The Grumpy Data Pipeline

basado en hechos reales

$WHOAMI

Valery Calderón | @valerybriz

Senior Data Engineer en Capitole Consulting

Fundadora de la comunidad Python Guatemala

anteriormente, organizadora de Pyladies CDMX

Érase una vez...

En un Cloud Data Warehouse con datos de todos los tipos...

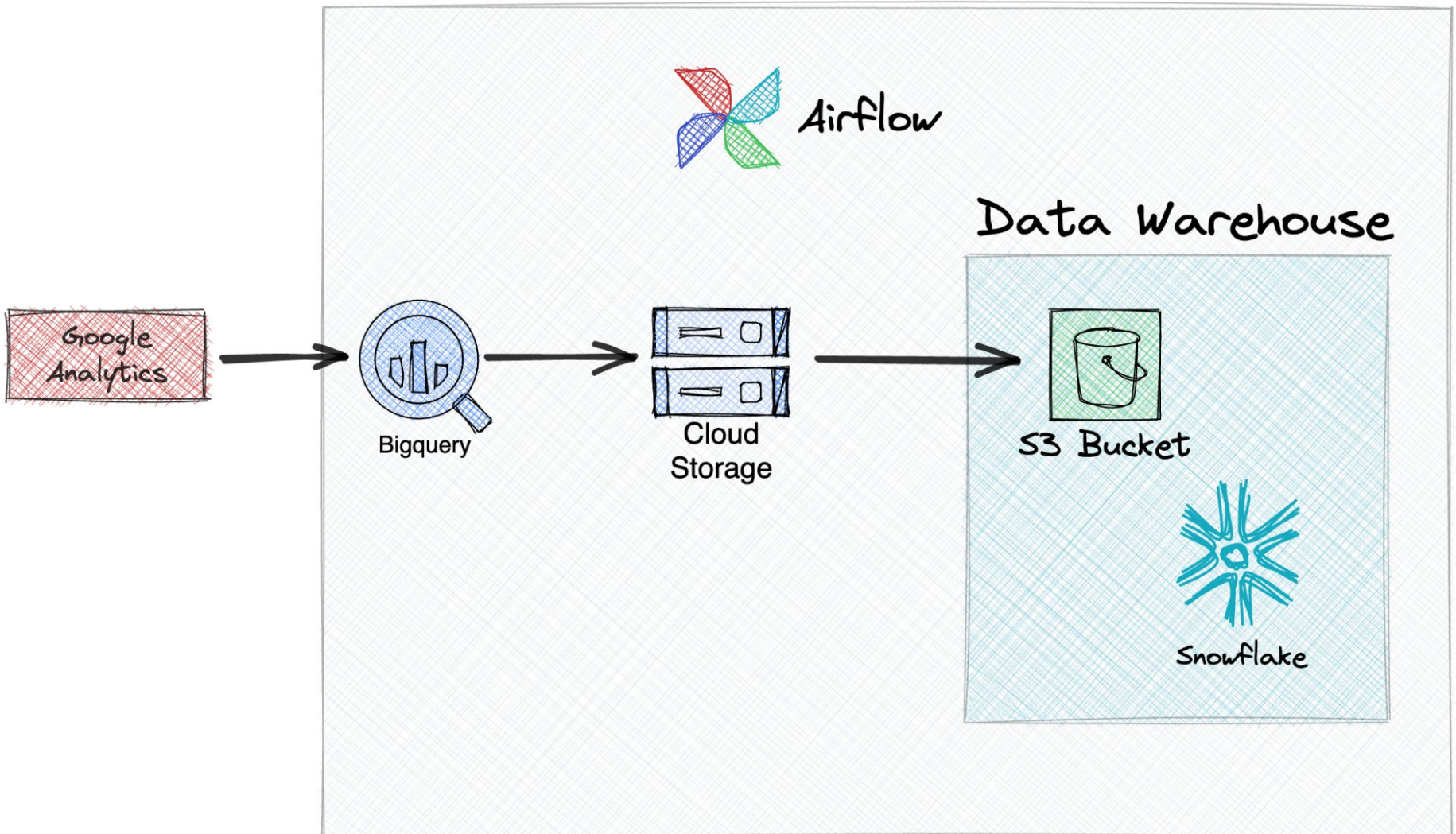
The grumpy data pipeline

En un Cloud Data Warehouse con datos de todos los tipos...

