Curso: Tecnologia em Data Science, Big Data, BI & Data Engineering

Projeto: CHALLENGE (012024)

Turmas do 1o ano: 1TSCPV

Sprint 3

DataStorm

Ana Beatriz Azevedo, RM557420

Heloiza Oliveira, RM558881

Isabelle Nahas, RM557405

Matheus Madrid, RM555799

Sara Sitta, RM555113

DISCIPLINA: ARCHITECTURE ANALYTICS & NOSQL

Professores: Leandro Romualdo da Silva, Milton Goya

São Paulo, Setembro de 2024

Documentação do Pipeline de Dados

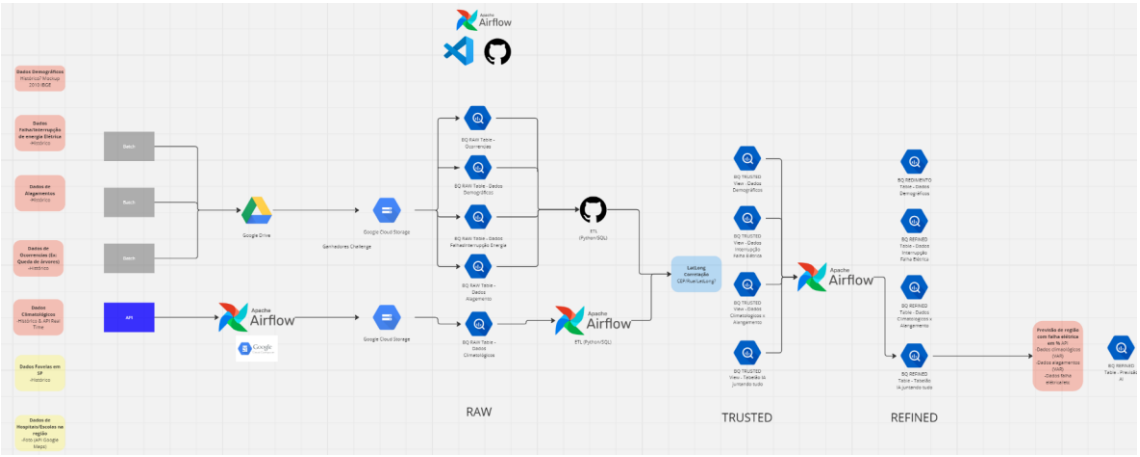
Visão Geral

Este documento descreve o pipeline de dados implementado para ingestão, transformação, armazenamento e consumo de dados climáticos coletados de uma API, usando Apache Airflow no Google Cloud Composer, Google Cloud Storage como Data Lake, e BigQuery para armazenamento e transformação dos dados. O pipeline é composto por três DAGs principais, cada uma responsável por uma etapa específica do fluxo de dados.

1. Ferramenta de ETL/Ingestão

Para atender aos requisitos de coleta, ingestão, armazenamento, transformação e orquestração, foram utilizados Apache Airflow no Google Cloud Composer para orquestração, Google Cloud Storage para armazenamento temporário, e BigQuery para transformação e armazenamento dos dados estruturados.

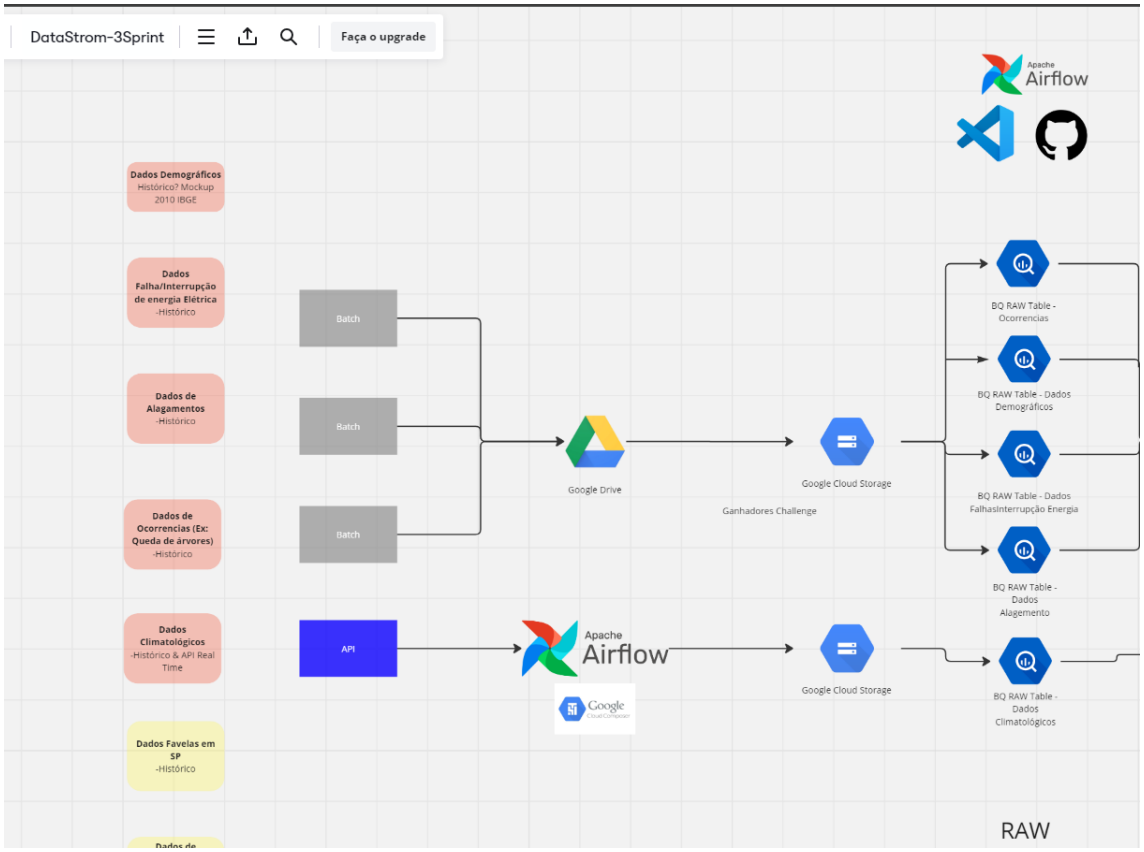
Arquitetura atualizada:



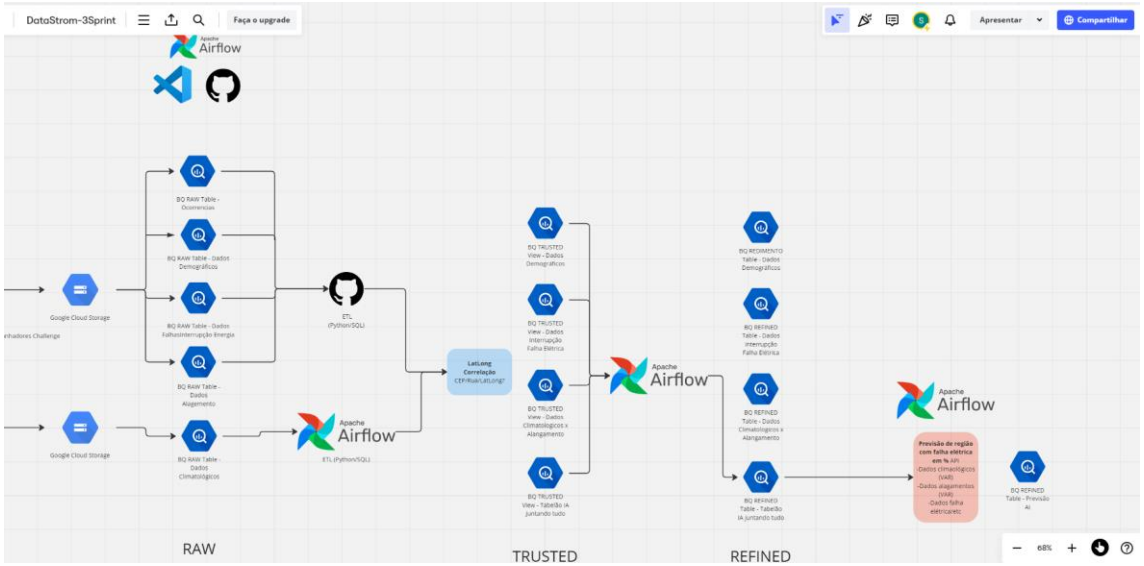
Link: https://miro.com/app/board/uXjVKmDD4gU=/?share_link_id=695605839813

https://miro.com/welcomeonboard/dnRiaHI0VEJPOW9KdW84ZUNCanJBTTF4dkpXTkQ2M3ZRNkFOend0cmxsRWNtZGlKNm8yYUtlWDBHeXF1Qk5CVHwzNDU4NzY0NTc3MzU0NTcyMjQ0fDI=?share_link_id=512033522479

