

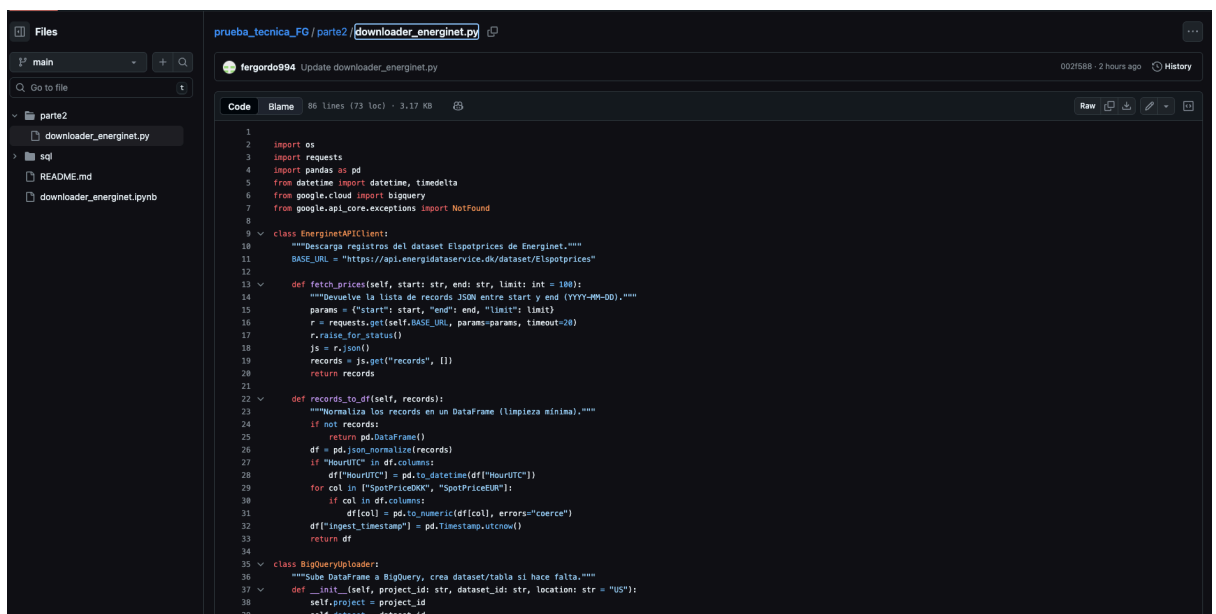
# Prueba Técnica: Mini-Proyecto de Pipeline de Datos

## Primera Parte: GitHub

- 1) Tienes que crear una cuenta de GitHub si no la tienes. Es gratuito, así que no debería de haber problema.
- 2) Crea un repositorio nuevo. Tiene que ser público para que lo podamos revisar (nos tendrás que pasar el enlace al repo).
- 3) A partir de ahora deberás hacer los ejercicios a continuación. Cada apartado numerado debería de ser un único commit (el mensaje que pongas en cada uno es libre, pero como sugerencia puedes poner el nombre del ejercicio y el número del apartado), pero puedes hacer tantos commits como necesites.

## Segunda Parte: Python + GCP + BQ

- 1) Tienes que crear un script de Python que se conecte a una API y se descargue los datos (con 100 registros más que suficiente).  
Tiene que haber una clase para descargar de la API y otra para subirla a BigQuery. Haz un commit con esta parte y súbelo al repositorio como “parte2” (puedes hacer los commits que quieras pero indicando cada parte)

