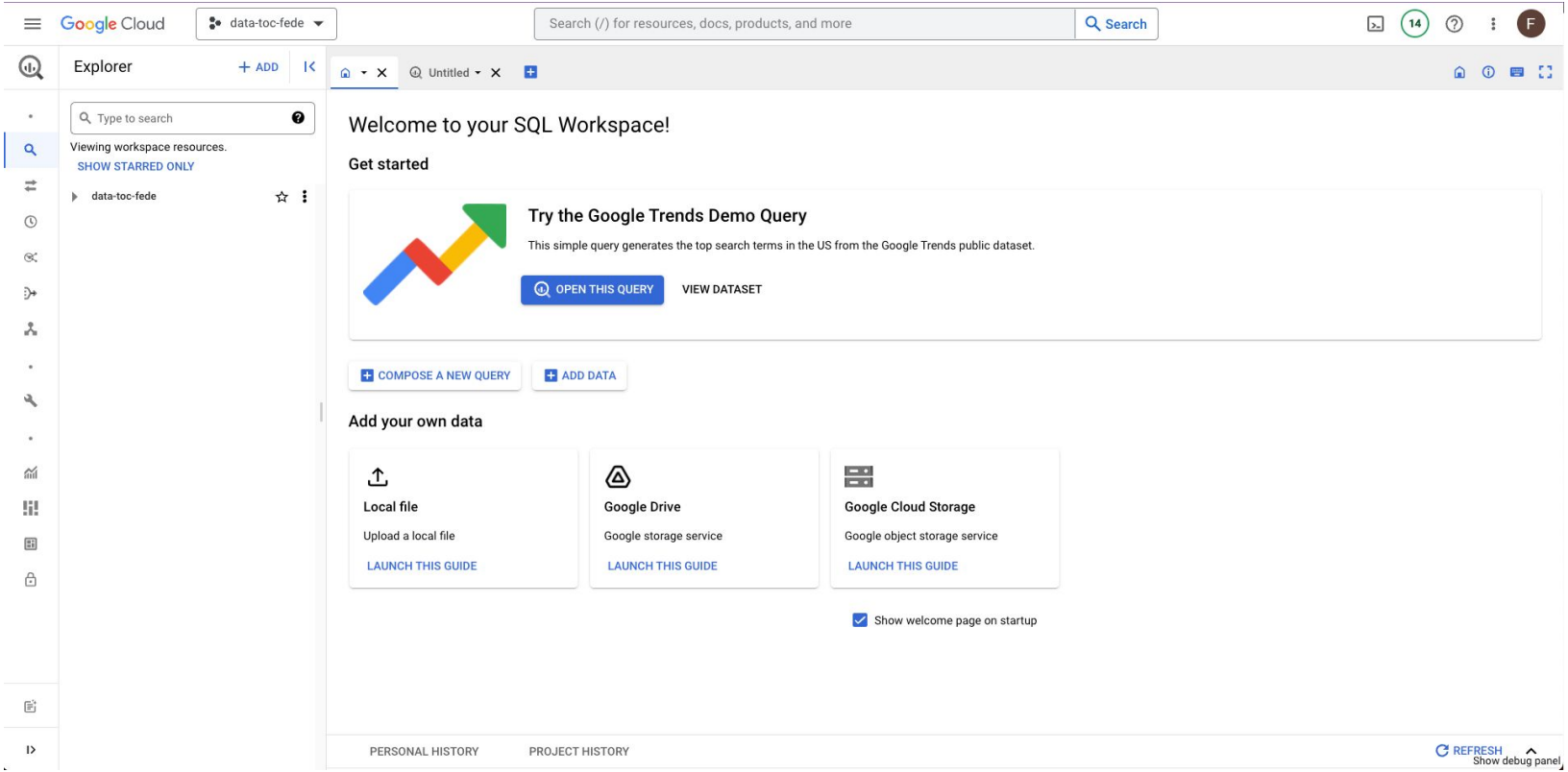
Escuela de Datos Vivos

Curso Data Engineer: Creando un pipeline de datos

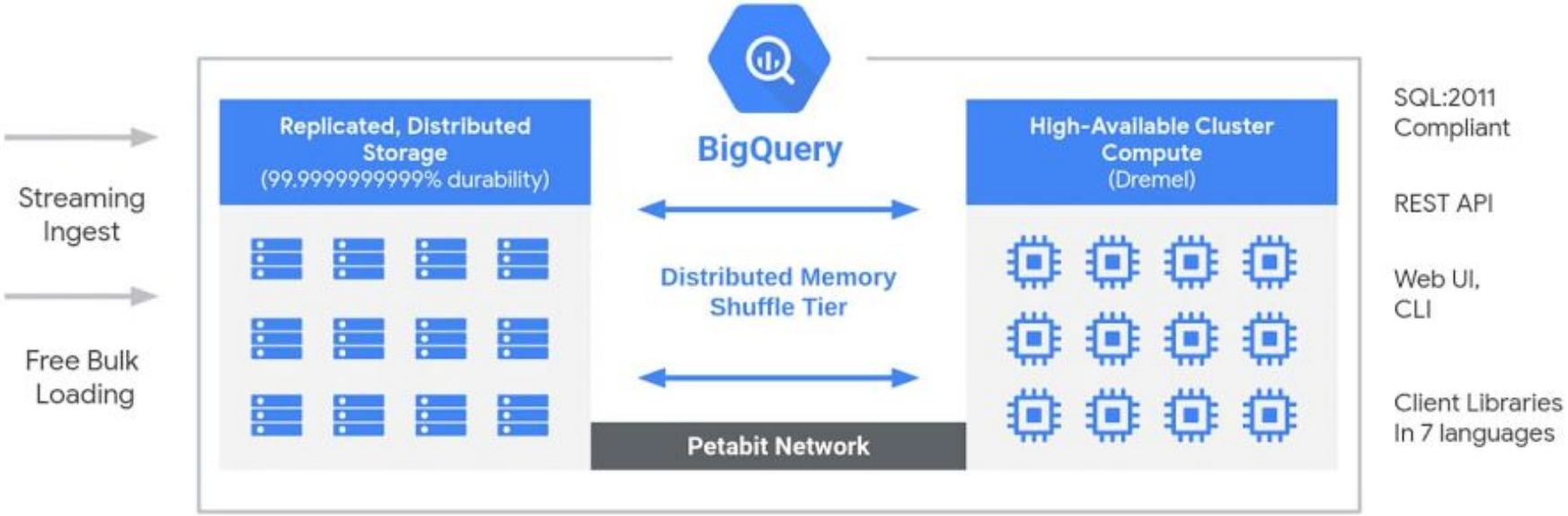
Módulo E - Clase 8

Cloud Datawarehouse

BIG QUERY


Google Cloud
data-toc-fede
Search (/) for resources, docs, products, and more
Search
Explorer
+ ADD
Type to search
Viewing workspace resources.
SHOW STARRED ONLY
data-toc-fede
Untitled
Welcome to your SQL Workspace!
Get started
Try the Google Trends Demo Query
This simple query generates the top search terms in the US from the Google Trends public dataset.
OPEN THIS QUERY
VIEW DATASET
COMPOSE A NEW QUERY
ADD DATA
Add your own data
Local file
Upload a local file
LAUNCH THIS GUIDE
Google Drive
Google storage service
LAUNCH THIS GUIDE
Google Cloud Storage
Google object storage service
LAUNCH THIS GUIDE
Show welcome page on startup
PERSONAL HISTORY
PROJECT HISTORY
REFRESH
Show debug panel

BIG QUERY


Streaming Ingest
Free Bulk Loading
Replicated, Distributed Storage
(99.9999999999% durability)
BigQuery
Distributed Memory Shuffle Tier
Petabit Network
High-Available Cluster Compute
(Dremel)
SQL:2011 Compliant
REST API
Web UI, CLI
Client Libraries In 7 languages

BIG QUERY

Ingestion time

```
bq query --destination_table mydataset.mytable
--time_partitioning_type=DAY
...
```

Any column that is of type DATE or TIMESTAMP

```
bq mk --table --schema a:STRING,tm:TIMESTAMP
--time_partitioning_field tm
```

Integer-typed column

```
bq mk --table --schema "customer_id:integer,value:integer"
--range_partitioning=customer_id,0,100,10 my_dataset.my_table
```

BIG QUERY

Particionamiento en BigQuery


```
SELECT c1, c3 FROM ...
WHERE eventDate BETWEEN
"2019-01-03" AND "2019-01-04"
```

