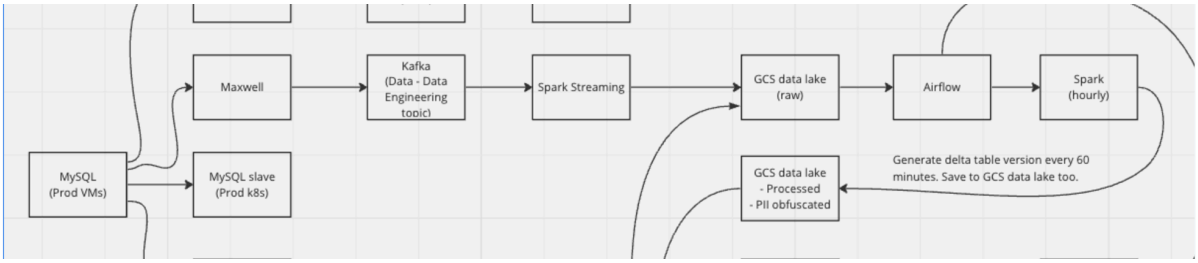
Problem 1: PII obfuscation in Airflow

a) Deploy this whole pipeline (microbatching/streaming) in GKE



Stack to be used:
Terraform/Terragrunt
Airflow
Maxwell/Debezium
Mysql
Spark
Kafka

b) Implement PII obfuscation Given: DB schema named `authentication`, two of its tables are shown below.

`users`

id	username	metadata	created_at
1	kat@email.com	{“secret”: “gQTKNMafpw”, “provider”: “google-oauth2”}	2023-09-01 08:01:02
2	mani@email.com	{“secret”: “sjmaIS2EmA”, “provider”: “basic-auth”}	2023-09-01 08:01:03

`groups`

id	name	description	created_at
1	SUPER_USER	Full access to all functions.	2015-01-01 04:05:06
2	DEFAULT_USER	Initial role assigned to a new user.	2015-01-01 05:06:07

1. Create the test tables as .parquet files in Google Cloud Storage (GCS)

    a. Exact values do not matter, feel free to change them. But maintain the table structure.
2. Create a Kubernetes configmap with fields that contain personally identifiable information (PII) for `authentication` schema
    a. `users` table has `username` and `metadata.secret` as PII fields
    b. `groups` table does not have any PII fields
3. Deploy Airflow on Kubernetes
4. Create an Airflow task that masks PII fields and generates new tables.
    a. Refer to the configmap for the list of fields.
    b. Assume they may be updated at any time.

`users_masked`

id	username	metadata	created_at
1	06ee1f38a053ae0a438ae546c40e3a6b0383e4de1167028f5cc3790f9762c005	{“secret”: “4c4fcedd8d581e8f2019bb3b3bafe5ae0ddabf998ae1f463725b81d01cf82e42”, “provider”: “google-oauth2”}	2023-09-01 08:01:02
2	805725326106b10a3c271a0f495bfef3790b4e53ebcbb0b60b88cc30a621a20f	{“secret”: “b62d11f552c80844db9946b8863d6f776e8e083d97c66bdb27b8ea7cd5f6f4ba”, “provider”: “basic-auth”}	2023-09-01 08:01:03

`groups_masked`

id	name	description	created_at
1	SUPER_USER	Full access to all functions.	2015-01-01 04:05:06
2	DEFAULT_USER	Initial role assigned to a new user.	2015-01-01 05:06:07

Problem 2. Trino

Given: The same DB schema named `authentication` with tables `users` and `groups`

1. Deploy Trino on Kubernetes.
2. Query the parquet tables from GCS.
3. Append more records to the tables.
4. Show that the additional records can immediately be queried in Trino.
5. Convert the parquet file to a delta file and query it with Trino.