1.Data Scources:

-Cloud SQL:

a. retailer-mysql-db

b. supplier-mysql-db

c. api\_reviews

Password - 12345

-Add Databases:

a. retailerDB

-Add users

myuser

mypass

-Add connections

In network add 0.0.0.0/0

-Save all changes

-Go to Cloud SQL Studio

Select the Databases we created Use the credentials as we created i.e. myuser and mypass and copy paste the table creating

queries from vs code

-Add Databases:

b. supplierDB

-Add users

myuser

mypass

-Add connections

In network add 0.0.0.0/0

-Save all changes

-Go to Cloud SQL Studio

Select the Databases we created Use the credentials as we created i.e. myuser and mypass and copy paste the table creating

queries from vs code

c. api-reviews:

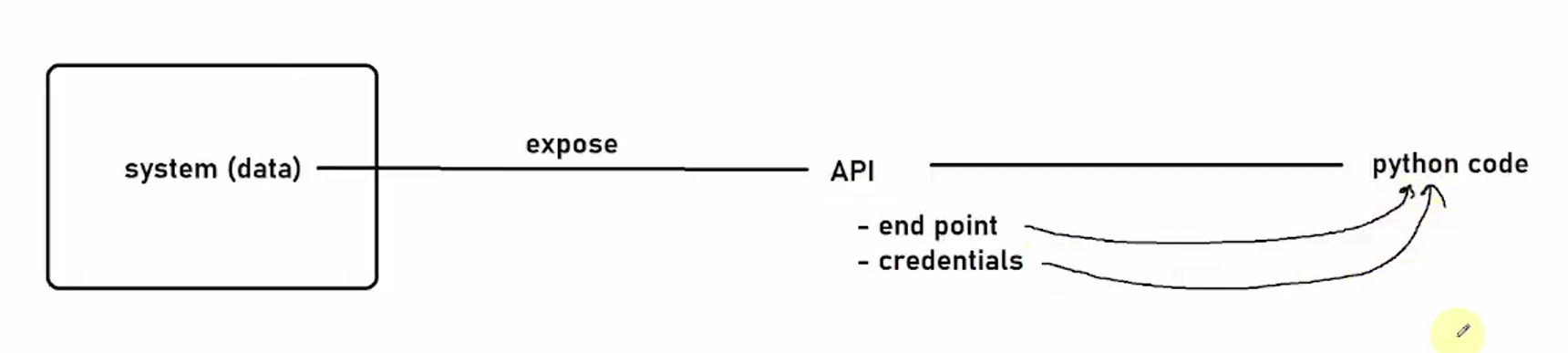
We are using mockapi for generating data as we are not having any app which can generate data hence we use mockapi to generate

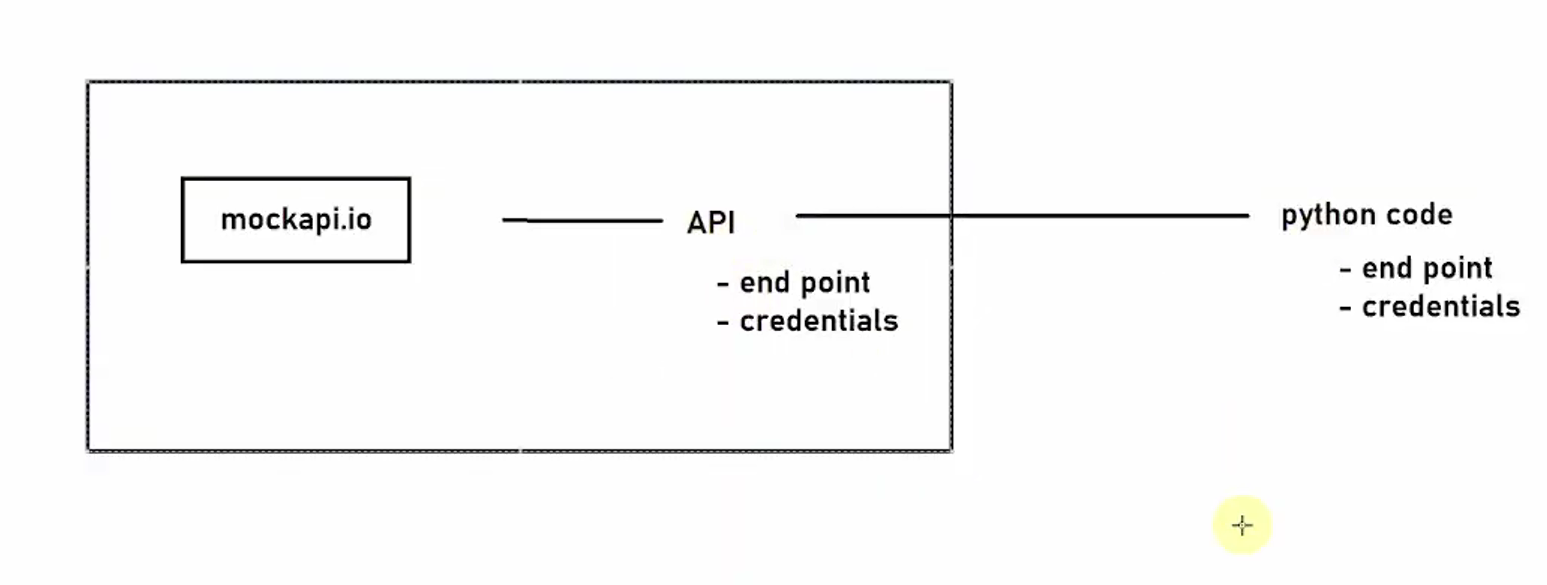
data

mockapi(app) --> api(endpoint,credentials) --> python code(using those endpoint and credentials we are creating python for data

ingestion)