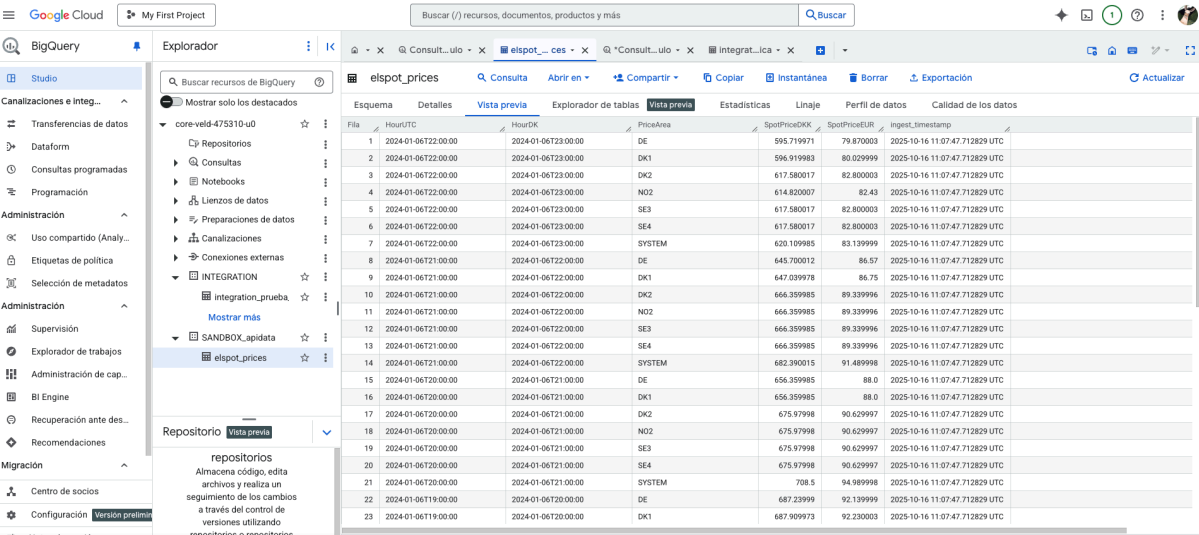


```
1
2 import os
3 import requests
4 import pandas as pd
5 from datetime import datetime, timedelta
6 from google.cloud import bigquery
7 from google.api_core.exceptions import NotFound
8
9 class EnerginetAPIClient:
10     """Descarga registros del dataset Elspotprices de Energinet."""
11     BASE_URL = "https://api.energidataservice.dk/dataset/Elspotprices"
12
13     def fetch_prices(self, start: str, end: str, limit: int = 100):
14         """Devuelve la lista de records JSON entre start y end (YYYY-MM-DD)."""
15         params = {"start": start, "end": end, "limit": limit}
16         r = requests.get(self.BASE_URL, params=params, timeout=20)
17         r.raise_for_status()
18         js = r.json()
19         records = js.get("records", [])
20         return records
21
22     def records_to_df(self, records):
23         """Normaliza los records en un DataFrame (limpieza mínima)."""
24         if not records:
25             return pd.DataFrame()
26         df = pd.json_normalize(records)
27         if "HourUTC" in df.columns:
28             df["HourUTC"] = pd.to_datetime(df["HourUTC"])
29         for col in ["SpotPriceDKK", "SpotPriceEUR"]:
30             if col in df.columns:
31                 df[col] = pd.to_numeric(df[col], errors="coerce")
32         df["ingest_timestamp"] = pd.Timestamp.utcnow()
33         return df
34
35 class BigQueryUploader:
36     """Sube DataFrame a BigQuery, crea dataset/tabla si hace falta."""
37     def __init__(self, project_id: str, dataset_id: str, location: str = "US"):
38         self.project = project_id
39         self.dataset = dataset_id
```

2) Los resultados de la conexión a la API los tienes que cargar en un DATASET que te crees en Bigquery.

a) El Dataset que reciba los datos de la API debe seguir esta nomenclatura:  
SANDBOX\_<nombre de tu aplicación>

Para esta parte, cuando lo tengas, puedes adjuntar captura de pantalla y subir el fichero que genera la tabla al repositorio.



