TASK 1 – The pipeline:

I have build a prod to use pipeline, it’s a little bit complex to run but is not only a test exercise pipeline, this is a pipeline ready to use in production, you can download from my git repository: <https://github.com/gilsonbellon/data-stack.git>, please clone that repository and run docker-compose up -d.

Please let me know when you have cloned the repository, than I will back to private once again, I don’t want anybody downloading my data stack.

Aside: I have running on Mac M1 and have no problem running it, it’s also should be run on Windows and Linux I recommend have at least 12 Gb mem available as we have a fill applications running as stack.

After docker up all the container, please follow the README.md file on this repo to be able to log in and add additional settings when applied, if for some reason you find that is quit complex and want me to demonstrate the stack please let me know I can be in the office with my computer and demonstrate this environment.

The stack consist of:

Airflow: a task coordinator.

Airbyte: data extraction.

Dbt: data transformation.

Minio: object storage.

Clickhouse: data storage.

Openmetadata: data catalog.

Metabase: data visualization.

Some screen shot has been taken to demonstration some of the tasks.

One of the task is to save the data in the object storage, such as AWS S3, I used Minio for that, but running the s3\_file\_uploader on Airflow this task has been done.

After running the s3\_file\_uploader file:

A screenshot of a computer

Description automatically generated

A screenshot of a phone

Description automatically generated

As you can see now originations json files and payments json files is on S3 object store:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Two sources has been set on Airbyte to Extract to our db warehouse:

One for the originations json files

And the other for payments json files

A screenshot of a computer

Description automatically generated

And the Clickhouse warehouse destination:

A screenshot of a phone

Description automatically generated

To extract data from the json files and insert into the dwh the connections has been created:

A screenshot of a computer

Description automatically generated

Payments data into Clickhouse:

A screenshot of a computer

Description automatically generated

Originations was decomposed into two tables,

The Originations one:

A screenshot of a computer

Description automatically generated

And Installments:

A screenshot of a computer

Description automatically generated

I could build and run another dag in Airflow to make the data transformation from json stream to db tables, but I prefer to do this using SQL to show a little bit of my skill on SQL as this is an important language for Data Engineers.

use dwh;

CREATE TABLE IF NOT EXISTS payment\_data

(

    paymentId String,

    installmentId String,

    paymentDate DateTime,

    paymentValue Decimal(10, 2)

) ENGINE = MergeTree()

ORDER BY (paymentId, installmentId);

INSERT INTO payment\_data

SELECT

JSONExtractString(\_airbyte\_data, 'paymentId') AS paymentId,

JSONExtractString(\_airbyte\_data, 'installmentId') AS installmentId,

toDateTime(JSONExtractString(\_airbyte\_data, 'paymentDate')) AS paymentDate,

toDecimal64(JSONExtractString(\_airbyte\_data, 'paymentValue'), 2) AS paymentValue

FROM dwh\_raw\_\_stream\_payments\_stream;

```

CREATE TABLE Originations (

    originationId String,

    clientId String,

    registerDate Date

) ENGINE = MergeTree()

ORDER BY originationId;

INSERT INTO Originations

SELECT

JSONExtractString(\_airbyte\_data, 'originationId') AS originationId,

    JSONExtractString(\_airbyte\_data, 'clientId') AS clientId,

    toDateTime(JSONExtractString(\_airbyte\_data, 'registerDate')) AS registerDate

FROM dwh\_raw\_\_stream\_originations\_stream;

CREATE TABLE Installments (

    originationId String,

    installmentId String,

    dueDate Date,

    installmentValue Decimal(10, 2)

) ENGINE = MergeTree()

ORDER BY (originationId, installmentId);

INSERT INTO Installments

SELECT

JSONExtractString(\_airbyte\_data, 'originationId') AS originationId,

   arrayJoin(arrayMap(x -> JSONExtractString(x, 'installmentId'), JSONExtractArrayRaw(\_airbyte\_data, 'installments'))) AS installmentId,

   arrayJoin(arrayMap(x -> toDateTime(JSONExtractString(x, 'dueDate')), JSONExtractArrayRaw(\_airbyte\_data, 'installments'))) AS dueDate,

   arrayJoin(arrayMap(x -> toDecimal64(JSONExtractString(x, 'installmentValue'), 2), JSONExtractArrayRaw(\_airbyte\_data, 'installments'))) AS installmentValue

FROM dwh\_raw\_\_stream\_originations\_stream;

You can play with all the stack tools for be able to see data quality transfer and even set some of the query on metabase to visualize it.

TASK 2 – Governance and Privacy.

Data governance and privacy are crucial aspects of modern information management.

Key aspects include:

1. Access control: Defining roles and permissions for data access
2. Data quality management: Ensuring data accuracy and consistency
3. Data lifecycle management: Handling data from creation to deletion
4. Compliance: Adhering to relevant regulations and standards

Regarding "who can access what":

1. Role-based access control (RBAC): Assign access rights based on job roles
2. Principle of least privilege: Grant users only the minimum necessary access
3. Data classification: Categorize data based on sensitivity and importance
4. Access monitoring and auditing: Track who accesses what data and when

Data Privacy: Data privacy focuses on the proper handling, processing, storage, and usage of personal information. It's about protecting individuals' data from unauthorized access and ensuring compliance with privacy laws. Key aspects include:

1. Data minimization: Collect only necessary data
2. Purpose limitation: Use data only for specified, legitimate purposes
3. Consent management: Obtain and manage user consent for data processing
4. Data subject rights: Allow individuals to access, correct, or delete their data

Regarding "how to store it in a private way":

1. Encryption: Use strong encryption for data at rest and in transit
2. Data masking: Obscure sensitive information in non-production environments
3. Secure data centers: Use facilities with physical and digital security measures
4. Access logging: Keep detailed logs of data access and modifications
5. Secure disposal: Properly destroy data when no longer needed

I have already implemented this on Openmetadata from the stack.

TASK 3 – SQL Coding.

I have run that query based on Clickhouse, one of the best Database for analytics at the moment.

Q 1:

SELECT  
 *toDate*(transaction\_date) AS date,  
 *sum*(*if*(Transaction\_types = 'Deposit', amount, 0)) AS daily\_total\_deposits,  
 *countIf*(Transaction\_types = 'Deposit') AS daily\_deposit\_count,  
 *uniqIf*(Playier\_ID, Transaction\_types = 'Deposit') AS daily\_unique\_depositors,  
 *if*(daily\_deposit\_count > 0, daily\_total\_deposits / daily\_deposit\_count, 0) AS daily\_average\_deposit  
FROM transactions  
WHERE *toYYYYMM*(transaction\_date) = 202404  
GROUP BY date  
ORDER BY date  
  
Q 2:

WITH daily\_stats AS (  
 SELECT   
 toDate(transaction\_date) AS date,  
 sum(if(Transaction\_types = 'Deposit', amount, 0)) AS daily\_total\_deposits,  
 uniqIf(playier\_id, Transaction\_types = 'Deposit') AS daily\_unique\_depositors  
 FROM your\_table  
 WHERE toYYYYMM(transaction\_date) = 202404  
 GROUP BY date  
)  
SELECT   
 date,  
 *sum*(daily\_total\_deposits) OVER (ORDER BY date) AS cumulative\_total\_deposits,  
 *uniqMerge*(*uniqState*(daily\_unique\_depositors)) OVER (ORDER BY date) AS cumulative\_unique\_depositors  
FROM daily\_stats  
ORDER BY date

Please if you find any issue that you want to discuss I am available to demonstrate personally for you in your office.

Best regards,

Gilson R Bellon.