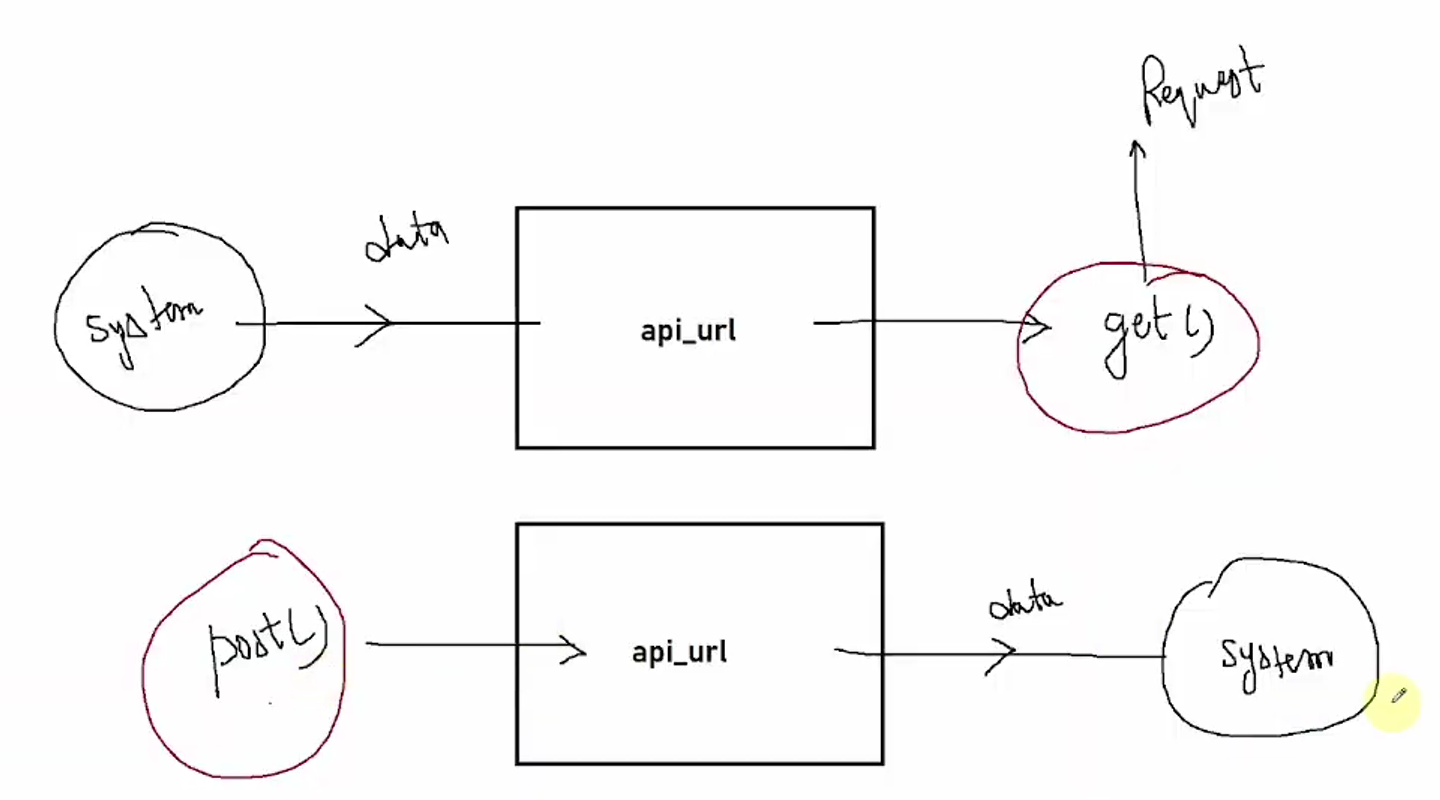
https://67ec0ae1aa794fb3222ca347.mockapi.io/retailer/:endpoint (replace :endpoint with reviews i.e. name of our project)





Below diagram tells about two methods used for getting for pushing the data from an api.

get() method is used to get the data and post() method is used to push the data.



2. Setup GCS:

- create bucket (bucket-name : retailer-datalake-project-01042025)

- create folders such as:

i. configs- any type of congigurations, we are maintaining metadata driven approach

-retailer\_config.csv

-supplier-config.csv

ii. landing- data landed into landing from data sources all 3 sources,

-retailer

-supplier

-reviews

iii. temp

-pipeline-logs

3. Data Ingestion:

-create dataproc cluster

- Enable all apis

- Allow storage admin permission

- Enable Private Google Access i.e. on it

- Paste below into cloud shell so as to create dp cluster

CLUSTER\_NAME="my-demo-cluster"

REGION="us-east1"

gcloud dataproc clusters create ${CLUSTER\_NAME} egion ${REGION} num-workers=2 worker-machine-type=n1-standard-2 worker-boot-disk-size=50 master-machine-type=n1-standard-2 master-boot-disk-size=50 image-version=2.0-debian10 enable-component-gateway optional-components=JUPYTER initialization-actions=gs://goog-dataproc-initialization-actions-${REGION}/connectors/connectors.sh metadata bigquery-connector-version=1.2.0 metadata spark-bigquery-connector-version=0.21.0

Q. Why we read data from config file?

-> -Will give idea that which tables are active and which tables supports what type of data loading weather its full or incremental load.

-If its incremental load then what is watermark.