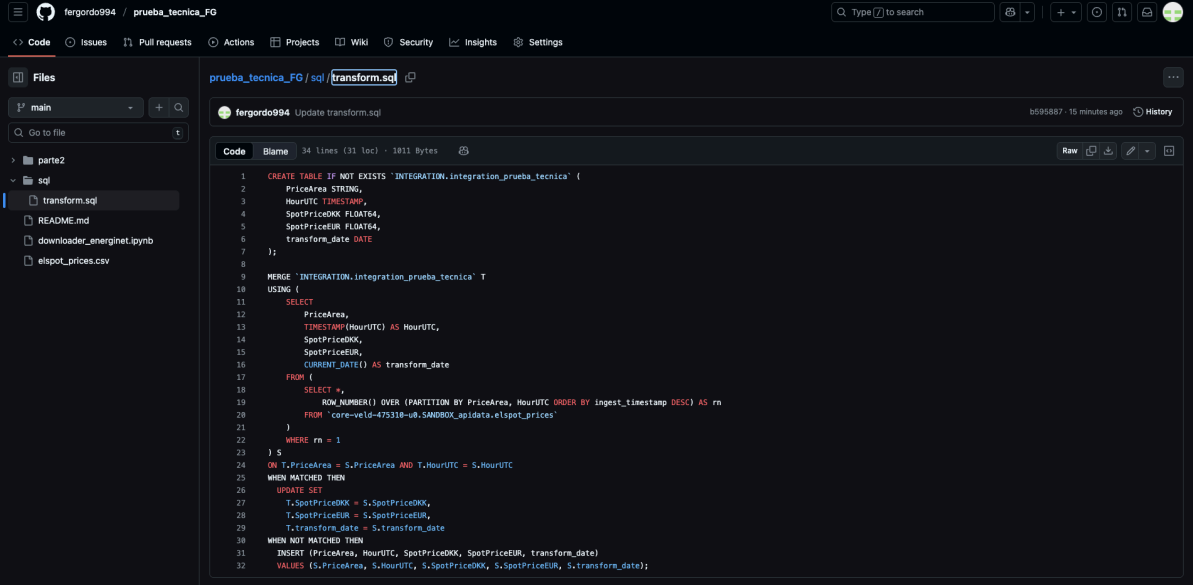
The screenshot displays the Google Cloud BigQuery interface. On the left, the 'Explorador' (Explorer) pane shows a project named 'core-velid-475310-u0' with a folder 'SANDBOX\_apidata' containing a table named 'elspot\_prices'. The main area shows the 'elspot\_prices' table with a 'Vista previa' (Preview) of its data. The table has columns for 'File', 'HourUTC', 'HourDK', 'PriceArea', 'SpotPriceDKK', 'SpotPriceEUR', and 'IngestTimestamp'. The data rows show hourly spot prices for various price areas (DE, DK1, DK2, NO2, SE3, SE4, SYSTEM) over a period from 2024-01-06T22:00:00 to 2024-01-06T23:00:00.