BIG QUERY

Clustering en BigQuery

SELECT c1, c3, c5 FROM ...
WHERE eventDate BETWEEN “2019-01-03” AND “2019-01-04”

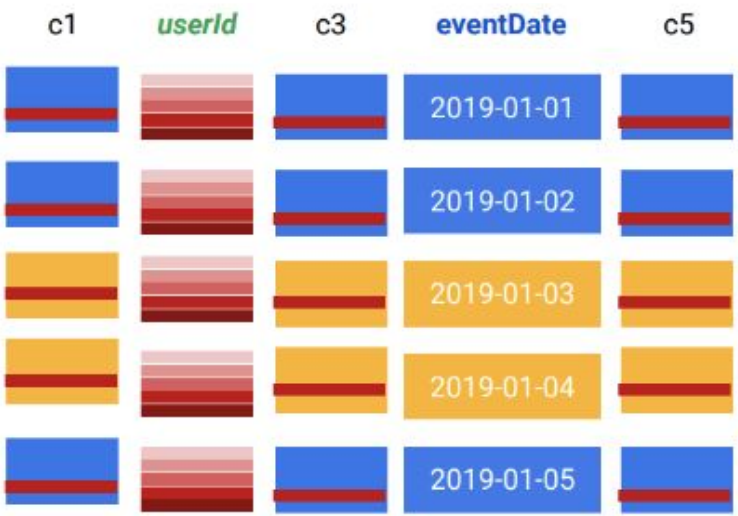
Partitioned tables

BIG QUERY

Clustering en BigQuery

SELECT c1, c3, c5 FROM ...
WHERE eventDate BETWEEN "2019-01-03" AND "2019-01-04"

Partitioned tables



SELECT c1, c3, c5 FROM ... WHERE userId BETWEEN 52 and 63 AND eventDate BETWEEN "2019-01-03" AND "2019-01-04"

Clustered tables

BIG QUERY

Clustered and Partitioned Tables

Orders table
Not Clustered; Not partitioned

Order_Date	Country	Status
2022-08-02	US	Shipped
2022-08-04	JP	Shipped
2022-08-05	UK	Canceled
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-05	US	Processing
2022-08-04	JP	Processing
2022-08-04	KE	Shipped
2022-08-06	UK	Canceled
2022-08-02	UK	Processing
2022-08-05	JP	Canceled
2022-08-06	UK	Processing
2022-08-05	US	Shipped
2022-08-06	JP	Processing
2022-08-02	KE	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country; Not partitioned

Order_Date	Country	Status
2022-08-04	JP	Shipped
2022-08-04	JP	Processing
2022-08-05	JP	Canceled
2022-08-06	JP	Processing
2022-08-06	KE	Shipped
2022-08-02	KE	Canceled
2022-08-04	KE	Shipped
2022-08-02	KE	Shipped
2022-08-05	UK	Processing
2022-08-06	UK	Canceled
2022-08-02	UK	Canceled
2022-08-06	UK	Processing
2022-08-02	US	Shipped
2022-08-05	US	Processing
2022-08-05	US	Shipped
2022-08-04	US	Shipped

Orders table
Clustered by Country; Partitioned by Order_Date (Daily)

	Order_Date	Country	Status
Partition: 2022-08-02	2022-08-02	KE	Shipped
	2022-08-02	KE	Canceled
Clusters: Country	2022-08-02	UK	Processing
	2022-08-02	US	Shipped

	Order_Date	Country	Status
Partition: 2022-08-04	2022-08-04	JP	Shipped
	2022-08-04	JP	Processing
Cluster: Country	2022-08-04	KE	Shipped
	2022-08-04	US	Shipped

	Order_Date	Country	Status
Partition: 2022-08-05	2022-08-05	JP	Canceled
	2022-08-05	UK	Canceled
Cluster: Country	2022-08-05	US	Shipped
	2022-08-05	US	Processing

	Order_Date	Country	Status
Partition: 2022-08-06	2022-08-06	JP	Processing
	2022-08-06	KE	Shipped
Cluster: Country	2022-08-06	UK	Canceled
	2022-08-06	UK	Processing

BIG QUERY

Como crear particiones en BigQuery

```
1  CREATE OR REPLACE TABLE bikes.trips_part
2  PARTITION BY DATE(starttime)
3  AS SELECT
4    *
5  FROM data-toc-fede.bikes.citibike_NY_trips;
6
```