-Target path

i. ingestion-1: Ingest the data from mysql-retailer-db using config files and creating audit\_tbls,archieve,logging load it into

gcs-bucket/landing/retailer-db/\*

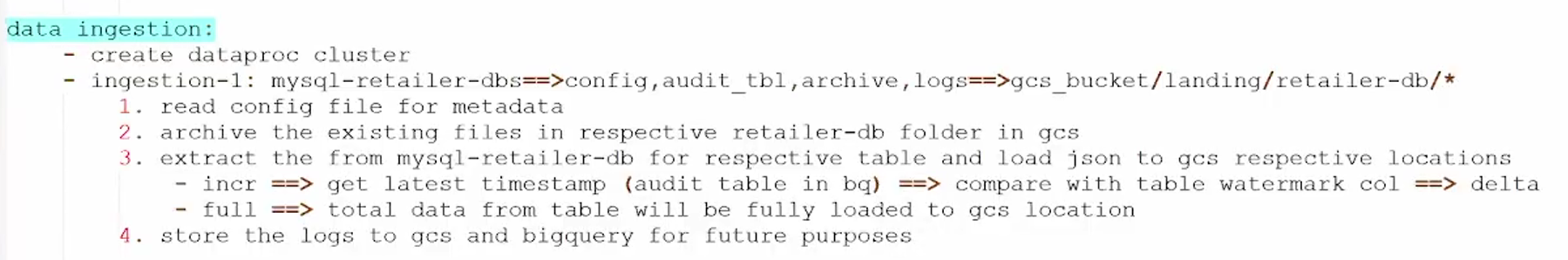
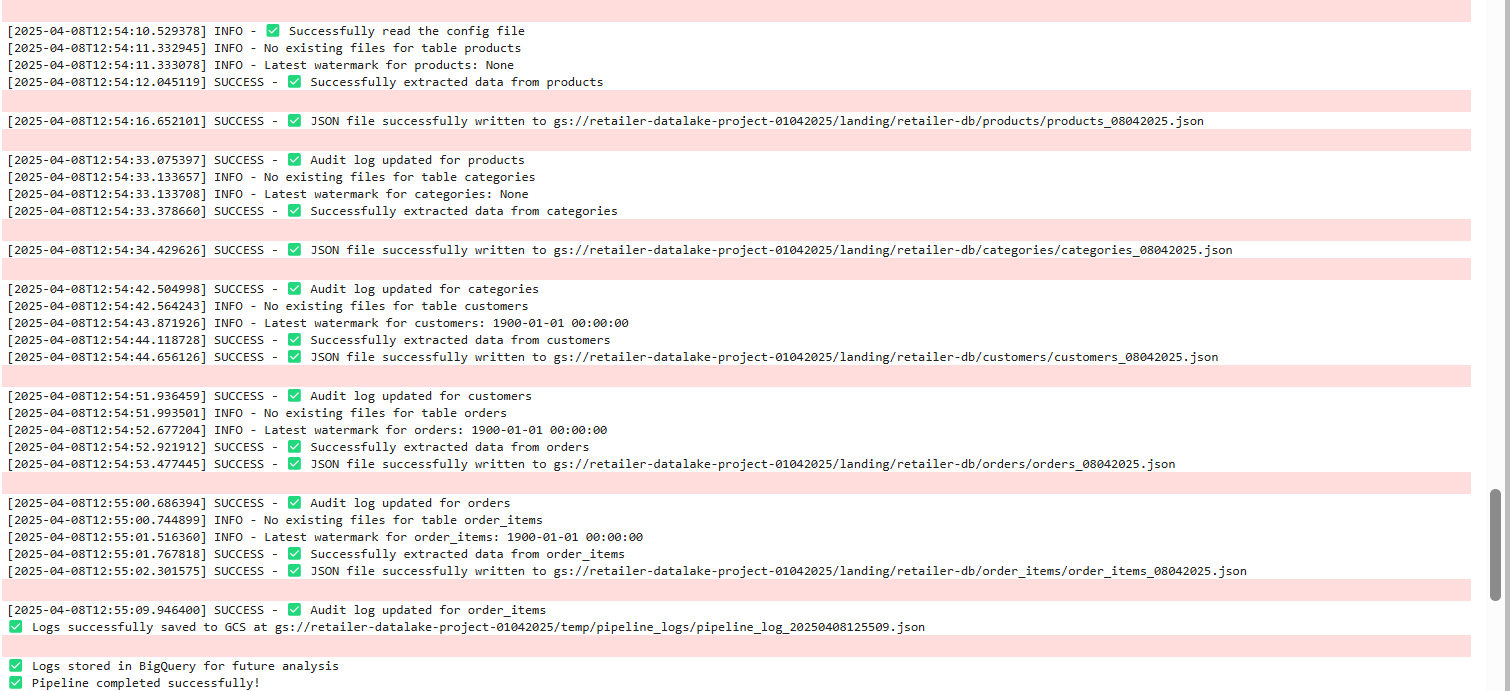


Fig. Ingestion-1

o/p:

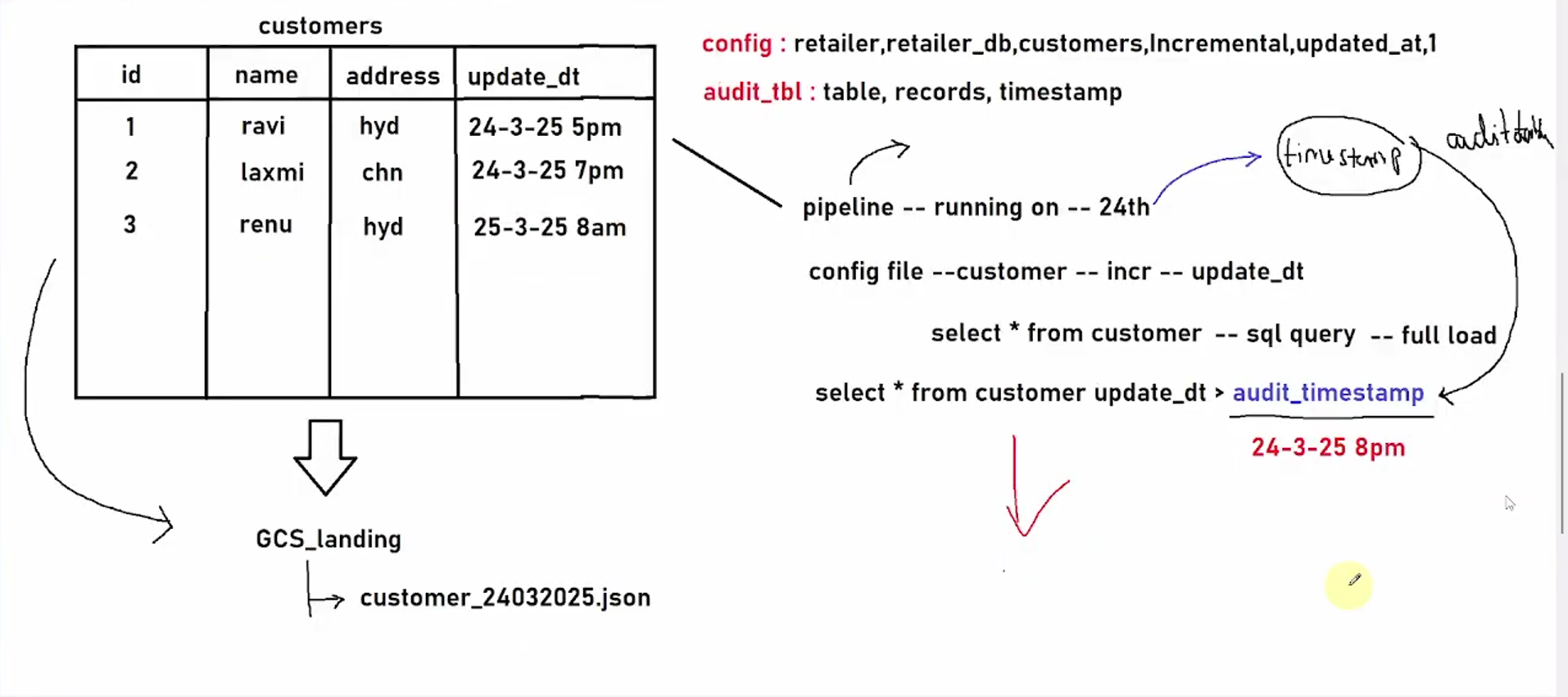


Q. What is audit tables?

-> Audit tables are used to deal with incremental data load if we want to get today records only then the query will be

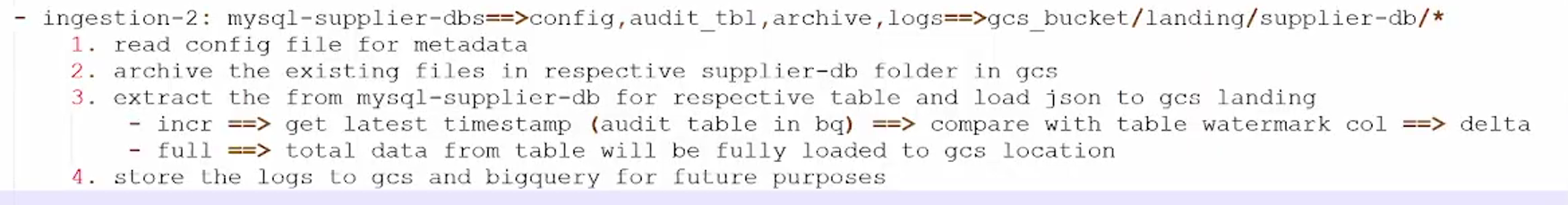
select \* from emp where updated\_at > audit\_timestamp (where this audit timestamp is the time when yesterday the last record was inserted

or last time when the pipeline got ran then will be getting incremental data and hence we use audit tables in our project)

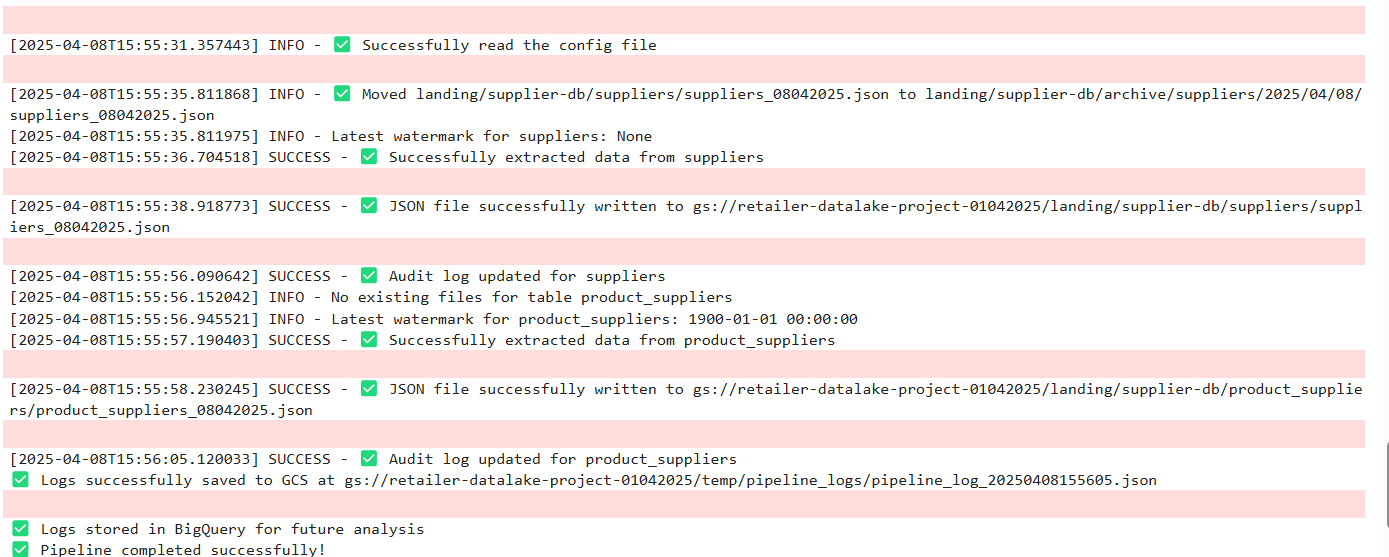


Above diagram tells us about metadata driven approach.