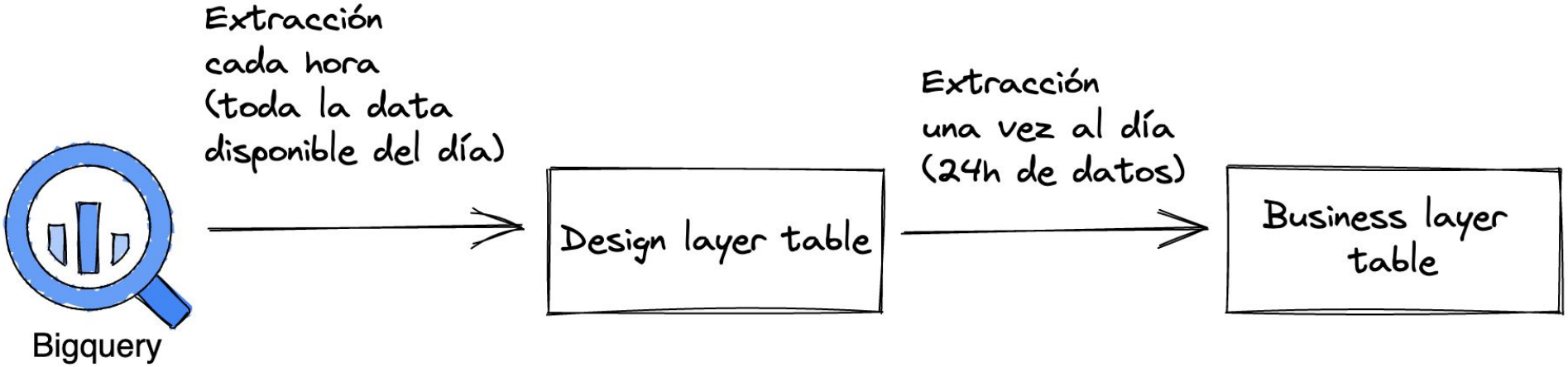
Existía un flujo de datos. Lento y muy liado...

Disclaimer

Disclaimer, los derechos sobre las imágenes del grumpy dwarf son de Disney... just in case


Airflow
Data Warehouse
Google Analytics
Bigquery
Cloud Storage
S3 Bucket
Snowflake

Conociendo un poco más al grumpy data pipeline



Pain Points

- Un SLA de 1 hora máximo de retraso en los datos “near-real time”.
- Time outs ocasionales en el DAG que extrae los datos y por ende en todo el pipeline (limite de tiempo de ejecución 1 hora debido al SLA).
- Data sets de distintos tamaños (entre 5 GB y 25 GB al día) como fuente de datos .
- Uno de los datasets tiene incluidos múltiples datasets que también se extraen por separado.

Pain Points

- Una vez en su tabla de destino, no hay forma de saber cuál fue la fuente de los datos.
- Una visualización en Airflow difícil de entender.
- Archivos históricos en GCS sin necesidad ya que también se tenía el histórico en BigQuery y en el Data Warehouse como Raw Data.
- GA3 exporta los datos de forma nativa a BigQuery en real time y full day (en 2 tablas distintas).

Pain Points

Y...

Casi nula documentación acerca del pipeline y de porqué funciona de esa forma actualmente.

Un día...

Un día surgió la necesidad de agregarle aún más fuentes de datos y el flujo empezó a fallar cada día...

Consecuencias

- Es necesario ejecutar manualmente todo el pipeline en caso de que falle.
- Ejecuciones sumamente largas, con un delay total de hasta 12 horas.
- Dificultad para rastrear los bugs en los datos hasta la fuente.
- Datos perdidos en medio del proceso del pipeline.
- Desconfianza en los datos.
- Consumo elevado de recursos para procesar los datos.
- Mezcla de dos grandes dominios de datos (B2B y B2C) y de dos estrategias de extracción en un mismo pipeline (@hourly y @daily).

La decisión...

Re - factorizar el pipeline completo (@daily y @hourly)

ó

Simplemente “poner un parche” (extrayendo únicamente los datos en el rango de tiempo necesario) @hourly.

La decisión...

La mayoría de las veces el principal conflicto en estas decisiones es

El tiempo.

Finalmente se ha decidido re-factorizar el pipeline

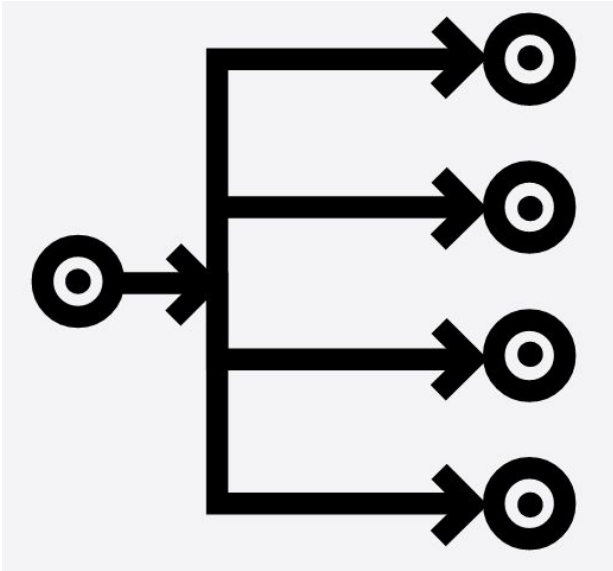
Al final del día, si únicamente hacíamos un arreglo simple, lo más seguro es que la cantidad de tiempo invertido en arreglar los demás errores y en tratar de obtener data confiable iba a ser igual o mayor al que se necesitaba para implementar la refactorización.

La propuesta

- Separar el procesamiento de cada sub dataset durante todo el pipeline, hasta llegar a la tabla final.