BIG QUERY

Como crear particiones en BigQuery

```
CREATE TABLE
  mydataset.newtable (transaction_id INT64, transaction_date DATE)
PARTITION BY
  transaction_date
  OPTIONS (
    partition_expiration_days = 3,
    require_partition_filter = TRUE);
```

HIVE

Como crear particiones en Hive

```
create table table_partition (name string, lastname string)
partitioned by (id int)
row format delimited
fields terminated by ','
```

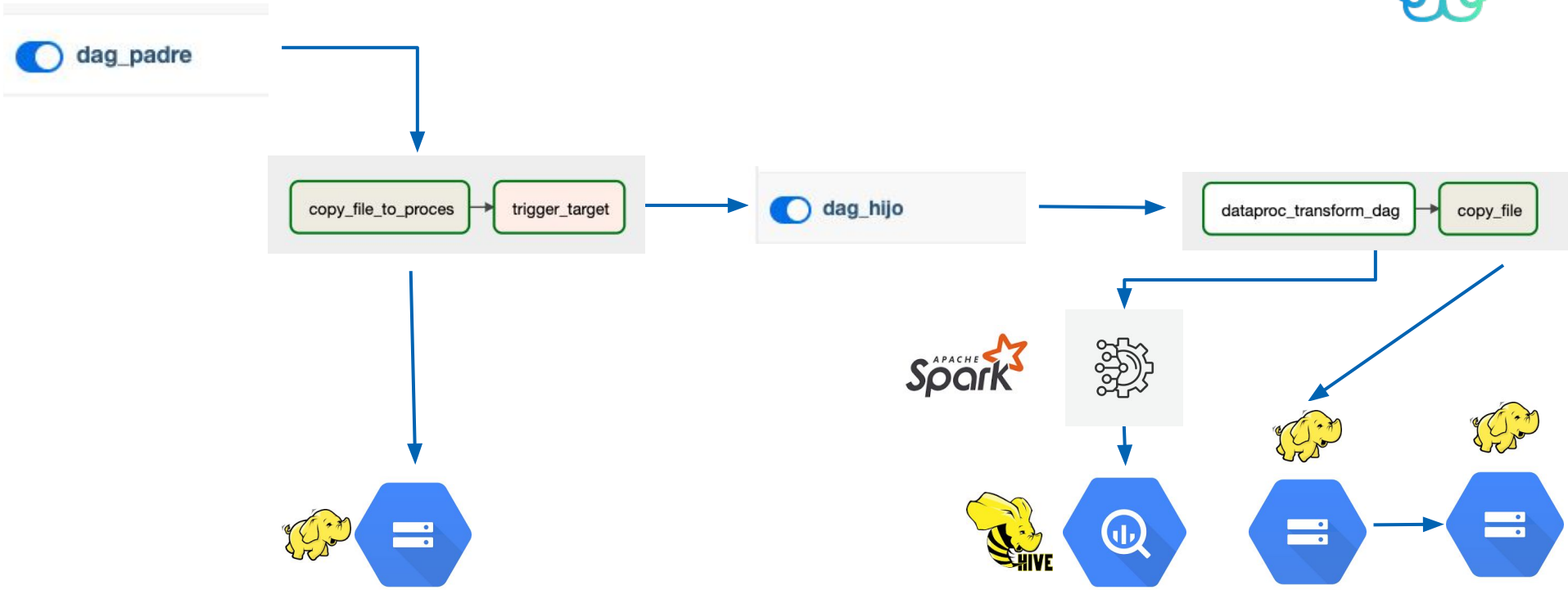
Airflow - Dependencia DAGs

Airflow

2 Dags

En casos que necesitemos ejecutar varias tareas dependientes en otros DAGs podemos utilizar el operador TriggerDagRunOperator, esto nos va a permitir desde un DAG padre o principal correr uno o varios DAGs hijos y esperar a que estos finalicen.