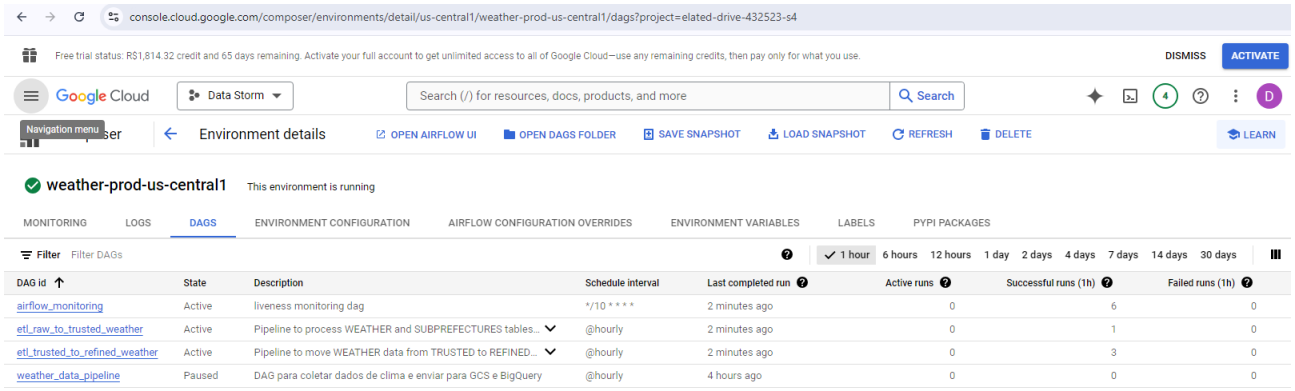
Parte1(ArquiteturaAtt)



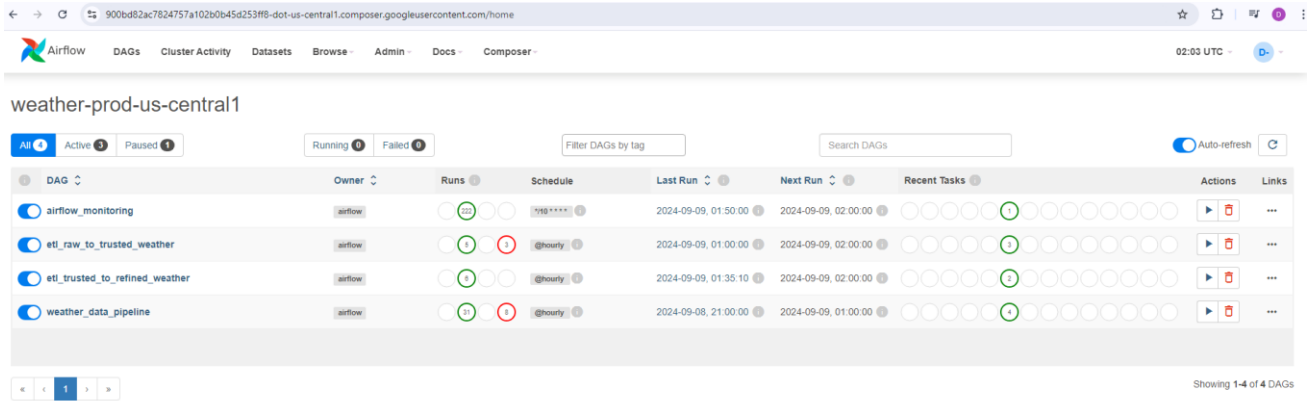
Parte2(ArqAtt)



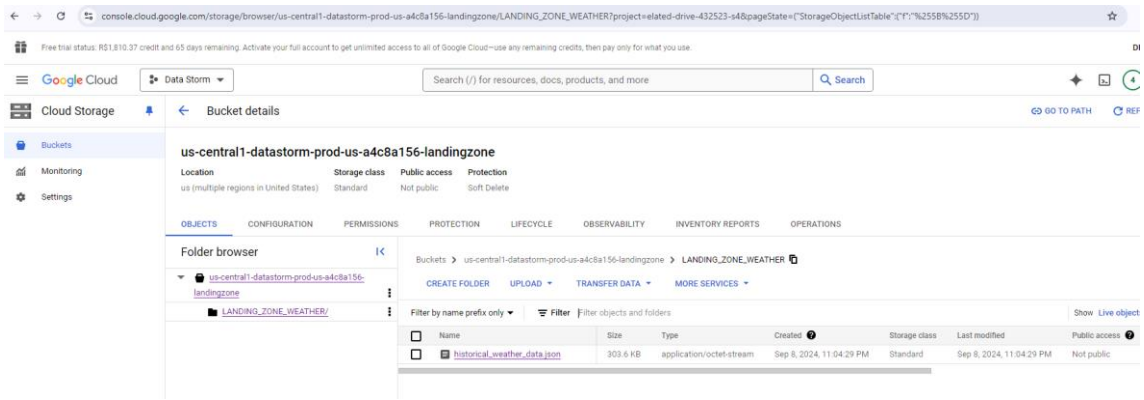
Google Cloud Composer



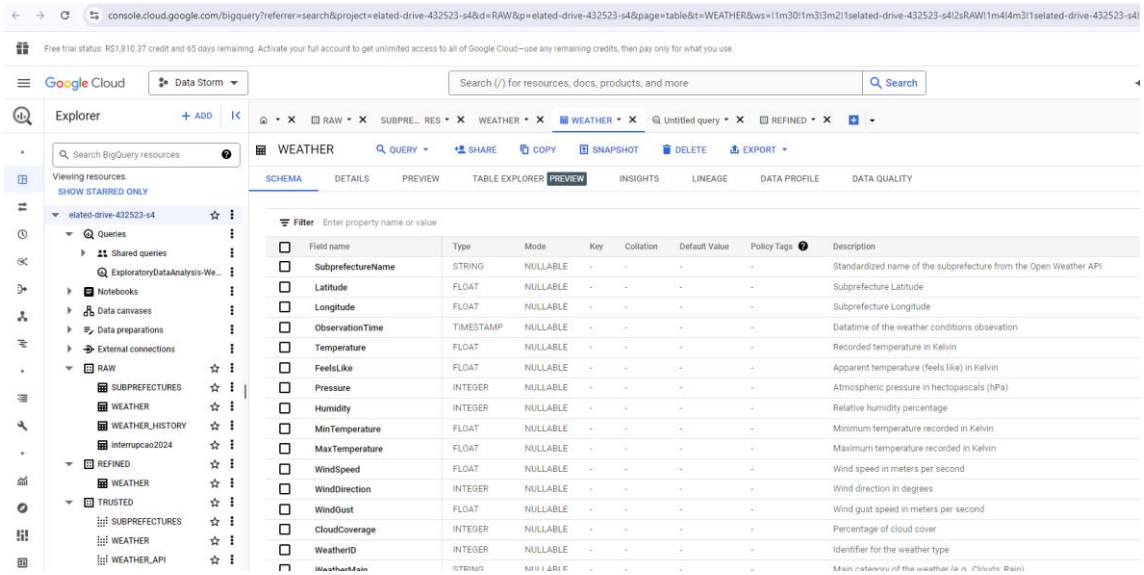
Apache Airflow:



CloudStorage – Landing Zone



Bigquery GCP



1.1 Coleta e Ingestão de Dados para o Data Lake e BigQuery (DAG: weather_data_pipeline_dag)

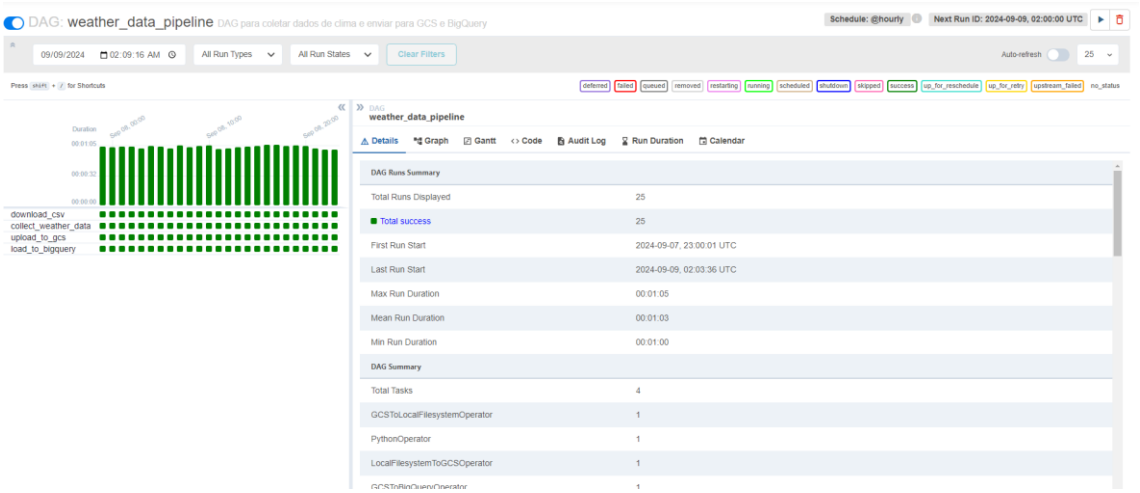
- **Nome da DAG:** weather_data_pipeline_dag
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; Google Cloud Storage para armazenamento dos dados e BigQuery para estruturação do Lake
- **Processo:**
    - **Coleta de Dados:** A DAG coleta dados climáticos da API do OpenWeatherMap em intervalos regulares (a cada 1 hora).
    - **Armazenamento Temporário:** Os dados coletados são salvos no Google Cloud Storage, que atua como o Data Lake.
    - **Carga para BigQuery:** Após salvar no Cloud Storage, os dados são carregados na camada RAW do BigQuery, utilizando operadores específicos do Airflow para transferência de arquivos do Cloud Storage para o BigQuery.

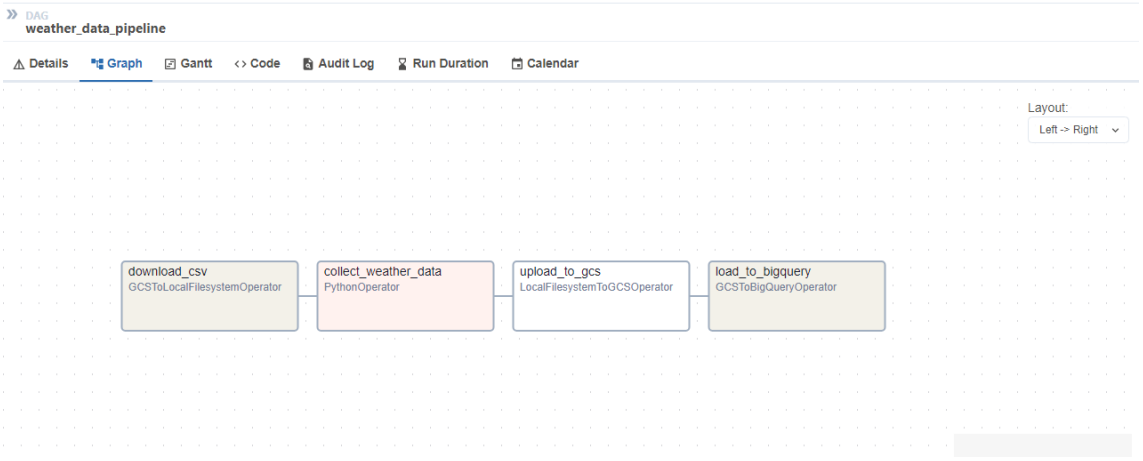
weather_data_pipeline_dag.py > ...

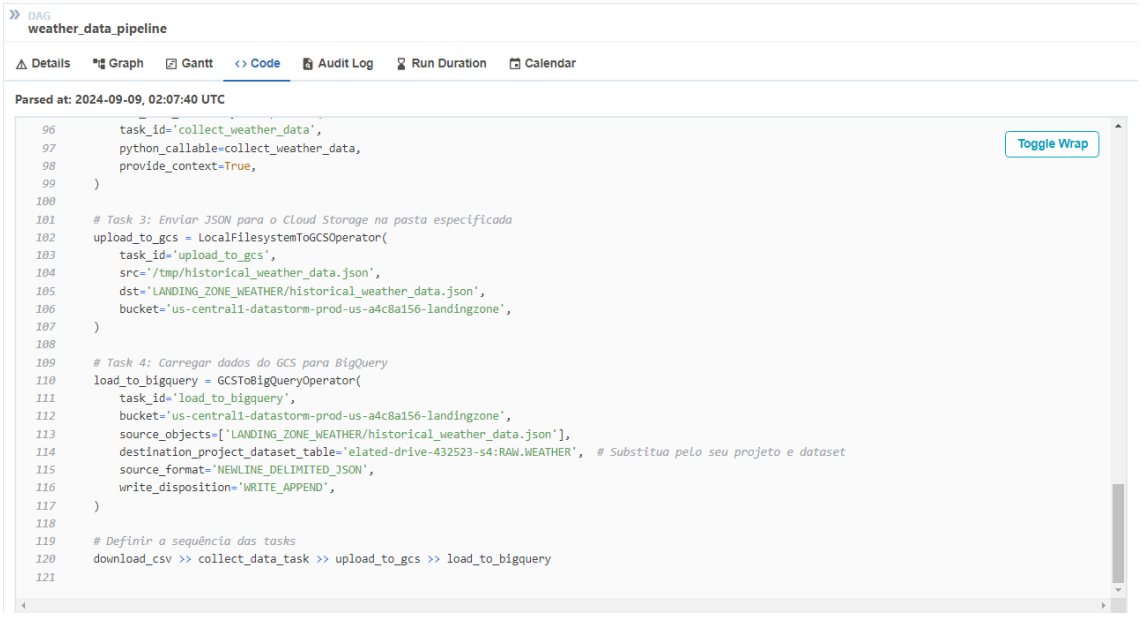
```python
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.google.cloud.transfers.local_to_gcs import LocalFilesystemToGCSOperator
from airflow.providers.google.cloud.transfers.gcs_to_bigquery import GCSToBigQueryOperator
from airflow.providers.google.cloud.transfers.gcs_to_local import GCSToLocalFilesystemOperator
from airflow.utils.dates import days_ago
import pandas as pd
import requests
import time
import json
from datetime import datetime

# Função para coletar dados das subprefeituras e salvar em formato adequado para o BigQuery
def collect_weather_data(**kwargs):
    # Baixar o CSV das subprefeituras diretamente do Google Cloud Storage
    subprefeituras = pd.read_csv('/tmp/subprefeituras-sp.csv', sep=';', header=0)
    subprefeituras.columns = subprefeituras.columns.str.strip().str.lower()

    API_KEY = '--'  # Substitua pela sua chave da API
    BASE_URL = 'https://history.openweathermap.org/data/2.5/history/city'

    results = []
    for index, row in subprefeituras.iterrows():
        lat = round(row['latitude'], 2)
        lon = round(row['longitude'], 2)
        subprefeitura_nome = row['subprefeitura']
        url = f"{BASE_URL}?lat={lat:.2f}&lon={lon:.2f}&appid={API_KEY}"

        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            if 'list' in data:
                for entry in data['list']:
                    result = {
                        'subprefeitura': subprefeitura_nome,
                        'latitude': lat,
                        'longitude': lon,
                        'timestamp': datetime.utcfromtimestamp(entry['dt']).isoformat(),  # Converter para timestamp ISO
                        'temperature': entry['main'].get('temp'),
                        'feels_like': entry['main'].get('feels_like'),
                        'pressure': entry['main'].get('pressure'),
                        'humidity': entry['main'].get('humidity'),
                        'temp_min': entry['main'].get('temp_min'),
                        'temp_max': entry['main'].get('temp_max'),
                        'wind_speed': entry['wind'].get('speed'),
                        'wind_deg': entry['wind'].get('deg')
```

Arquivo anexado - weather_data_pipeline_dag.py

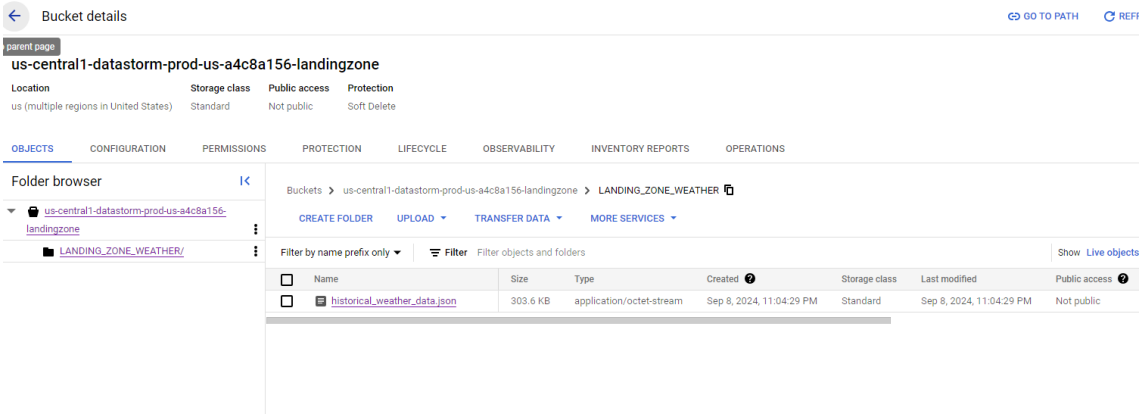
weather_data_pipeline_dag. DAG no Airflow