File	HourUTC	HourDK	PriceArea	SpotPriceDKK	SpotPriceEUR	IngestTimestamp
1	2024-01-06T22:00:00	2024-01-06T23:00:00	DE	595.719971	79.870003	2025-10-16 11:07:47.712829 UTC
2	2024-01-06T22:00:00	2024-01-06T23:00:00	DK1	596.919983	80.029999	2025-10-16 11:07:47.712829 UTC
3	2024-01-06T22:00:00	2024-01-06T23:00:00	DK2	617.580017	82.800003	2025-10-16 11:07:47.712829 UTC
4	2024-01-06T22:00:00	2024-01-06T23:00:00	NO2	614.820007	82.43	2025-10-16 11:07:47.712829 UTC
5	2024-01-06T22:00:00	2024-01-06T23:00:00	SE3	617.580017	82.800003	2025-10-16 11:07:47.712829 UTC
6	2024-01-06T22:00:00	2024-01-06T23:00:00	SE4	617.580017	82.800003	2025-10-16 11:07:47.712829 UTC
7	2024-01-06T22:00:00	2024-01-06T23:00:00	SYSTEM	620.109985	83.139999	2025-10-16 11:07:47.712829 UTC
8	2024-01-06T22:00:00	2024-01-06T23:00:00	DE	645.700012	86.57	2025-10-16 11:07:47.712829 UTC
9	2024-01-06T22:00:00	2024-01-06T23:00:00	DK1	647.039978	86.75	2025-10-16 11:07:47.712829 UTC
10	2024-01-06T22:00:00	2024-01-06T23:00:00	DK2	666.359985	89.339996	2025-10-16 11:07:47.712829 UTC
11	2024-01-06T22:00:00	2024-01-06T23:00:00	NO2	666.359985	89.339996	2025-10-16 11:07:47.712829 UTC
12	2024-01-06T22:00:00	2024-01-06T23:00:00	SE3	666.359985	89.339996	2025-10-16 11:07:47.712829 UTC
13	2024-01-06T22:00:00	2024-01-06T23:00:00	SE4	666.359985	89.339996	2025-10-16 11:07:47.712829 UTC
14	2024-01-06T22:00:00	2024-01-06T23:00:00	SYSTEM	682.390015	91.489998	2025-10-16 11:07:47.712829 UTC
15	2024-01-06T22:00:00	2024-01-06T23:00:00	DE	656.359985	88.0	2025-10-16 11:07:47.712829 UTC
16	2024-01-06T22:00:00	2024-01-06T23:00:00	DK1	656.359985	88.0	2025-10-16 11:07:47.712829 UTC
17	2024-01-06T22:00:00	2024-01-06T23:00:00	DK2	675.97998	90.629997	2025-10-16 11:07:47.712829 UTC
18	2024-01-06T22:00:00	2024-01-06T23:00:00	NO2	675.97998	90.629997	2025-10-16 11:07:47.712829 UTC
19	2024-01-06T22:00:00	2024-01-06T23:00:00	SE3	675.97998	90.629997	2025-10-16 11:07:47.712829 UTC
20	2024-01-06T22:00:00	2024-01-06T23:00:00	SE4	675.97998	90.629997	2025-10-16 11:07:47.712829 UTC
21	2024-01-06T22:00:00	2024-01-06T23:00:00	SYSTEM	708.5	94.989998	2025-10-16 11:07:47.712829 UTC
22	2024-01-06T22:00:00	2024-01-06T23:00:00	DE	687.23999	92.139999	2025-10-16 11:07:47.712829 UTC
23	2024-01-06T22:00:00	2024-01-06T23:00:00	DK1	687.909973	92.230003	2025-10-16 11:07:47.712829 UTC

Archivo subido al repositorio.

3) Vas a transformar los datos del sandbox. Dentro del repositorio, debe haber una carpeta sql/ con al menos un archivo:

1. **transform.sql**: Este archivo debe contener una única consulta SQL que:
  - Lea los datos de la tabla almacenada en SANDBOX\_<nombre de tu aplicación>
  - Realice alguna transformación simple. Por ejemplo: eliminar posibles duplicados del día, añadir una columna con la fecha en que se ejecuta la transformación...
  - Inserte el resultado transformado en la tabla INTEGRATION.integration\_prueba\_tecnica.
  - **Requisito clave:** La consulta debe ser **idempotente**. Es decir, si se ejecuta varias veces sobre los mismos datos crudos del día, el resultado en la tabla final debe ser el mismo (no debe generar duplicados).



```
1 CREATE TABLE IF NOT EXISTS 'INTEGRATION.integration_prueba_tecnica' (  
2   PriceArea STRING,  
3   HourUTC TIMESTAMP,  
4   SpotPriceDKK FLOAT64,  
5   SpotPriceEUR FLOAT64,  
6   transform_date DATE  
7 );  
8  
9 MERGE 'INTEGRATION.integration_prueba_tecnica' T  
10 USING (  
11   SELECT  
12     PriceArea,  
13     TIMESTAMP(HourUTC) AS HourUTC,  
14     SpotPriceDKK,  
15     SpotPriceEUR,  
16     CURRENT_DATE() AS transform_date  
17   FROM (  
18     SELECT *  
19     ROW_NUMBER() OVER (PARTITION BY PriceArea, HourUTC ORDER BY ingest_timestamp DESC) AS rn  
20     FROM 'core-veld-475318-u0.SANDBOX_apidata.elspot_prices'  
21   )  
22   WHERE rn = 1  
23 ) S  
24 ON T.PriceArea = S.PriceArea AND T.HourUTC = S.HourUTC  
25 WHEN MATCHED THEN  
26   UPDATE SET  
27     T.SpotPriceDKK = S.SpotPriceDKK,  
28     T.SpotPriceEUR = S.SpotPriceEUR,  
29     T.transform_date = S.transform_date  
30 WHEN NOT MATCHED THEN  
31   INSERT (PriceArea, HourUTC, SpotPriceDKK, SpotPriceEUR, transform_date)  
32   VALUES (S.PriceArea, S.HourUTC, S.SpotPriceDKK, S.SpotPriceEUR, S.transform_date);
```

