was around 4 minutes to upload 456 true-color files of 139MB in one API call and about 12 minutes for the same number of raw images weighting 2.08GB in just one API request.

## 2. Discussion

The successful development of the ETL microservice during the internship period has shown a practical way to build a simple and extensible data ingestion pipeline using Quarkus. The system effectively integrates an external data service, CDSE, with an on-premises data lake, ULake, which provides an ETL pipeline structure for satellite imagery. The microservice pattern used in this project to clearly separate authentication, catalog querying, image retrieval, data preprocessing, and file uploading not only improves the maintenance ability but also allows future scaling with additional data sources or different regions. In terms of the performance, the continuous extraction process over the Red River Delta in Vietnam within a year for true-color images and six months for full-band color images shows that the system can manage hundreds of metadata records and images at once. In addition, the transformation phase provides a fast, simple, and effective down-sampling method leveraging the nearest-neighbor resampling algorithm form the gdal_translate tool. The last phase of the pipeline works well as the system was able to organize and upload binary files of a whole year period without any difficulty.

In terms of limitations, the pipeline has to face with several critical issues such as CDSE authentication token expiring within a ten-minute run, shortage of other data transformation methods like deduplication or the ability to avoid blank images, inability to store image metadata on the data lake, and no automation process. All of these problems must be addressed and resolve in future work.

# CONCLUSION

This report has presented the design and implementation of an ETL microservice dedicated to ingesting satellite imagery from CDSE to the on-premises ULake at the ICT lab. The proposed microservice enables an end-to-end ingestion workflow including metadata and image extraction, raw TIFF image transformation, and binary file uploading. The microservice was built using the Quarkus framework that supports secure authentication and interaction with both CDSE and ULake APIs. Through the testing over the Red River Delta region in Vietnam during 2023, the system successfully extracted over 450 metadata records and their corresponding true-color and raw images. It also effectively performs down-sampling on full-band-color images to reduce their size by 70% while still preserving all twelve spectral bands. The transformed data was then logically structured and uploaded to ULake using the existed folder and file microservices.

While the system achieved its intended goals, this project still requires numerous future works. Firstly, a fallback method must be implemented to avoid the CDSE authentication token expiring. In the transformation phase, several other data-preprocessing methods must be developed to avoid deduplication and valueless blank images from the data source if any. Furthermore, a schedule process must be set up so that the system can perform the data ingestion daily to reduce the manual effort. Finally, there should be a proper logging system to keep track of detailed progress in each phase of the ETL pipelines.

To sum up, the project contributes a reusable and scalable solution for Earth observation data ingestion, aligning with both academic exploration and practical data engineering practices. This work can be helpful in strengthening ICT lab's data infrastructure and improving the ability to do researches on climate modeling, agricultural analysis, and environmental AI-based forecasting models.

# REFERENCES

[1] *Reinsel, D., Gantz, J., & Rydning, J. (2017). Data age 2025: The evolution of data to life-critical don't focus on big data. Framingham: IDC Analyze the Future.*

[2] *Abdalla, H. B. (2022). A brief survey on big data: technologies, terminologies and data-intensive applications. Journal of Big Data, 9(1), 107.*

[3] *Siddiqui, M. R. (2024). Big Data vs. Traditional Data, Data Warehousing, AI, and Beyond.*

[4] *Harby, A. A., & Zulkernine, F. (2025). Data lakehouse: a survey and experimental study. Information Systems, 127, 102460.*

[5] *Dinesh, L., & Devi, K. G. (2024). An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. Journal of Cloud Computing, 13(1), 12.*

[6] *Swapnil, G. (2020). ETL in Near-real-time Environment: A Review of Challenges and Possible Solutions. ResearchGate. com.*

[7] *Newman, S. (2021). Building microservices: designing fine-grained systems. " O'Reilly Media, Inc.".*

[8] *Escoffier, C., & Finnigan, K. (2021). Reactive Systems in Java. " O'Reilly Media, Inc.".*

[9] *Stonebraker, M., & Kemnitz, G. (1991). The POSTGRES next generation database management system. Communications of the ACM, 34(10), 78-92.*

[10] *Thenkabail, P. S., Lyon, J. G., & Huete, A. (Eds.). (2018). Fundamentals, sensor systems, spectral libraries, and data mining for vegetation. CRC Press.*

[11] *Aschbacher, J., & Milagro-Pérez, M. P. (2012). The European Earth monitoring (GMES) programme: Status and perspectives. Remote Sensing of Environment, 120, 3-8.*

[12] *Nguyen, D., Wagner, W., Naeimi, V., & Cao, S. (2015). Rice-planted area extraction by time series analysis of ENVISAT ASAR WS data using a phenology-based classification approach: A case study for Red River Delta, Vietnam. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 40, 77-83.*

[13] *Baboo, S. S., & Devi, M. R. (2010). An analysis of different resampling methods in Coimbatore, District. Global Journal of Computer Science and Technology, 10(15), 61-66.*