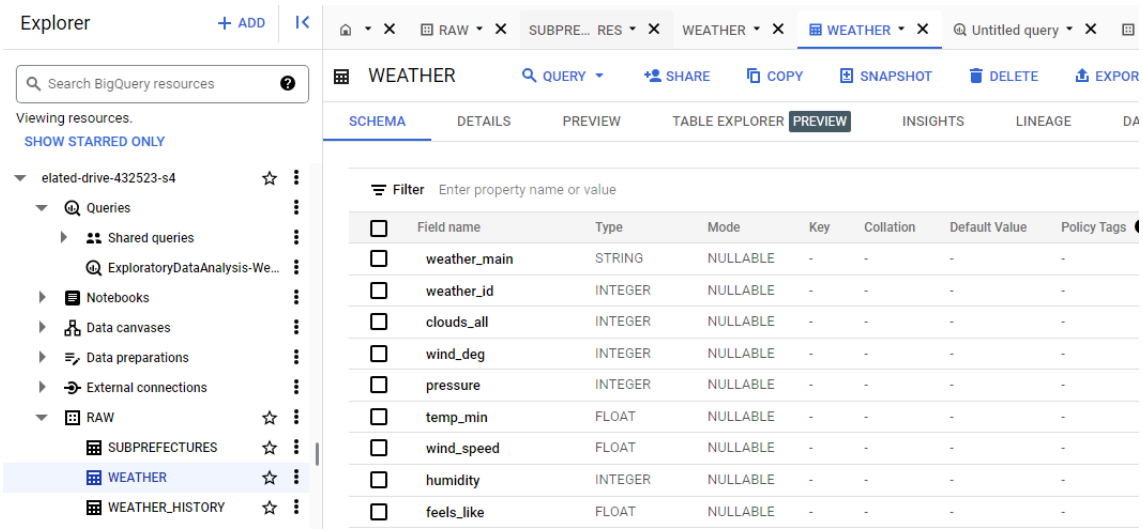




LANDING ZONE no Google Cloud Storage



Camada RAW.WEATHER



1.2 Estruturação e Preparação dos Dados para a Camada Estruturada (DAG: etl_raw_to_trusted_weather)

- **Nome da DAG:** etl_raw_to_trusted_weather
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; Google Cloud Storage para armazenamento dos dados e BigQuery para transformação dos dados.
- **Processo:**
    - Esta DAG transforma e estrutura os dados movendo-os da camada RAW para a camada TRUSTED no BigQuery.
    - **Transformações Incluídas:**
        - Ajuste de precisão de latitude e longitude para garantir a consistência entre os conjuntos de dados.
        - Conversão de CEP de string para inteiro, removendo caracteres indesejados.
        - Criação de views para padronizar os nomes das colunas para o padrão de Data Lake (letra maiúscula, sem underscores).

Task 4 da DAG weather_data_pipeline_dag

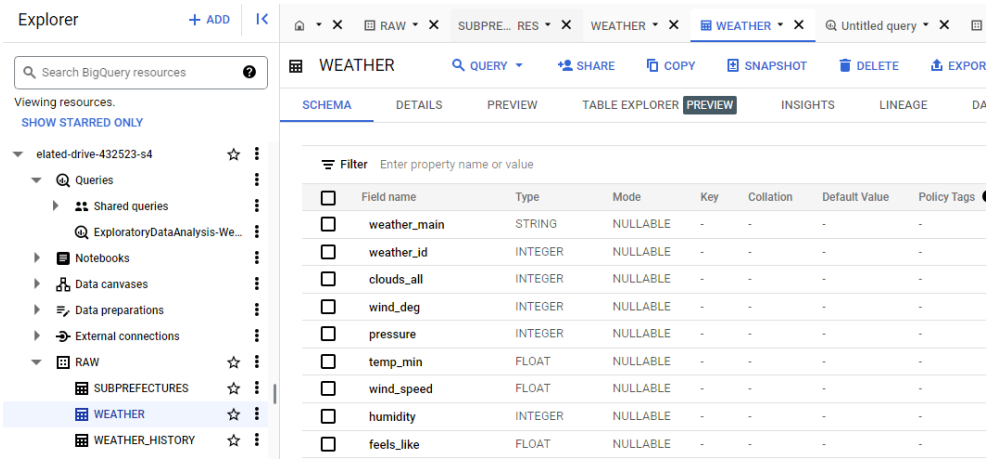
DAG
weather_data_pipeline

Details | Graph | Gantt | Code | Audit Log | Run Duration | Calendar

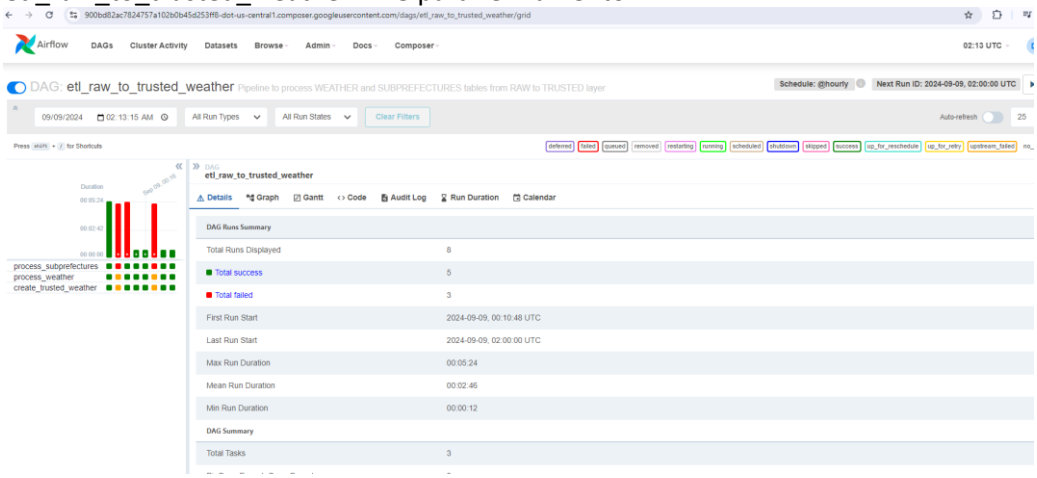
Parsed at: 2024-09-09, 02:07:40 UTC

```
        task_id='collect_weather_data',
        python_callable=collect_weather_data,
        provide_context=True,
    )

    # Task 3: Enviar JSON para o Cloud Storage na pasta especificada
    upload_to_gcs = LocalFilesystemToGCSOperator(
        task_id='upload_to_gcs',
        src='/tmp/historical_weather_data.json',
        dst='LANDING_ZONE_WEATHER/historical_weather_data.json',
        bucket='us-central1-datastorm-prod-us-a4c8a156-landingzone',
    )

    # Task 4: Carregar dados do GCS para BigQuery
    load_to_bigquery = GCSToBigQueryOperator(
        task_id='load_to_bigquery',
        bucket='us-central1-datastorm-prod-us-a4c8a156-landingzone',
        source_objects=['LANDING_ZONE_WEATHER/historical_weather_data.json'],
        destination_project_dataset_table='elated-drive-432523-s4:RAW.WEATHER',  # Substitua pelo seu projeto e dataset
        source_format='NEWLINE_DELIMITED_JSON',
        write_disposition='WRITE_APPEND',
    )

    # Definir a sequência das tasks
    download_csv >> collect_data_task >> upload_to_gcs >> load_to_bigquery
```