Explorador

Buscar recursos de BigQuery

Mostrar solo los destacados

core-veld-475310-u0

Repositorios

Consultas

Notebooks

Lienzos de datos

Preparaciones de datos

Canalizaciones

Conexiones externas

INTEGRATION

integration\_prueba

Mostrar más

SANDBOX\_apidata

elspot\_prices

Repositorio

Vista previa

repositorios

Almacena código, edita archivos y realiza un seguimiento de los cambios a través del control de versiones utilizando repositorios o repositorios

Consultas sin titulo

Ejecutar

Guardar

Descargar

Compartir

```
1 CREATE TABLE IF NOT EXISTS `INTEGRATION.integration_prueba_tecnica` (  
2   PriceArea STRING,  
3   HourUTC TIMESTAMP,  
4   SpotPriceDKK FLOAT64,  
5   SpotPriceEUR FLOAT64,  
6   transform_date DATE  
7 );  
8  
9 MERGE `INTEGRATION.integration_prueba_tecnica` T  
10 USING (  
11   SELECT  
12     PriceArea,  
13     TIMESTAMP(HourUTC) AS HourUTC,  
14     SpotPriceDKK,  
15     SpotPriceEUR,  
16     CURRENT_DATE() AS transform_date  
17   FROM (
```

Esta secuencia de comandos procesará 0 B cuando se ejecute.

Todos los resultados

Tiempo transcurrido

Declaraciones procesadas

Estado del trabajo

3 s

2

SUCCESS

Estado

Hora de finalización

SQL

Acción

4:11 p.m. [2:1]

CREATE TABLE IF NOT EXISTS `INTEGRATION.integration\_prueba\_tecnica`

Ver resultados

4:11 p.m. [11:1]

MERGE `INTEGRATION.integration\_prueba\_tecnica` T

Ver resultados

Explorador

Buscar recursos de BigQuery

Mostrar solo los destacados

core-veld-475310-u0

Repositorios

Consultas

Notebooks

Lienzos de datos

Preparaciones de datos

Canalizaciones

Conexiones externas

INTEGRATION

integration\_prueba

Mostrar más

SANDBOX\_apidata

elspot\_prices

Repositorio

Vista previa

repositorios

Almacena código, edita archivos y realiza un seguimiento de los cambios a través del control de versiones utilizando repositorios o repositorios

