Hi everyone, today I’ll Walk you through our **Streaming Data Ingestion architecture** this is the logical design we’re implementing to build a **real-time data ingestion pattern for our Lakehouse using Confluent Cloud**.

The goal is to continuously ingest, enrich, and persist data from multiple enterprise sources into Iceberg tables for analytics.

**1️⃣ Data Ingestion Layer**

Let’s start from the **left side** of the diagram.

We have multiple data sources —

* **Databases** that send changes via **Change Data Capture (CDC)**,
* **Google Cloud Storage (GCS)** for **historical and vendor files**,
* **Oracle** as an enterprise system of record, and
* **APIs** for operational or external data feeds.

Each of these is connected using **Confluent Source Connectors**:

* **GCS Source Connector** for batch and historical loads,
* **Oracle CDC Connector** for real-time database changes,
* **HTTP Source Connector** for APIs.

All the incoming data flows into the **Kafka Cluster** on **Confluent Cloud**, which is our core messaging backbone.

This is where data streams are standardized, and any required **Single Message Transformations (SMTs)** are applied for cleansing or enrichment.

**2️⃣ Data Processing & Enrichment**

Once data lands in Kafka, it is organized into **topics**.

At this stage, we have the flexibility to introduce **data enrichment** — this is marked as *“Open for Options”* in the architecture.

For example, enrichment could be performed using **Flink SQL pipelines** or **Kafka Streams applications** to join reference data or apply business logic.

All message schemas are registered and version-controlled in the **Schema Registry**, ensuring consistency and compatibility across producers and consumers.

**3️⃣ Data Storage & Consumption**

From Kafka topics, data moves downstream via **GCS or Iceberg Sink Connectors**.

These connectors convert enriched JSON data into a **columnar Iceberg table format**, which is more efficient for analytics and long-term storage.

We then integrate with a **Data Catalog** and **Hive Metastore** so that these Iceberg tables are discoverable and queryable.

Analytics users can access the data through **Starburst**, enabling **near-real-time querying** and insights generation.

We also have an **interim JSON-to-Iceberg solution** in place for early validation while we mature the end-to-end connector flow.

**4️⃣ Scope Overview**

Now, looking at the **legend** and color codes:

* The **dotted box** defines the **MVP scope**, which includes ingestion through CDC and GCS connectors, Kafka topics, schema registry, and Iceberg sink integration.
* The **yellow boxes** represent items **waiting on network or DNS resolution**, such as Oracle and HTTP connectors.
* The **orange area** is **out of scope for MVP**, mainly the external API ingestion.

So our MVP primarily focuses on establishing a **working ingestion-to-Iceberg pipeline** with **CDC and GCS data sources**, catalog integration, and downstream analytics through Starburst.

**5️⃣ DevOps & Infrastructure Automation**

From an engineering standpoint, we’re also building **DevOps automation** for both **application and infrastructure layers**:

* For **application DevOps**, we use **CI/CD pipelines** that automate deployment of connectors, schemas, and Flink jobs via Terraform modules and YAML configs.
* For **infrastructure DevOps**, the entire **Confluent environment, topics, and service accounts** are provisioned and managed through **Infrastructure as Code (IaC)** using **Terraform**, ensuring consistency across regions and environments.

Monitoring and observability are integrated through Confluent metrics and log aggregation for pipeline health tracking.

**6️⃣ Closing Summary**

So, in summary —

this architecture delivers a **scalable, real-time ingestion framework** on Confluent Cloud,

supports both **batch (historical)** and **streaming (CDC)** use cases,

and enables **near-real-time analytics on Iceberg tables**.