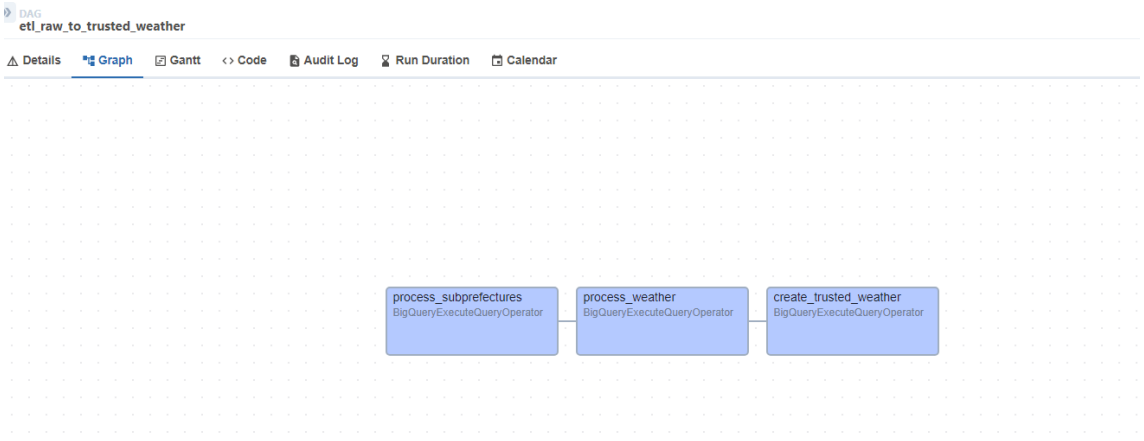
Camada RAW.WEATHER

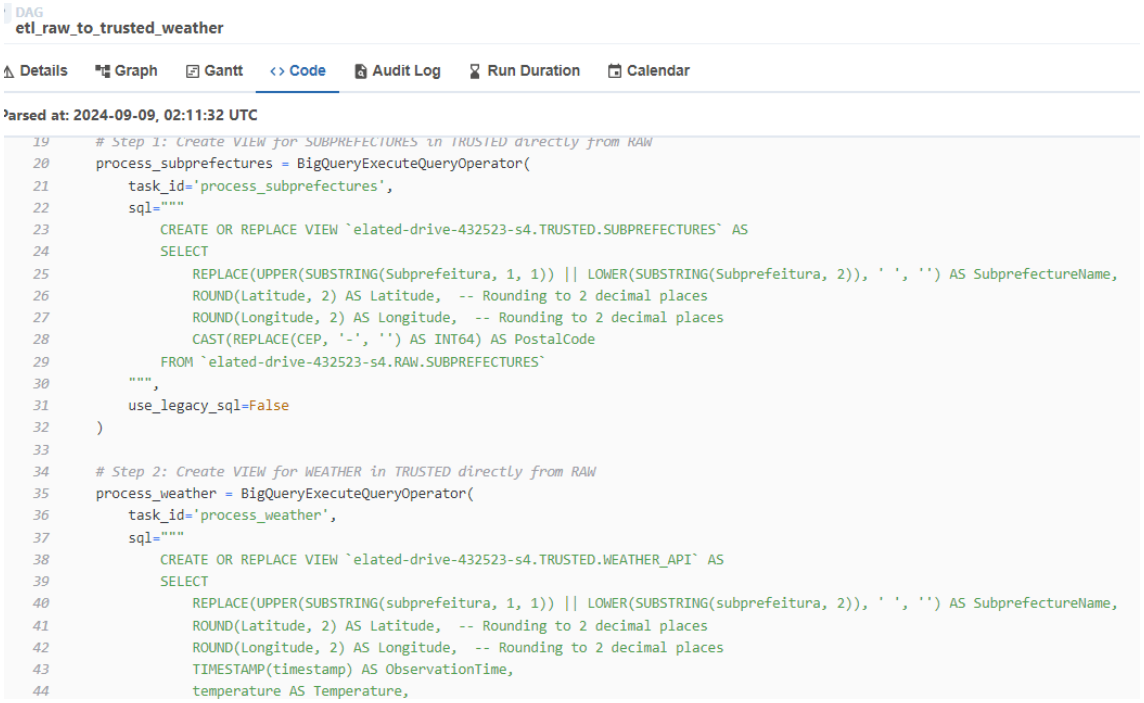


etl_raw_to_trusted_weather DAG para refinamento



Tasks da DAG

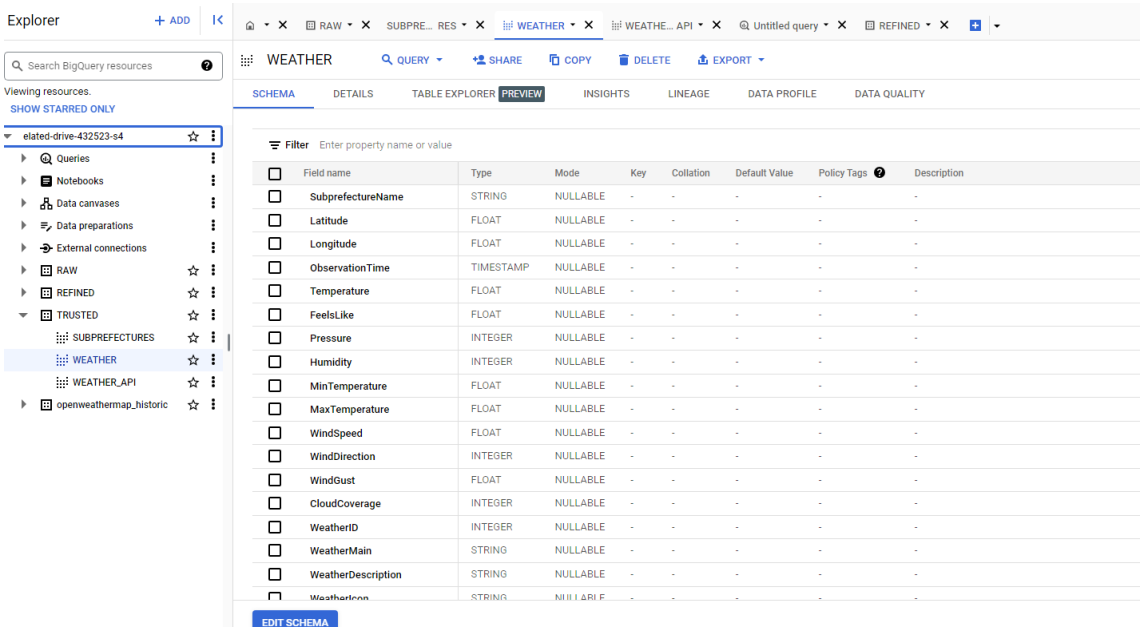





```
    # Step 1: Create VIEW for SUBPREFECTURES in TRUSTED directly from RAW
    process_subprefectures = BigQueryExecuteQueryOperator(
        task_id='process_subprefectures',
        sql="""
            CREATE OR REPLACE VIEW `elated-drive-432523-s4.TRUSTED.SUBPREFECTURES` AS
            SELECT
                REPLACE(UPPER(SUBSTRING(Subprefeitura, 1, 1)) || LOWER(SUBSTRING(Subprefeitura, 2)), ' ', '') AS SubprefectureName,
                ROUND(Latitude, 2) AS Latitude,  -- Rounding to 2 decimal places
                ROUND(Longitude, 2) AS Longitude,  -- Rounding to 2 decimal places
                CAST(REPLACE(CEP, '-', '') AS INT64) AS PostalCode
            FROM `elated-drive-432523-s4.RAW.SUBPREFECTURES`
        """,
        use_legacy_sql=False
    )

    # Step 2: Create VIEW for WEATHER in TRUSTED directly from RAW
    process_weather = BigQueryExecuteQueryOperator(
        task_id='process_weather',
        sql="""
            CREATE OR REPLACE VIEW `elated-drive-432523-s4.TRUSTED.WEATHER_API` AS
            SELECT
                REPLACE(UPPER(SUBSTRING(subprefeitura, 1, 1)) || LOWER(SUBSTRING(subprefeitura, 2)), ' ', '') AS SubprefectureName,
                ROUND(Latitude, 2) AS Latitude,  -- Rounding to 2 decimal places
                ROUND(Longitude, 2) AS Longitude,  -- Rounding to 2 decimal places
                TIMESTAMP(timestamp) AS ObservationTime,
                temperature AS Temperature,
```

