Project Overview

Migration from CDL to Arcus. CDL is built on Hadoop on cloudera.

Reasons:

- To reduce the maintenance cost paid to the cloudera platform.
- To make the changes in the data: Currently the data is stored in parquet and apache HUDI. It is more restricted, as it was difficult edit the data.

MinIO

It is like a software that gives object store where ant type of data can be stored like files, folder, images, videos, anything

Object = Data + Metadata + Unique ID

One can run it in cloud or even on laptop.

Rancher

A tool used helps manage the Kubernetes clusters. Kubernetes is powerful but it can be complex to set up, monitor and control. Rancher makes Kubernetes much easier by giving you a dashboard, tools and controls to handle many clusters from one place.

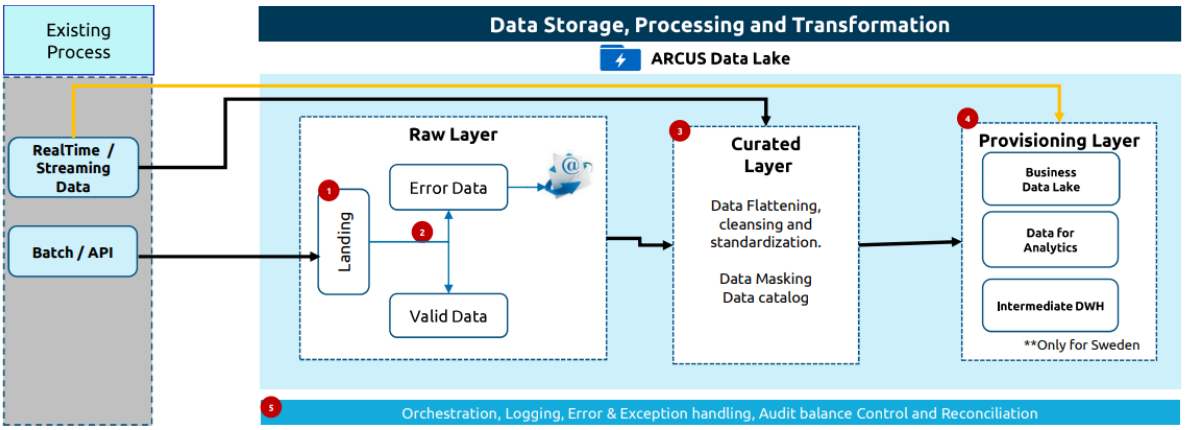
Note: MinIO and rancher are used separately. They are not tightly integrated; they don't depend on each other to function. Each tool does its job independently and they can be upgraded, scaled or replaced without affecting the other.

When tightly coupled compute depends directly on storage been present on the same machine or system. When the job starts data must be directly available in the same node or local disk, entire data may need to be copied locally before computation so therefore the job may fail if the data fails. When decoupled compute can freely pull data over the network whenever it needs it.

Types of data

- B2B – Transactions and interactions between two businesses. The data must be in parquet format as it is stored in S3.
- B2C - Transactions between a business and individual consumers.

Data layers in Arcus



Types of data

- B2B – Transactions and interactions between two businesses. The data must be in parquet format as it is stored in S3.
- B2C - Transactions between a business and individual consumers.

Data layers in Arcus

STEP 1: Ingestion Services (Ingress)

Collect data from various internal and external systems.

Sources:

- Batch files (MFT/SFTP): Periodic file drops, CSVs, flat files, logs, etc.
- Kafka (Message Queue): Real-time event streams, IoT data, logs.
- API (Microservices): Applications directly send real-time data through APIs.

STEP 2: Data Capture (RAW Layer)

Capture incoming data as fast as possible without modifying it → "Landing zone."

Tools Used:

- NiFi (Event Data Ingestion): Automates data flow. Handles both batch and streaming ingestion from MFT.
- Apache Spark (Batch Data Ingestion): For larger datasets that come in bulk. Reads files from APIs → processes them in Spark jobs.
- MinIO (Raw Zone Storage): Object storage that holds raw, untouched data. Acts like AWS S3 but self-managed. This is your Data Lake → stores all raw data for historical reference and auditing.

The landing data will have error data and valid data. The valid data is sent for next layer.

STEP 3: Data Curation (Curated Zone)

Clean, validate, and prepare data for analysis.

Tools Used:

- Spark Structured Streaming (Stream-Event Processing): Processes Kafka streams. Does filtering, aggregation, enrichment on real-time streams.
- Spark (Data Curation & Quality Engine):
- Handles batch curation. Remove duplicates. Apply business rules. Data standardization and Schema validations.
- MinIO + Apache Hudi (Curated Zone Lakehouse):
- Apache Hudi provides transactional capabilities over object storage. Supports upserts, deletes, incremental queries. Data here is cleaner, more structured than raw zone.

STEP 4: Data Provisioning (ETL/ELT)

Transform curated data into business-ready datasets.

Sub-steps:

a. Batch ETL (Spark + Airflow + Kubernetes)
- Heavier ETL jobs.
- Resource-intensive transformations.
- Spark reads curated zone → applies transformations → writes to provisioned zone.

b. Storage:
- Provisioned Zone (MinIO + Hudi): More structured, final datasets.
- Intermediate Data Warehouse (Postgres): Temporary staging layer for business view creation.
- Business View Data Warehouse (Postgres): Fully curated, business-consumable data for reporting.

c. Low Latency DB (Postgres):
- Real-time data for operational queries.
- Quick response to API calls or dashboards.