Airflow


dag_padre
copy_file_to_proces
trigger_target
dag_hijo
dataproc_transform_dag
copy_file
APACHE
Spark
HIVE

Airflow

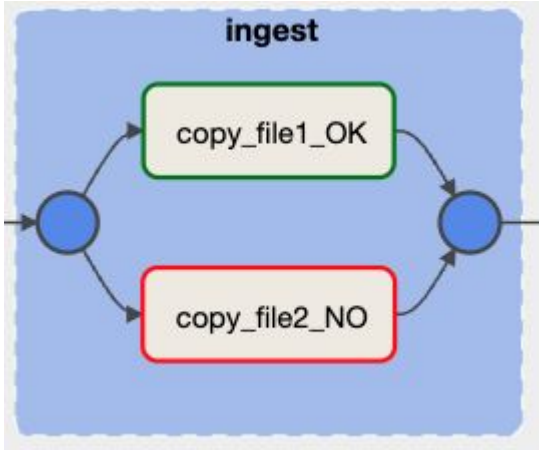
```
trigger_target = TriggerDagRunOperator(
    task_id = 'trigger_target',
    trigger_dag_id = 'dag_hijo',
    execution_date = '{{ ds }}',
    reset_dag_run = True,
    wait_for_completion = True,
    poke_interval = 30
)
```

- **reset_dag_run:** borra (clear) la ejecución de DAG existente si ya existe. Esto es útil cuando se hace un backfill o se vuelve a ejecutar una ejecución dag existente.
- **wait_for_completion:** Espera hasta que el dag hijo finalice. (default: False)
- **poke_interval:** Revisa cada x tiempo si el DAG hijo finalizó. (default: 60)

Airflow - Manejo de errores

Airflow

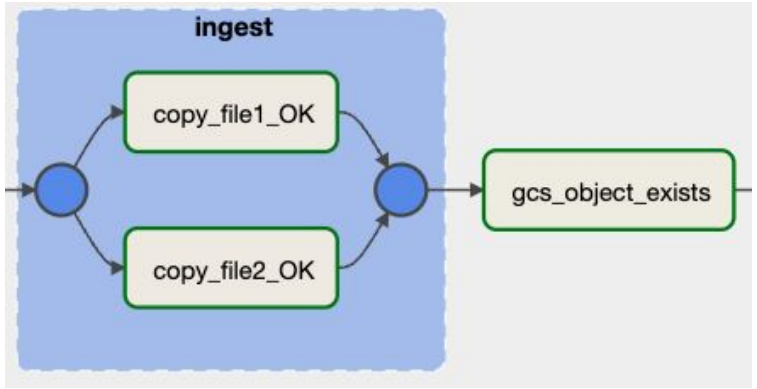
Task Group



Cuando necesitamos realizar ingesta de varias fuentes en un mismo DAG, una buena práctica es agrupar las tareas, de esta manera podremos independizar los logs para una fácil lectura e identificar fácilmente si hubo un error.