Dag anexada no zip etl_raw_to_trusted_weather.py

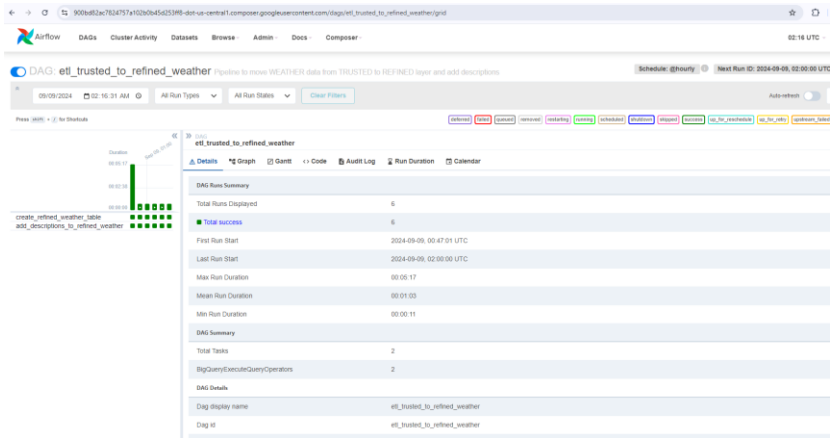
CAMADA TRUSTED

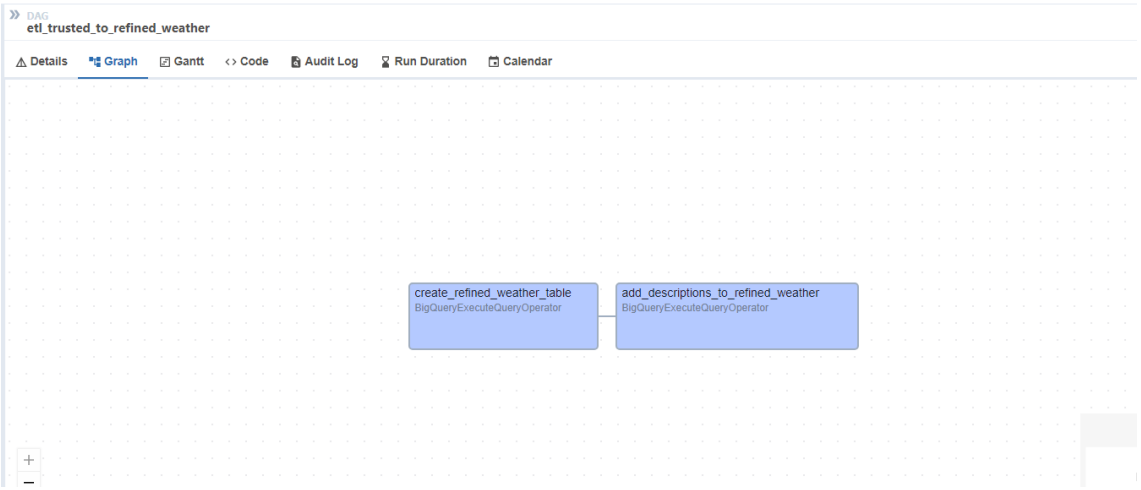


1.3 Refinamento e Enriquecimento dos Dados (DAG: etl_trusted_to_refined_weather)

- **Nome da DAG:** etl_trusted_to_refined_weather
- **Ferramenta Utilizada:** Apache Airflow com Google Cloud Composer para orquestração; BigQuery para refinamento e enriquecimento dos dados.
- **Processo:**
  - A última DAG move os dados da camada TRUSTED para a camada REFINED no BigQuery.
  - **Refinamento Inclui:**
    - Criação de tabelas finais na camada REFINED.
    - Adição de descrições das colunas para melhorar a documentação e a compreensão dos dados.
    - Junção de dados climáticos com informações de subprefeituras para fornecer um conjunto de dados enriquecido.

DAG etl_trusted_to_refined_weather no Airflow





DAG
etl_trusted_to_refined_weather

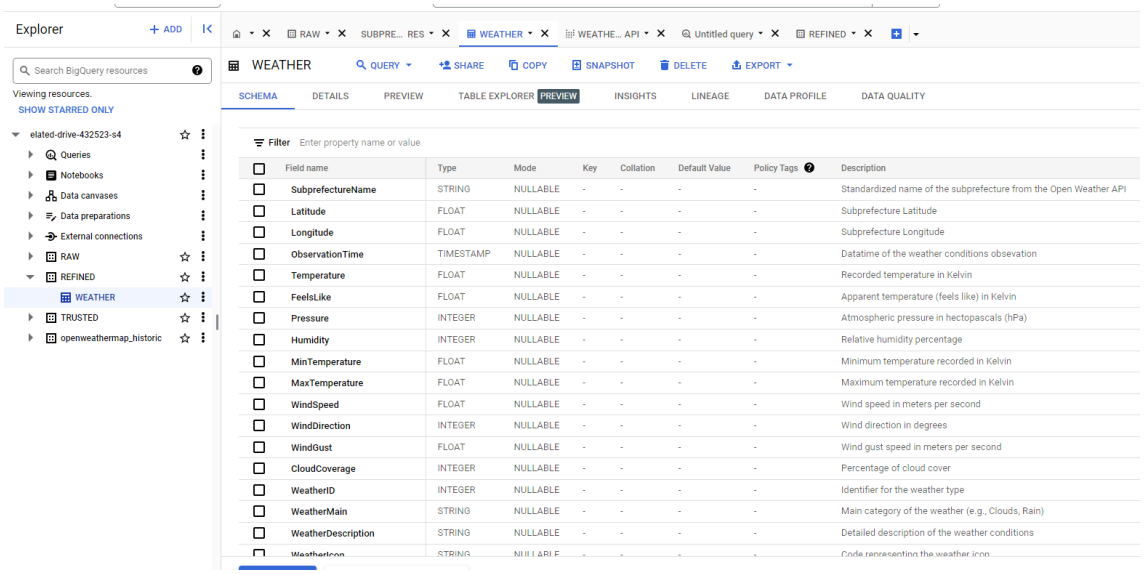
Details | Graph | Gantt | Code | Audit Log | Run Duration | Calendar

Parsed at: 2024-09-09, 02:15:25 UTC

```
    # Step 2: Consolidated ALTER TABLE to Add Descriptions to REFINED.WEATHER Table
    add_descriptions_to_refined_weather = BigQueryExecuteQueryOperator(
        task_id='add_descriptions_to_refined_weather',
        sql="""
            ALTER TABLE `elated-drive-432523-s4.REFINED.WEATHER`
            SET OPTIONS (
              description = 'Refined table with enriched weather and subprefecture data'
            );

            ALTER TABLE `elated-drive-432523-s4.REFINED.WEATHER`
            ALTER COLUMN SubprefectureName SET OPTIONS (description = 'Standardized name of the subprefecture from the Open Weather API'),
            ALTER COLUMN Latitude SET OPTIONS (description = 'Subprefecture Latitude'),
            ALTER COLUMN Longitude SET OPTIONS (description = 'Subprefecture Longitude'),
            ALTER COLUMN ObservationTime SET OPTIONS (description = 'Datatime of the weather conditions obsevation'),
            ALTER COLUMN Temperature SET OPTIONS (description = 'Recorded temperature in Kelvin'),
            ALTER COLUMN FeelsLike SET OPTIONS (description = 'Apparent temperature (feels like) in Kelvin'),
            ALTER COLUMN Pressure SET OPTIONS (description = 'Atmospheric pressure in hectopascals (hPa)'),
            ALTER COLUMN Humidity SET OPTIONS (description = 'Relative humidity percentage'),
            ALTER COLUMN MinTemperature SET OPTIONS (description = 'Minimum temperature recorded in Kelvin'),
            ALTER COLUMN MaxTemperature SET OPTIONS (description = 'Maximum temperature recorded in Kelvin'),
            ALTER COLUMN WindSpeed SET OPTIONS (description = 'Wind speed in meters per second'),
            ALTER COLUMN WindDirection SET OPTIONS (description = 'Wind direction in degrees'),
            ALTER COLUMN WindGust SET OPTIONS (description = 'Wind gust speed in meters per second'),
            ALTER COLUMN CloudCoverage SET OPTIONS (description = 'Percentage of cloud cover'),
            ALTER COLUMN WeatherID SET OPTIONS (description = 'Identifier for the weather type'),
```