ii. Ingestion-2:



o/p:



iii. Ingestion-3:

reviews\_api 🡪 gcs\_bucket/landing/customer\_reviews/\*

* First fetch the data
* If status code is 200 then convert it into pandas df
* Store this df into local
* From local write to gcs landing

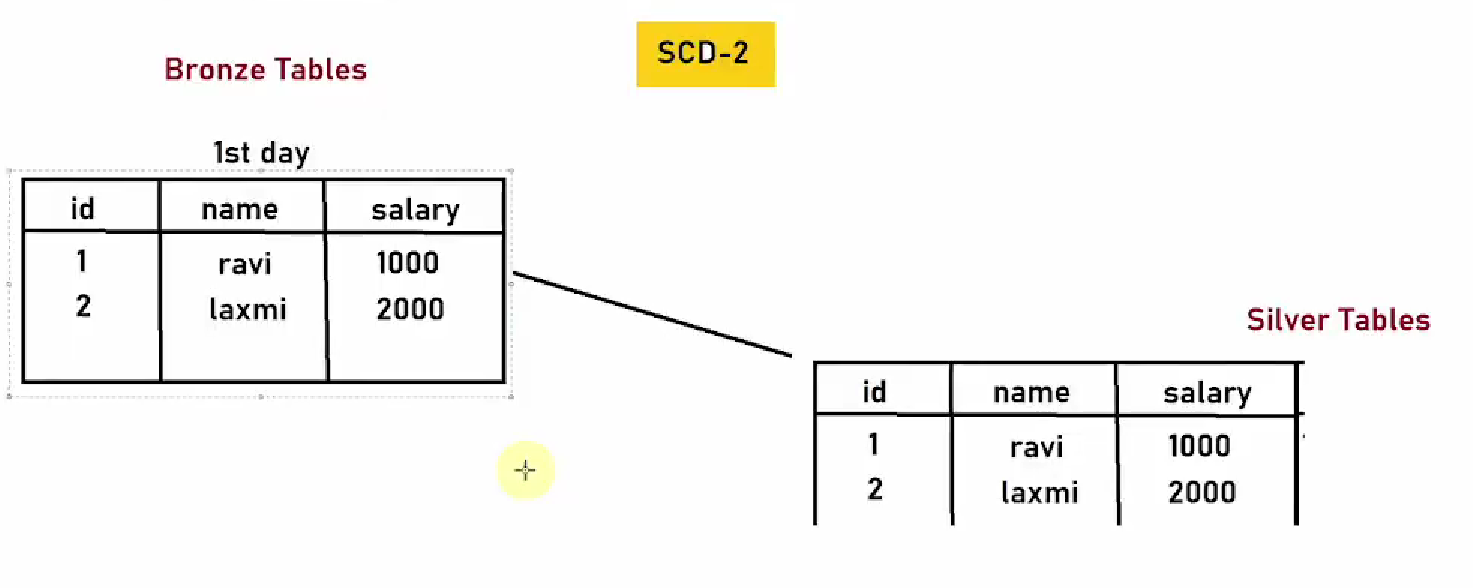
Successfully loaded the data!!!

**4. Big Query:**

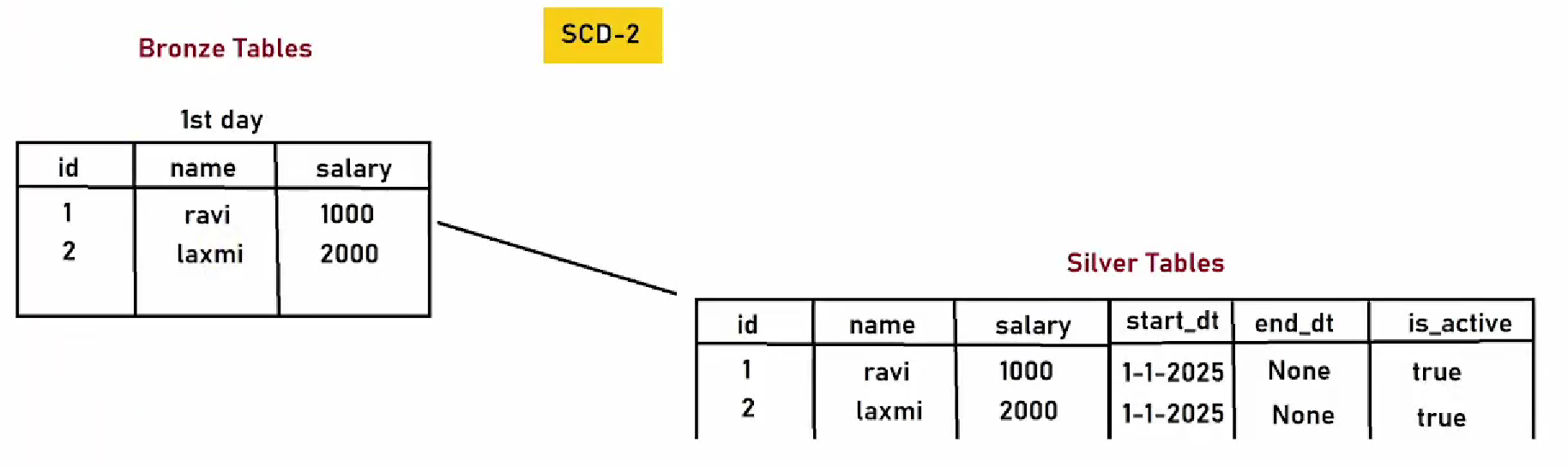
**A. Bronze layer:** Now we have data in GCS so we need to create external tables to take data into bronze layer.