Airflow

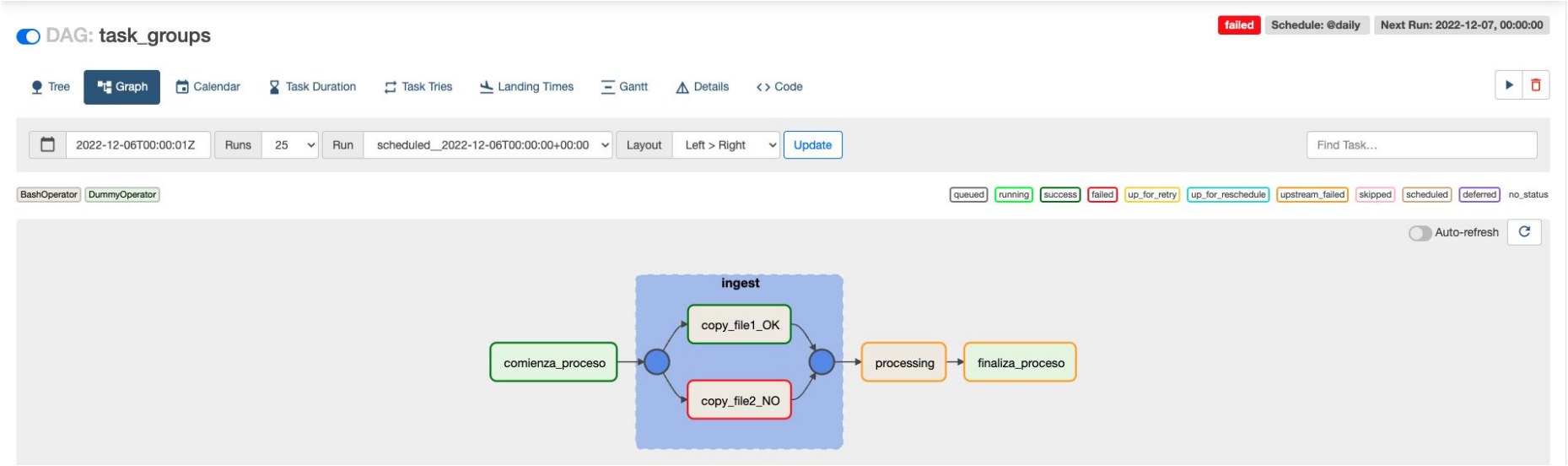
Task Group



Para controlar errores podremos hacerlo desde el mismo job, por medio de código o comandos shell.

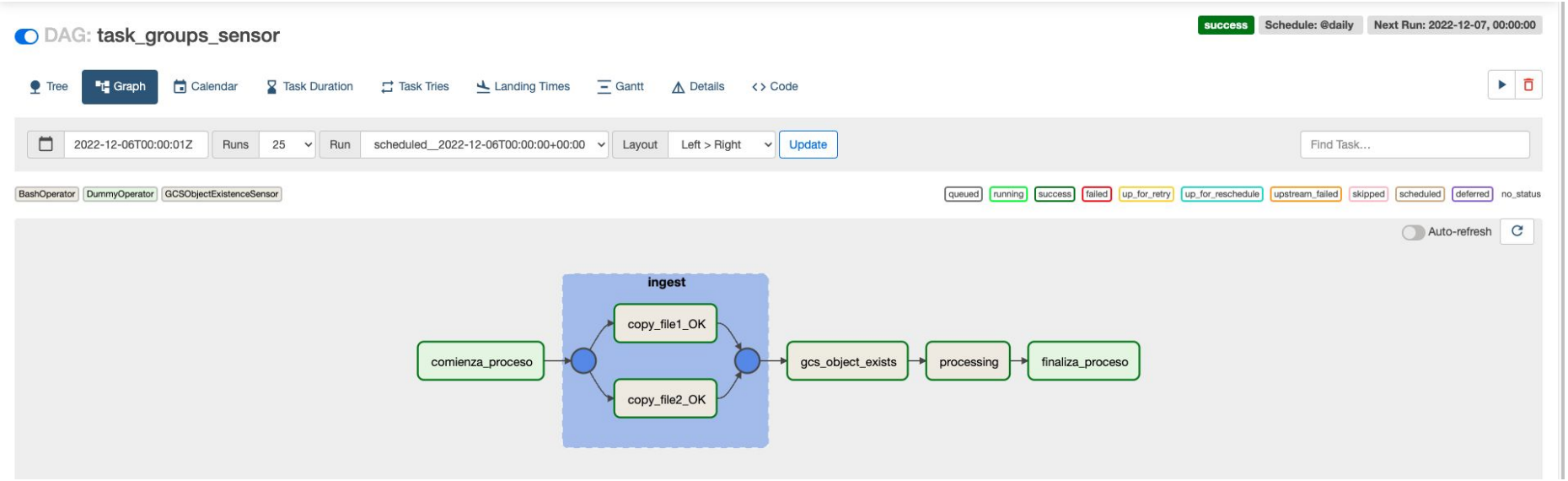
Otra opción es agregar un sensor luego de este, para determinar si efectivamente pudo obtener las fuentes.

Airflow



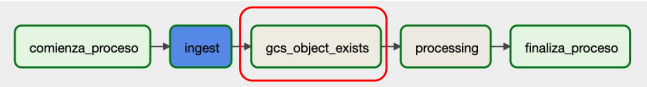
triger_rule: all_done

Airflow


DAG: task_groups_sensor
success
Schedule: @daily
Next Run: 2022-12-07, 00:00:00
Tree
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
2022-12-06T00:00:01Z
Runs
25
Run
scheduled__2022-12-06T00:00:00+00:00
Layout
Left > Right
Update
Find Task...
BashOperator
DummyOperator
GCSObjectExistenceSensor
queued
running
success
failed
up_for_retry
up_for_reschedule
upstream_failed
skipped
scheduled
deferred
no_status
Auto-refresh
ingest
copy_file1_OK
copy_file2_OK
comienza_proceso
gcs_object_exists
processing
finaliza_proceso