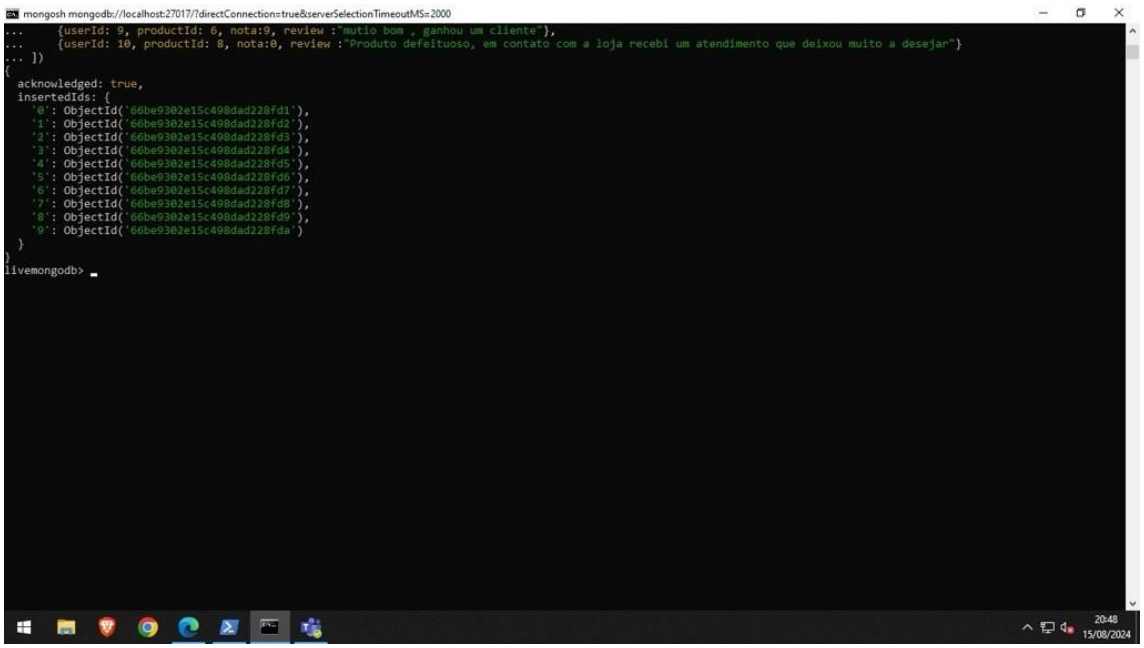
Arquivo anexado no zip etl_trusted_to_refined_weather.py

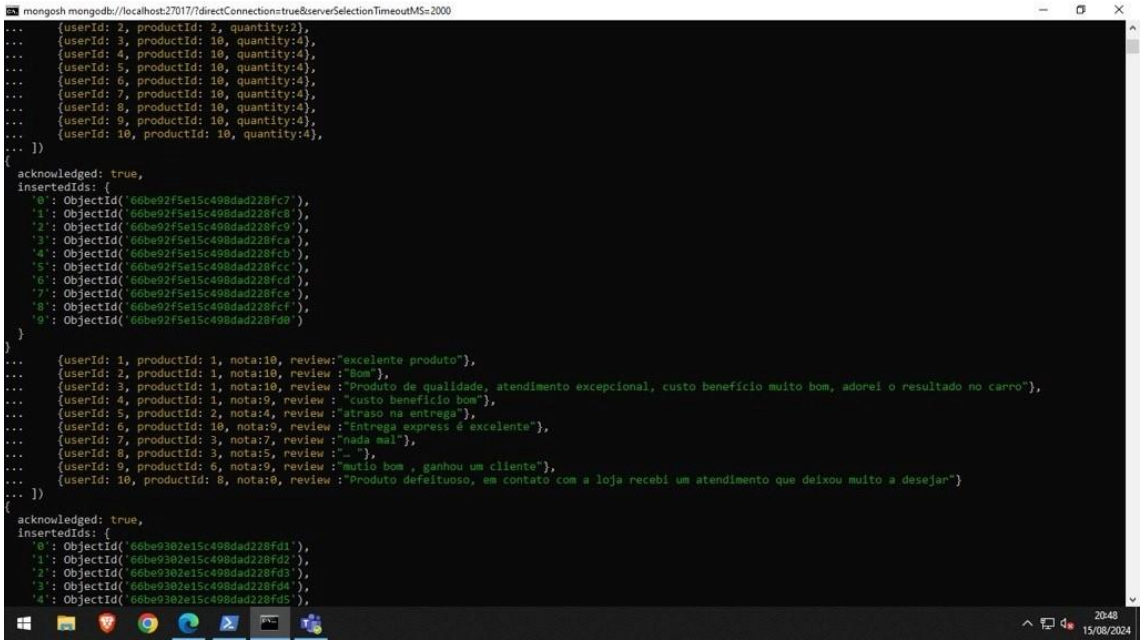
CAMADA REFINED com a tabela WEATHER final com as Descrições

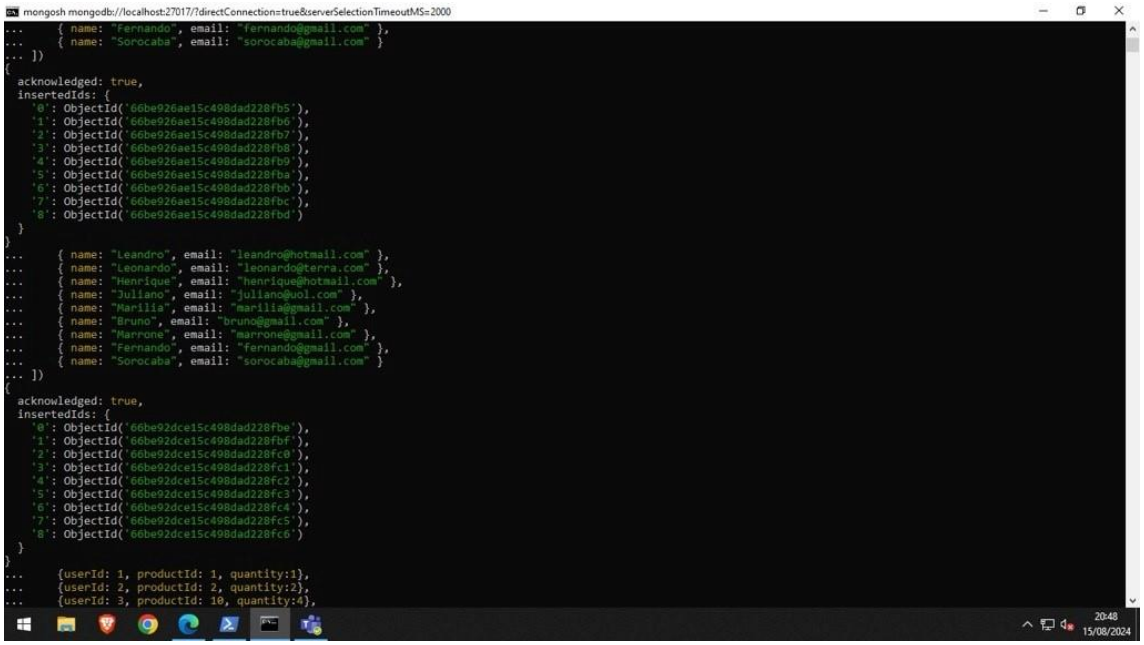


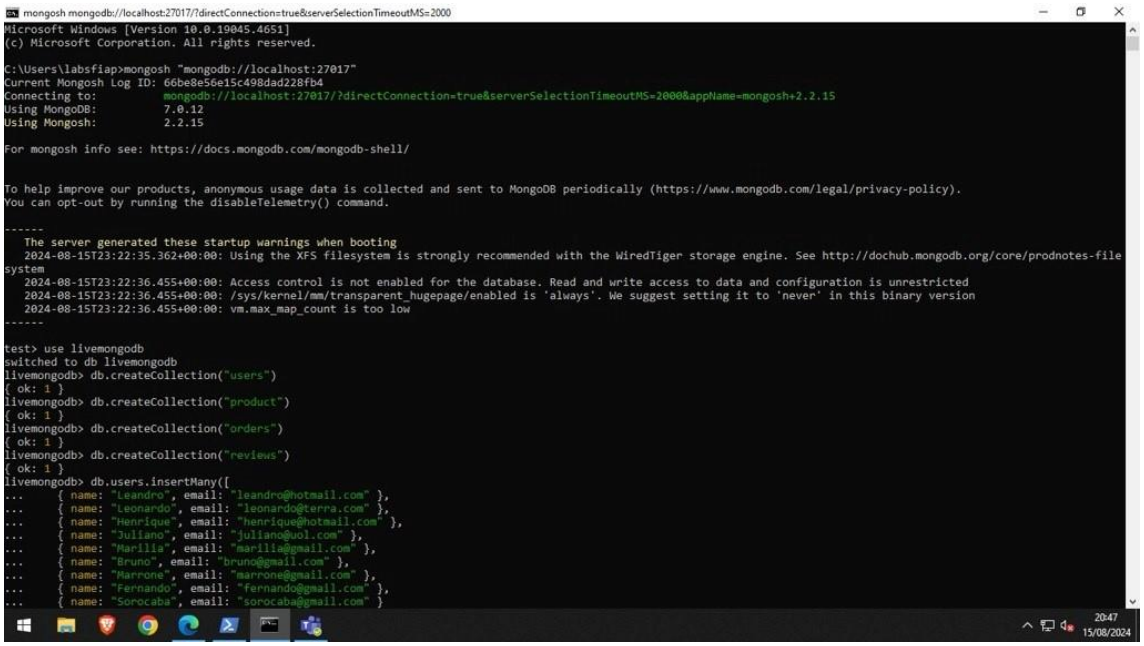
2. Implantação de um banco de dados NoSQL. O grupo deve escolher um banco de dados NoSQL e adicionar na sua solução em qualquer fase do pipeline de dados. Pesquise, questione o professor da disciplina e veja o que mais se adequa ao seu cenário e faça a implantação.