integration\_prueba\_tecnica

Consulta

Abrir en

Compartir

Copiar

Instantánea

Borrar

Exportación

Actualizar

Esquema

Detalles

Vista previa

Explorador de tablas

Vista previa

Estadísticas

Linaje

Perfil de datos

Calidad de los datos

Fila	PriceArea	HourUTC	SpotPriceDKK	SpotPriceEUR	transform_da...
1	DE	2024-01-01 08:00:00 UTC	0.3	0.04	2025-10-16
2	DK1	2024-01-01 08:00:00 UTC	0.6	0.08	2025-10-16
3	DE	2024-01-01 09:00:00 UTC	0.45	0.06	2025-10-16
4	SE3	2024-01-01 09:00:00 UTC	308.179993	41.34	2025-10-16
5	DK2	2024-01-01 09:00:00 UTC	308.179993	41.34	2025-10-16
6	DK1	2024-01-01 09:00:00 UTC	308.179993	41.34	2025-10-16
7	SE4	2024-01-01 09:00:00 UTC	308.179993	41.34	2025-10-16
8	SYSTEM	2024-01-01 09:00:00 UTC	314.070007	42.130001	2025-10-16
9	NO2	2024-01-01 09:00:00 UTC	355.670013	47.709999	2025-10-16
10	DE	2024-01-01 10:00:00 UTC	4.03	0.54	2025-10-16
11	DK1	2024-01-01 10:00:00 UTC	324.359985	43.509998	2025-10-16
12	DK2	2024-01-01 10:00:00 UTC	324.359985	43.509998	2025-10-16
13	SE3	2024-01-01 10:00:00 UTC	324.359985	43.509998	2025-10-16
14	SE4	2024-01-01 10:00:00 UTC	324.359985	43.509998	2025-10-16
15	SYSTEM	2024-01-01 10:00:00 UTC	333.679993	44.759998	2025-10-16
16	NO2	2024-01-01 10:00:00 UTC	391.450012	52.509998	2025-10-16
17	DE	2024-01-01 11:00:00 UTC	16.700001	2.24	2025-10-16
18	DK1	2024-01-01 11:00:00 UTC	320.700012	43.02	2025-10-16
19	DK2	2024-01-01 11:00:00 UTC	320.700012	43.02	2025-10-16
20	SE4	2024-01-01 11:00:00 UTC	320.700012	43.02	2025-10-16
21	SE3	2024-01-01 11:00:00 UTC	320.700012	43.02	2025-10-16
22	SYSTEM	2024-01-01 11:00:00 UTC	335.609985	45.02	2025-10-16
23	NO2	2024-01-01 11:00:00 UTC	421.720001	56.57	2025-10-16
24	DE	2024-01-01 12:00:00 UTC	14.61	1.06	2025-10-16