Airflow

File Sensor



Al agregar el sensor de files podemos determinar fehacientemente si los archivos ya están listos para ser procesados .

```
from airflow.contrib.sensors.file_sensor import FileSensor
```

```
from airflow.providers.google.cloud.sensors.gcs import GCSObjectExistenceSensor
```

```
from airflow.providers.apache.hdfs.sensors.hdfs import HdfsSensor
```

Airflow - envío de alertas por correo

```
[smtp]
# If you want airflow to send emails on retries, failure, and you want to use
# the airflow.utils.email.send_email_smtp function, you have to configure an
# smtp server here
smtp_host = your-smtp-host.com
smtp_starttls = True
smtp_ssl = False
# Uncomment and set the user/pass settings if you want to use SMTP AUTH
# smtp_user =
# smtp_password =
smtp_port = 587
smtp_mail_from = envio_alertas@mi_dominio.com
```

Configurar smtp en el archivo de configuración airflow.cfg

Airflow - envío de alertas por correo

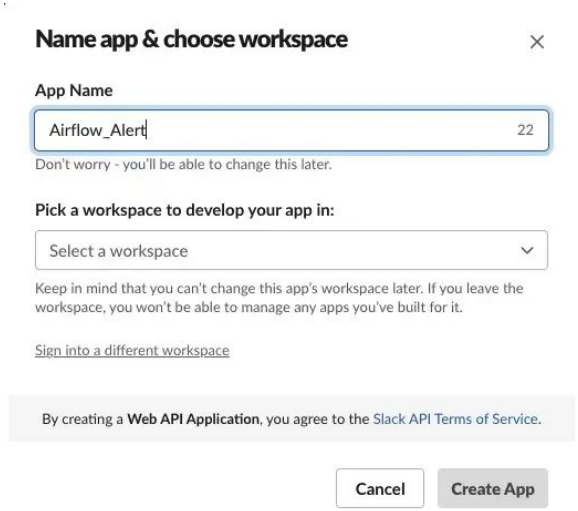
```python
from datetime import datetime
from airflow import DAG

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2022, 1, 1),
    'email': ['alerts@mi_dominio.com'],
    'email_on_failure': True
}

with DAG('sample_dag',
    default_args=default_args,
    schedule_interval='@daily',
    catchup=False) as dag:
```

En los default_args configurar en email la casilla de correo donde queremos que lleguen las alertas y configurar email_on_failure: True, de esta manera al fallar un job nos enviará un correo avisando que ese job ha fallado.

Airflow - envío de alertas por slack



Crear una nueva APP desde Slack

Airflow - envío de alertas por slack

Activate Incoming Webhooks

On

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include message attachments to display richly-formatted messages.

Adding incoming webhooks requires a bot user. If your app doesn't have a bot user, we'll add one for you.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to Revoke All OAuth Tokens.