For all 8 tables create external table.

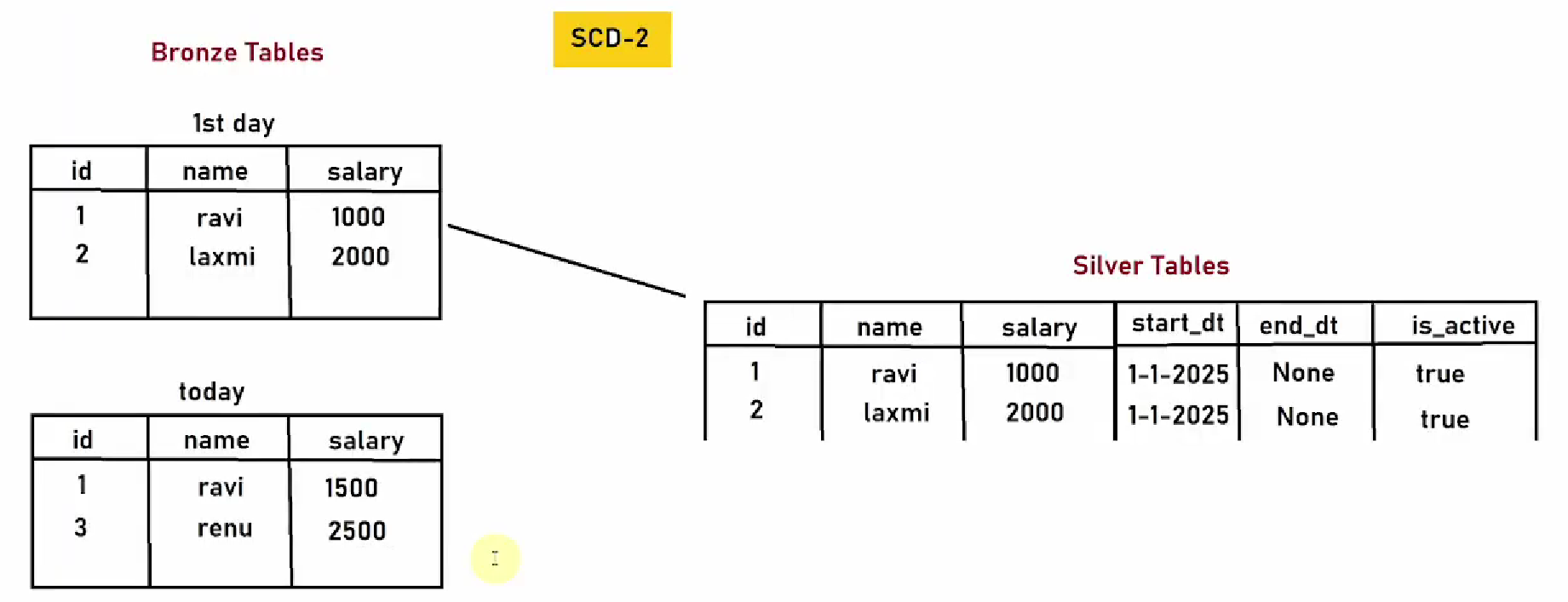
**B. Silver Layer:** Handle Nulls using quarantine, Clean the data, Handle Duplicates and main is maintain scd type 2 if its incr then go for scd2 if full load truncate and load.



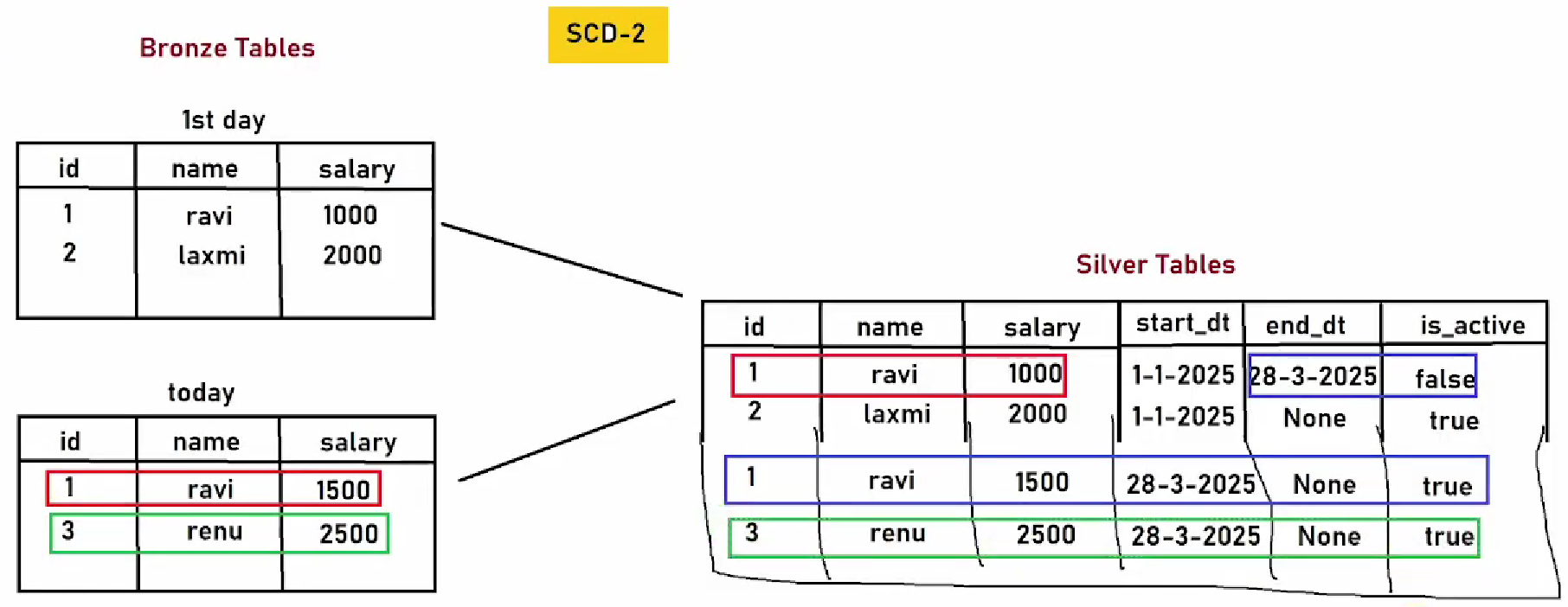
- First load the 1st day data to silver table we need to add some cols to silver table to maintain scd type 2.