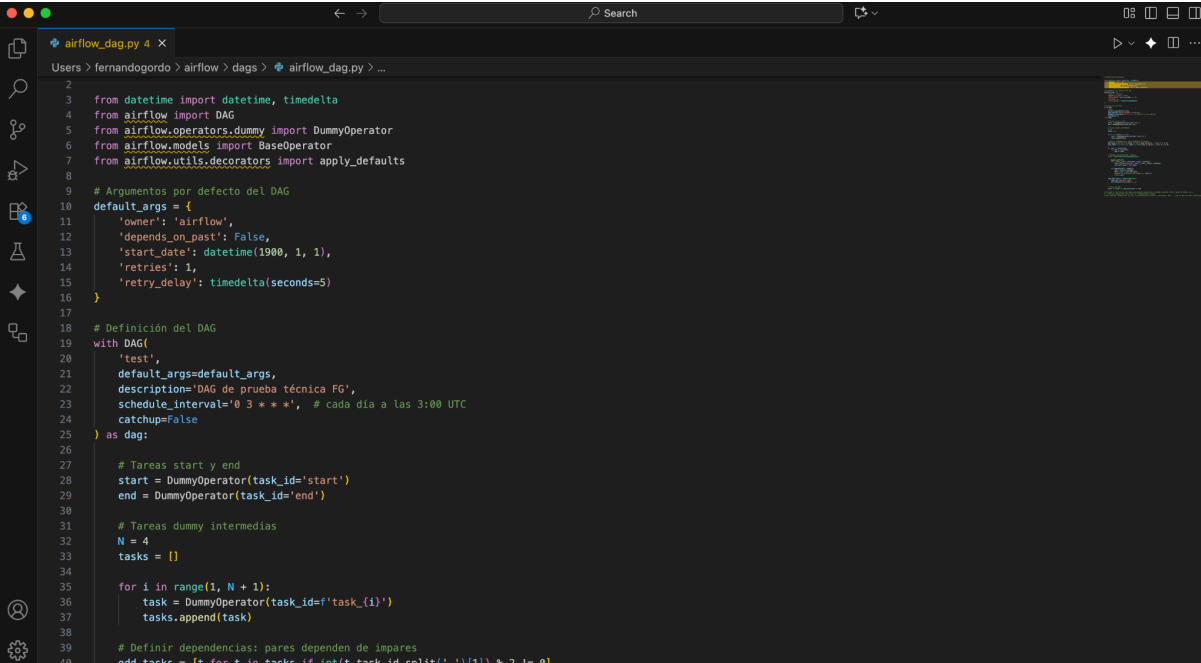
## Tercera Parte: Airflow

Para esta prueba vas a necesitar tener el módulo Airflow instalado en local o en una máquina de docker para poder lanzarlo.

- 1) Define un DAG llamado **test** que se ejecuta cada día a las 3:00 UTC, con los siguientes argumentos por defecto:

```
from datetime import datetime, timedelta
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(1900, 1, 1),
    'retries': 1,
    'retry_delay': timedelta(seconds=5)
}
```

Creamos el archivo script airflow\_dag.py para crear el dag test.

A screenshot of a code editor window titled 'airflow\_dag.py 4'. The editor shows a Python script for creating an Airflow DAG. The script includes imports for datetime, timedelta, DAG, DummyOperator, BaseOperator, and apply\_defaults. It defines default\_args with owner 'airflow', no dependencies on past tasks, a start date of 1900-01-01, 1 retry, and a 5-second retry delay. The DAG is named 'test', has the same default\_args, a description 'DAG de prueba técnica FG', a daily schedule at 3:00 UTC, and no catchup. It defines a start task, an end task, and a loop of 4 dummy tasks. Finally, it sets dependencies so that even-numbered tasks depend on the previous odd-numbered task.

```
2
3 from datetime import datetime, timedelta
4 from airflow import DAG
5 from airflow.operators.dummy import DummyOperator
6 from airflow.models import BaseOperator
7 from airflow.utils.decorators import apply_defaults
8
9 # Argumentos por defecto del DAG
10 default_args = {
11     'owner': 'airflow',
12     'depends_on_past': False,
13     'start_date': datetime(1900, 1, 1),
14     'retries': 1,
15     'retry_delay': timedelta(seconds=5)
16 }
17
18 # Definición del DAG
19 with DAG(
20     'test',
21     default_args=default_args,
22     description='DAG de prueba técnica FG',
23     schedule_interval='0 3 * * *', # cada día a las 3:00 UTC
24     catchup=False
25 ) as dag:
26
27     # Tareas start y end
28     start = DummyOperator(task_id='start')
29     end = DummyOperator(task_id='end')
30
31     # Tareas dummy intermedias
32     N = 4
33     tasks = []
34
35     for i in range(1, N + 1):
36         task = DummyOperator(task_id=f'task_{i}')
37         tasks.append(task)
38
39     # Definir dependencias: pares dependen de impares
40     odd_tasks = [t for t in tasks if int(t.task_id.split('.')[1]) % 2 != 0]
```

Comprobando en terminal con airflow dags list, vemos que se ha creado correctamente

```
(venv) (base) fernandogordo@CARTOs-MacBook-Pro dags % airflow dags list
```

latest_only_with_trigger	/Users/fernandogordo/venv/lib/python3.11/site-packages/airflow/example_dags/example_latest_only_with_trigger.py	airflow	True
test	/Users/fernandogordo/airflow/dags/airflow_dag.py	airflow	True
tutorial	/Users/fernandogordo/venv/lib/python3.11/site-packages/airflow/example_dags/tutorial.py	airflow	True
tutorial_dag	/Users/fernandogordo/venv/lib/python3.11/site-packages	airflow	True

2) Incluye las tareas **start** y **end** como **DummyOperator** y que end vaya detrás de start en el DAG.

```
# Tareas start y end
start = DummyOperator(task_id='start')
end = DummyOperator(task_id='end')
```