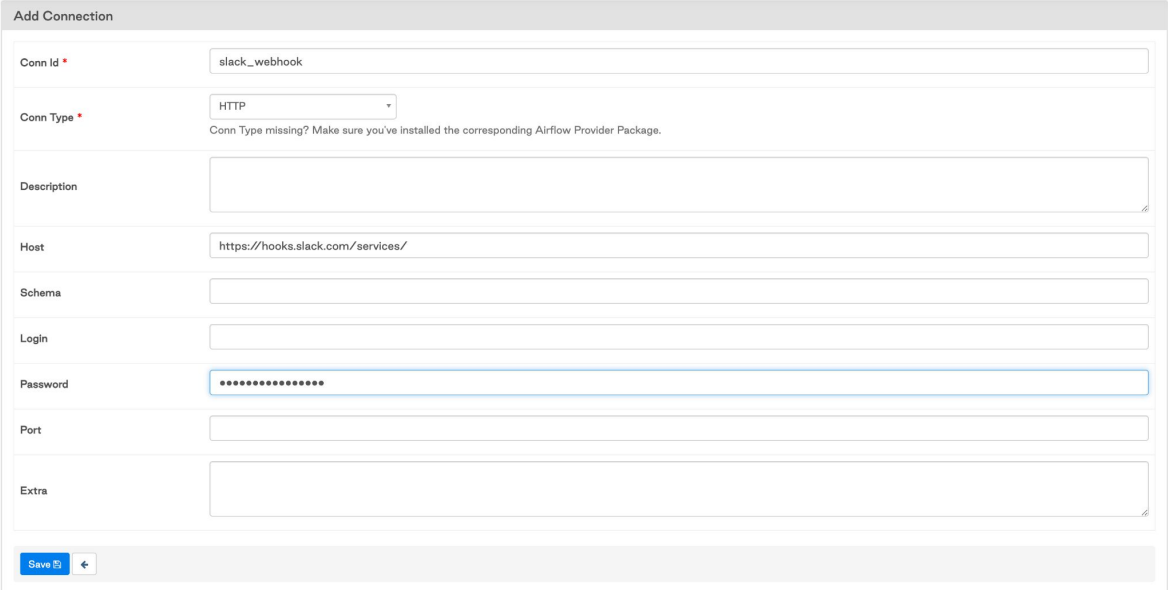
Activar entradas desde Webhooks

Airflow - envío de alertas por slack

https://hooks.slack.com/services/T***/B*****/*********

Copiar la URL del webhook

Airflow - envío de alertas por slack



Agregar una conexión en Airflow, conn type = HTTP, ingregar el host https://hooks.slack.com/services/

y la password la parte con el formato T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX

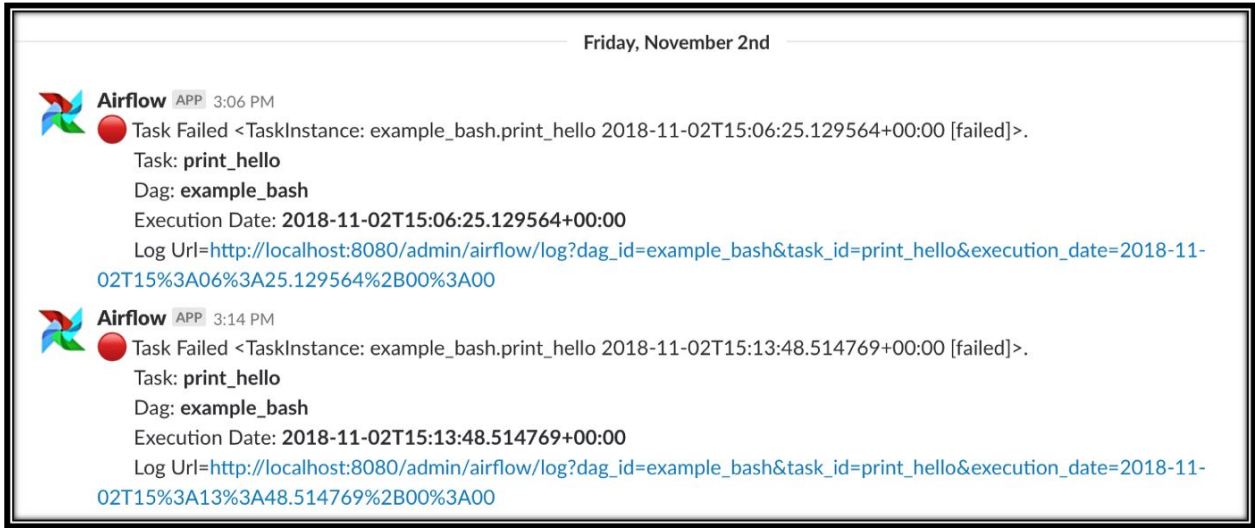
Airflow - envío de alertas por slack

```python
from airflow.providers.slack.operators.slack_webhook import SlackWebhookOperator

def slack_notification(context):
    slack_msg = """
            :red_circle: Task Failed.
            *Task*: {task}
            *Dag*: {dag}
            *Execution Time*: {exec_date}
            *Log Url*: {log_url}
            """.format(
            task=context.get('task_instance').task_id,
            dag=context.get('task_instance').dag_id,
            ti=context.get('task_instance'),
            exec_date=context.get('execution_date'),
            log_url=context.get('task_instance').log_url,
        )
    failed_alert = SlackWebhookOperator(
        task_id='slack_notification',
        http_conn_id='slack_webhook',
        message=slack_msg)
    return failed_alert.execute(context=context)
```

Luego en el DAG configurar los parámetros que queremos enviar a Slack

Airflow - envío de alertas por slack



Preguntas ?

Gracias