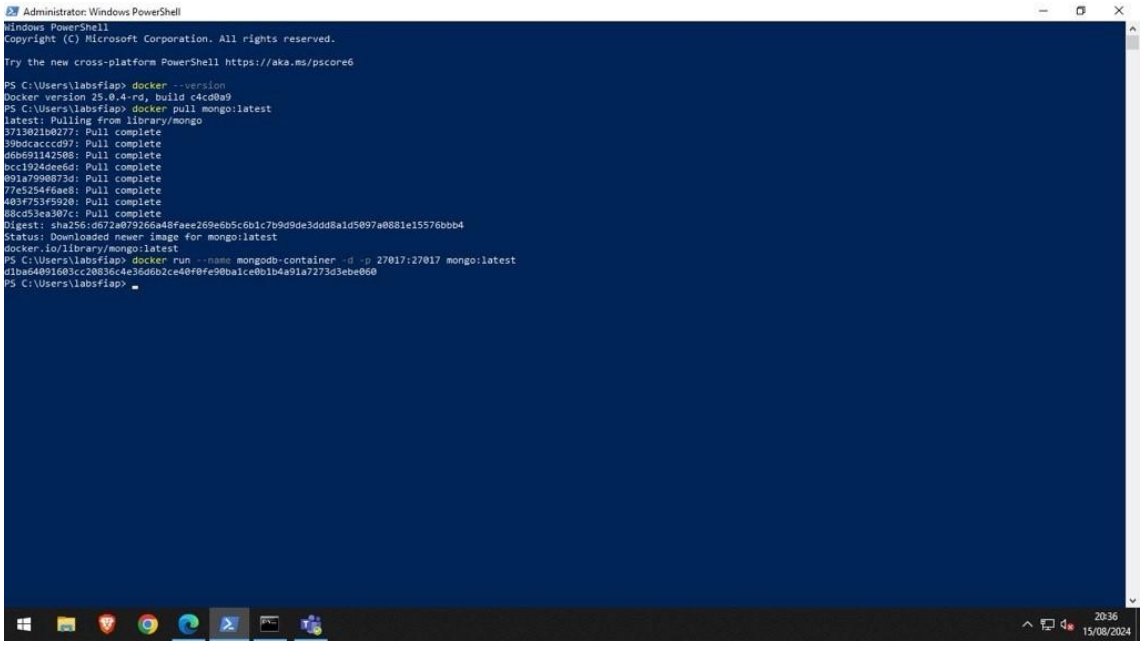
Todas as imagens anexadas no zip




mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
... {userId: 2, productId: 2, quantity:2},
... {userId: 3, productId: 10, quantity:4},
... {userId: 4, productId: 10, quantity:4},
... {userId: 5, productId: 10, quantity:4},
... {userId: 6, productId: 10, quantity:4},
... {userId: 7, productId: 10, quantity:4},
... {userId: 8, productId: 10, quantity:4},
... {userId: 9, productId: 10, quantity:4},
... {userId: 10, productId: 10, quantity:4},
...])
{
acknowledged: true,
insertedIds: {
'0': ObjectId('66be92f5e15c498dad228fc7'),
'1': ObjectId('66be92f5e15c498dad228fc8'),
'2': ObjectId('66be92f5e15c498dad228fc9'),
'3': ObjectId('66be92f5e15c498dad228fca'),
'4': ObjectId('66be92f5e15c498dad228fcb'),
'5': ObjectId('66be92f5e15c498dad228fcc'),
'6': ObjectId('66be92f5e15c498dad228fcd'),
'7': ObjectId('66be92f5e15c498dad228fce'),
'8': ObjectId('66be92f5e15c498dad228fcf'),
'9': ObjectId('66be92f5e15c498dad228fd0')
}
}
... {userId: 1, productId: 1, nota:10, review:"excelente produto"},
... {userId: 2, productId: 1, nota:10, review :"Bom"},
... {userId: 3, productId: 1, nota:10, review :"Produto de qualidade, atendimento excepcional, custo benefício muito bom, adorei o resultado no carro"},
... {userId: 4, productId: 1, nota:9, review : "custo benefício bom"},
... {userId: 5, productId: 2, nota:4, review :"atraso na entrega"},
... {userId: 6, productId: 10, nota:9, review :"Entrega express é excelente"},
... {userId: 7, productId: 3, nota:7, review :"nada mal"},
... {userId: 8, productId: 3, nota:5, review :"… "},
... {userId: 9, productId: 6, nota:9, review :"mutio bom , ganhou um cliente"},
... {userId: 10, productId: 8, nota:0, review :"Produto defeituoso, em contato com a loja recebi um atendimento que deixou muito a desejar"}
...])
{
acknowledged: true,
insertedIds: {
'0': ObjectId('66be9302e15c498dad228fd1'),
'1': ObjectId('66be9302e15c498dad228fd2'),
'2': ObjectId('66be9302e15c498dad228fd3'),
'3': ObjectId('66be9302e15c498dad228fd4'),
'4': ObjectId('66be9302e15c498dad228fd5'),
20:48
15/08/2024


mongosh mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000
... { name: "Fernando", email: "fernando@gmail.com" },
... { name: "Sorocaba", email: "sorocaba@gmail.com" }
...])
{
acknowledged: true,
insertedIds: {
'0': ObjectId('66be926ae15c498dad228fb5'),
'1': ObjectId('66be926ae15c498dad228fb6'),
'2': ObjectId('66be926ae15c498dad228fb7'),
'3': ObjectId('66be926ae15c498dad228fb8'),
'4': ObjectId('66be926ae15c498dad228fb9'),
'5': ObjectId('66be926ae15c498dad228fba'),
'6': ObjectId('66be926ae15c498dad228fbb'),
'7': ObjectId('66be926ae15c498dad228fbc'),
'8': ObjectId('66be926ae15c498dad228fbd')
}
}
... { name: "Leandro", email: "leandro@hotmail.com" },
... { name: "Leonardo", email: "leonardo@terra.com" },
... { name: "Henrique", email: "henrique@hotmail.com" },
... { name: "Juliano", email: "juliano@uol.com" },
... { name: "Marilia", email: "marilia@gmail.com" },
... { name: "Bruno", email: "bruno@gmail.com" },
... { name: "Marrone", email: "marrone@gmail.com" },
... { name: "Fernando", email: "fernando@gmail.com" },
... { name: "Sorocaba", email: "sorocaba@gmail.com" }
...])
{
acknowledged: true,
insertedIds: {
'0': ObjectId('66be92dce15c498dad228fbe'),
'1': ObjectId('66be92dce15c498dad228fbf'),
'2': ObjectId('66be92dce15c498dad228fc0'),
'3': ObjectId('66be92dce15c498dad228fc1'),
'4': ObjectId('66be92dce15c498dad228fc2'),
'5': ObjectId('66be92dce15c498dad228fc3'),
'6': ObjectId('66be92dce15c498dad228fc4'),
'7': ObjectId('66be92dce15c498dad228fc5'),
'8': ObjectId('66be92dce15c498dad228fc6')
}
}
... {userId: 1, productId: 1, quantity:1},
... {userId: 2, productId: 2, quantity:2},
... {userId: 3, productId: 10, quantity:4},
20:48
15/08/2024





3. Documentação e Evidências

Todas as etapas foram capturadas e documentadas para comprovar o funcionamento do pipeline.

Conclusão

O pipeline foi implementado utilizando Google Cloud Composer com Apache Airflow para orquestrar a coleta, transformação e carga de dados climáticos. As três DAGs (weather_data_pipeline_dag, etl_raw_to_trusted_weather, e

etl_trusted_to_refined_weather) garantem que os dados sejam coletados, transformados e enriquecidos de forma escalável e automatizada. As descrições dos campos na camada REFINED ajudam a documentar e entender o conjunto de dados final para consumo analítico.

Este documento serve como uma referência completa das ferramentas e processos utilizados, garantindo uma visão clara do fluxo de dados desde a coleta até o refinamento final para uso analítico.