Podemos verificar esto y la estructura de nuestro dag con el siguiente comando: airflow dags show test

```
(venv) (base) fernandogordo@CARTOs-MacBook-Pro dags % airflow dags show test
```

```
[2025-10-17T11:34:03.419+0200] {workday.py:41} WARNING - Could not import pandas. Holidays will not be considered.
digraph test {
  graph [label=test labelloc=t rankdir=LR]
  end [color="#000000" fillcolor="#e8f7e4" label=end shape=rectangle style="filled,rounded"]
  start [color="#000000" fillcolor="#e8f7e4" label=start shape=rectangle style="filled,rounded"]
  task_1 [color="#000000" fillcolor="#e8f7e4" label=task_1 shape=rectangle style="filled,rounded"]
  task_2 [color="#000000" fillcolor="#e8f7e4" label=task_2 shape=rectangle style="filled,rounded"]
  task_3 [color="#000000" fillcolor="#e8f7e4" label=task_3 shape=rectangle style="filled,rounded"]
  task_4 [color="#000000" fillcolor="#e8f7e4" label=task_4 shape=rectangle style="filled,rounded"]
  time_diff_task [color="#000000" fillcolor="#ffffff" label=time_diff_task shape=rectangle style="filled,rounded"]
  start --> task_1
  start --> task_2
  start --> task_3
  start --> task_4
  task_1 --> task_2
  task_1 --> task_4
  task_1 --> time_diff_task
  task_2 --> time_diff_task
  task_3 --> task_2
  task_3 --> task_4
  task_3 --> time_diff_task
  task_4 --> time_diff_task
  time_diff_task --> end
}
```

3) Define una lista de tareas dummy **task\_n** con **N** tareas, donde cada tarea con **n** par dependa de todas las tareas impares.

En el pantallazo anterior, vemos como task 1 esta relacionada con task 2 y task 4, mientras que task 3 se relaciona con task 2 y 4 también. Pares con impares.

Podemos sacar un listado de las diferentes tareas:

```
(venv) (base) fernandogordo@CARTOs-MacBook-Pro dags % airflow tasks list test
end
start
task_1
task_2
task_3
task_4
time_diff_task
```

- 4) Define un nuevo operador **TimeDiff** que parta del **BaseOperator**, que reciba una fecha (**diff\_date**) como entrada y muestre la diferencia con la actual. Crea una tarea nueva con el operador.

En esta parte del código creamos un operador personalizado como clase, que lo hace es decirnos la diferencia entre la fecha diff, que hemos dicho que sea 2024,1,1 y la fecha actual calculada mediante `datetime.now`

```
# Operador personalizado: TimeDiff
class TimeDiffOperator(BaseOperator):

    @apply_defaults
    def __init__(self, diff_date, *args, **kwargs):
        super(TimeDiffOperator, self).__init__(*args, **kwargs)
        self.diff_date = diff_date

    def execute(self, context):
        now = datetime.utcnow()
        diff = now - self.diff_date
        print(f"La diferencia de tiempo es: {diff}")
        return diff

time_diff_task = TimeDiffOperator(
    task_id='time_diff_task',
    diff_date=datetime(2024, 1, 1)
)
```

Si queremos probar que funciona correctamente, podemos probar de la siguiente manera:

airflow tasks test test time\_diff\_task 2025-10-17

```
time_diff_task
(venv) (base) fernandogordo@CARTOs-MacBook-Pro dags % airflow tasks test test time_diff_task 2025-10-17

La diferencia de tiempo es: 655 days, 9:41:21.691115
[2025-10-17T11:41:21.691+0200] {taskinstance.py:441} INFO - ::group::Post task execution logs
[2025-10-17T11:41:21.693+0200] {taskinstance.py:1206} INFO - Marking task as SUCCESS. dag_id=test,
te=20251017T000000, start_date=, end_date=20251017T094121
```

- 5) ¿Qué es un Hook? ¿En qué se diferencia de una conexión? Puedes responder en un comentario dentro del código.

Respuesta incluida dentro del código.