After 1 month we got new records:



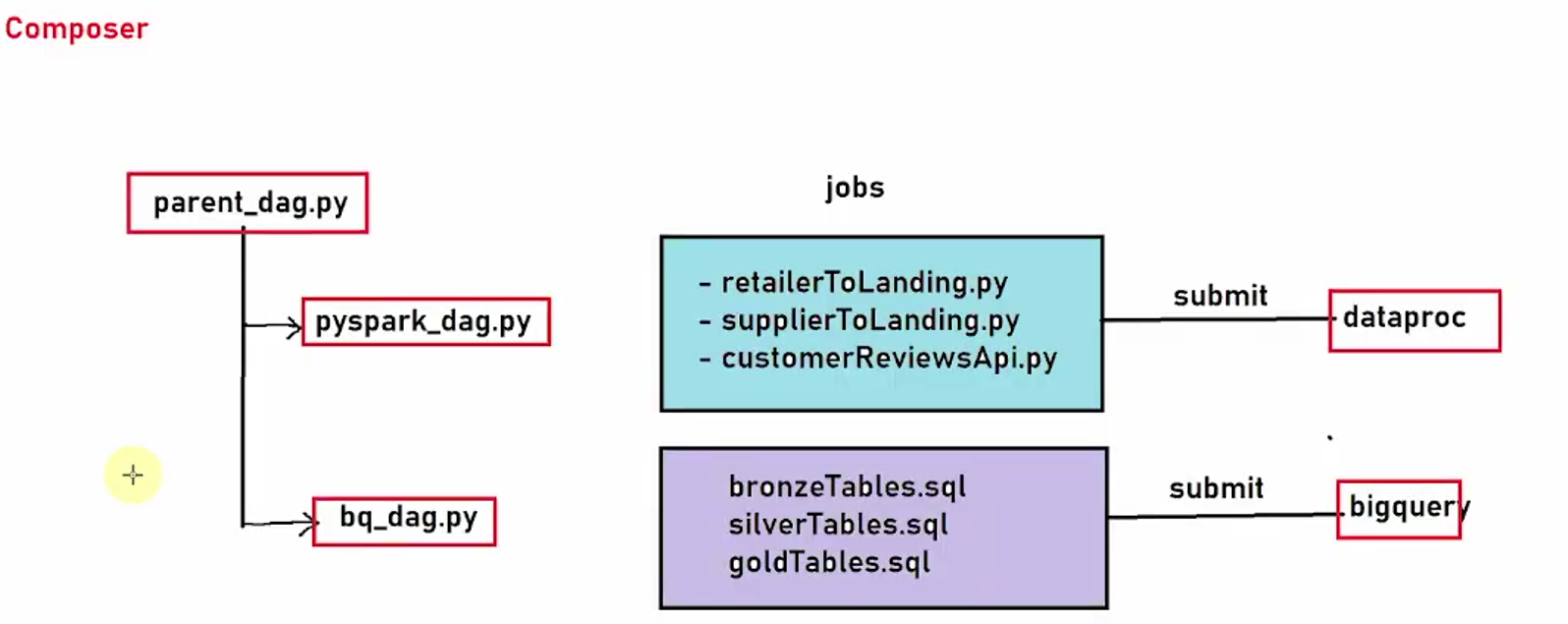
We need to update add new row for ravi and change the end\_dt and is\_active cols and for renu add new record as it is.



**C. Gold Layer:** Valuable insights we are finding in gold layer.

🡪 Create Workflow Orchestration:

It will take 20 mins to create a composer meanwhile we need to create DAG first. We have 3 DAGS i.e. pyspark\_dag, bigquery\_dag and parent dag to create dependency or manage those both dags.

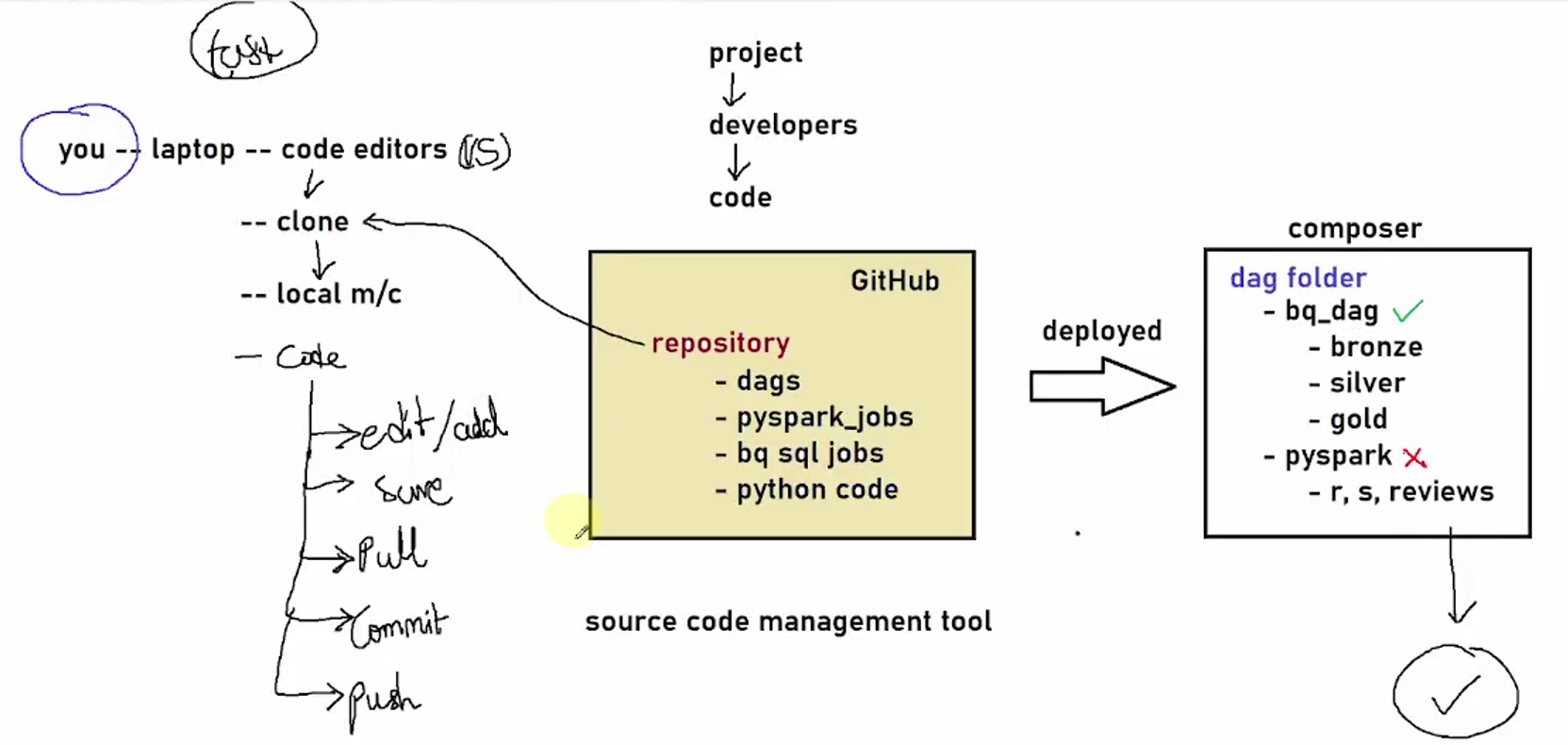


**5. CI/CD:**

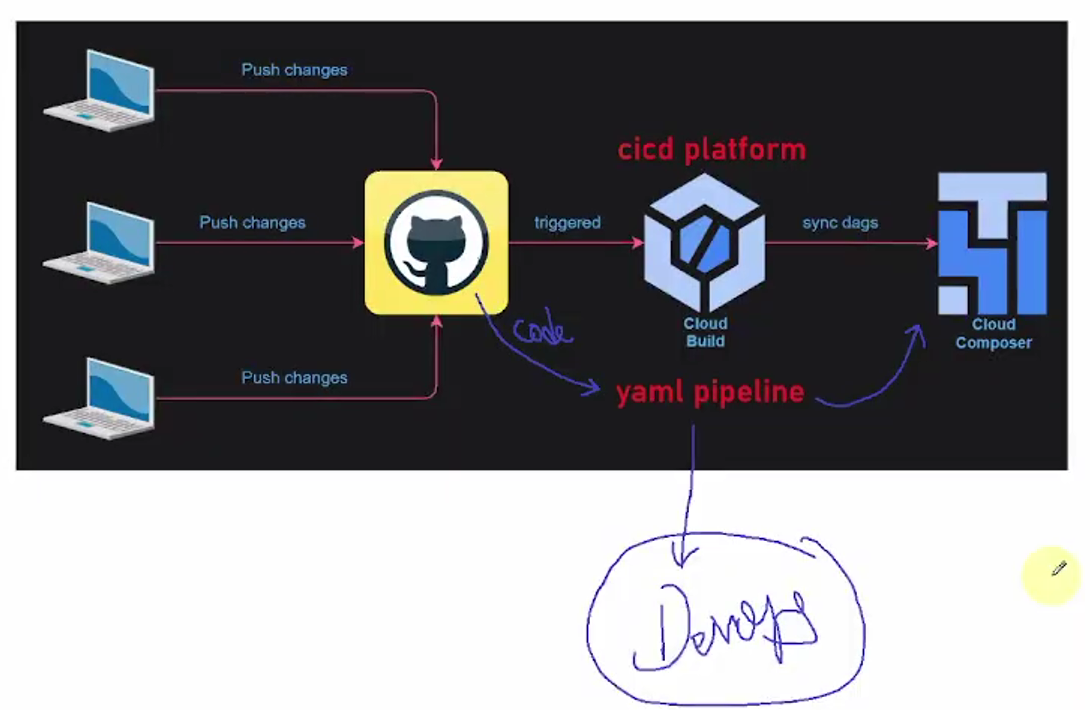
- Setup airflow

- Setup github

- Setup Cloud Build Trigger

****

Whenever you join a company the code is available on the github. The first task as a employee is to clone the code from github to local. The deployement is done using composer which is a workflow orchestration tool automatically. The dags are available in github and should be also present in composer to run the dags properly hence loaded into **dag folder** properly.

Cloud Build is a CI/CD Platform where yaml file is running and as per the changes in the code are integrated or updated into the cloud composer.