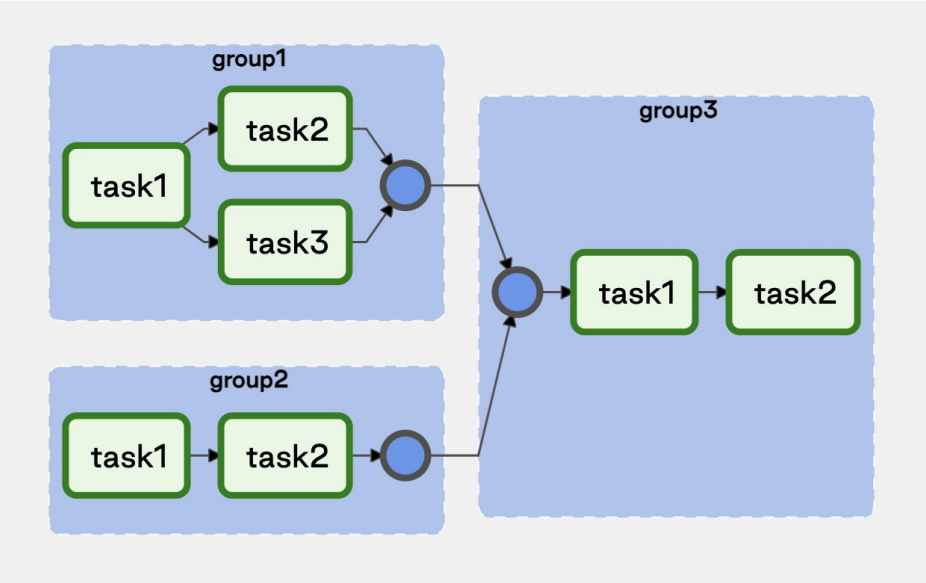
- Filtrar los datos desde Bigquery tanto por dataset como por rango de tiempo.

- Agregar identificadores de la fuente de los datos dentro de cada registro para poder identificarlos en cualquier punto del proceso.



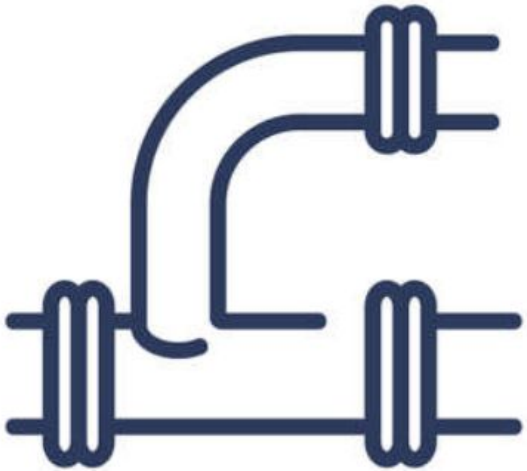
La propuesta

- Agrupar las tareas con Task Groups para lograr una visualización mucho más clara para las personas que mantienen y monitorean los DAG's.



La propuesta

- Separar el procesamiento de datos @hourly del procesamiento @daily en 2 pipelines.



La propuesta

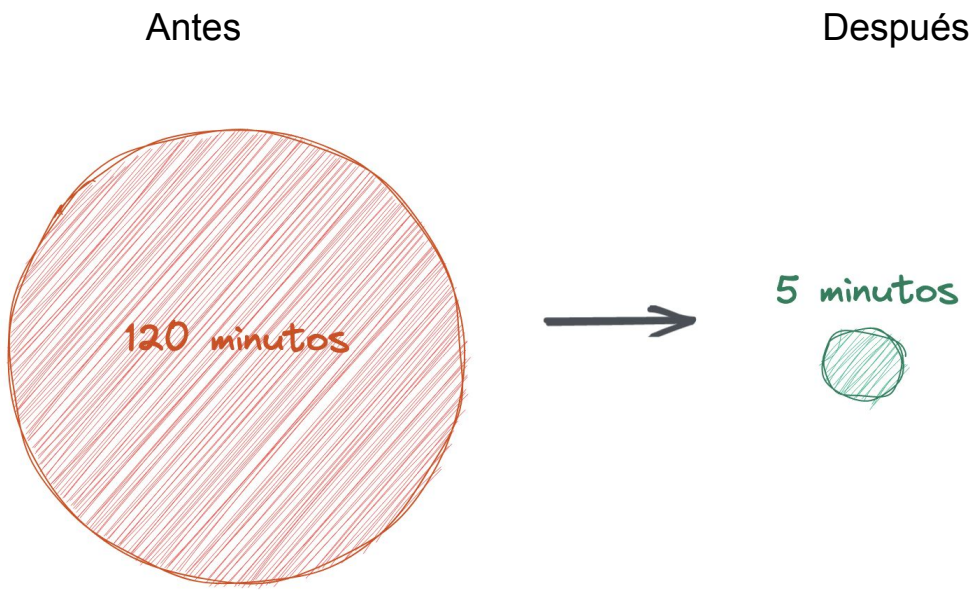
- Eliminar por completo el histórico que se guardaba en GCS y únicamente utilizarlo como un paso temporal al momento del procesamiento y finalmente limpiarlos.

Consecuencias

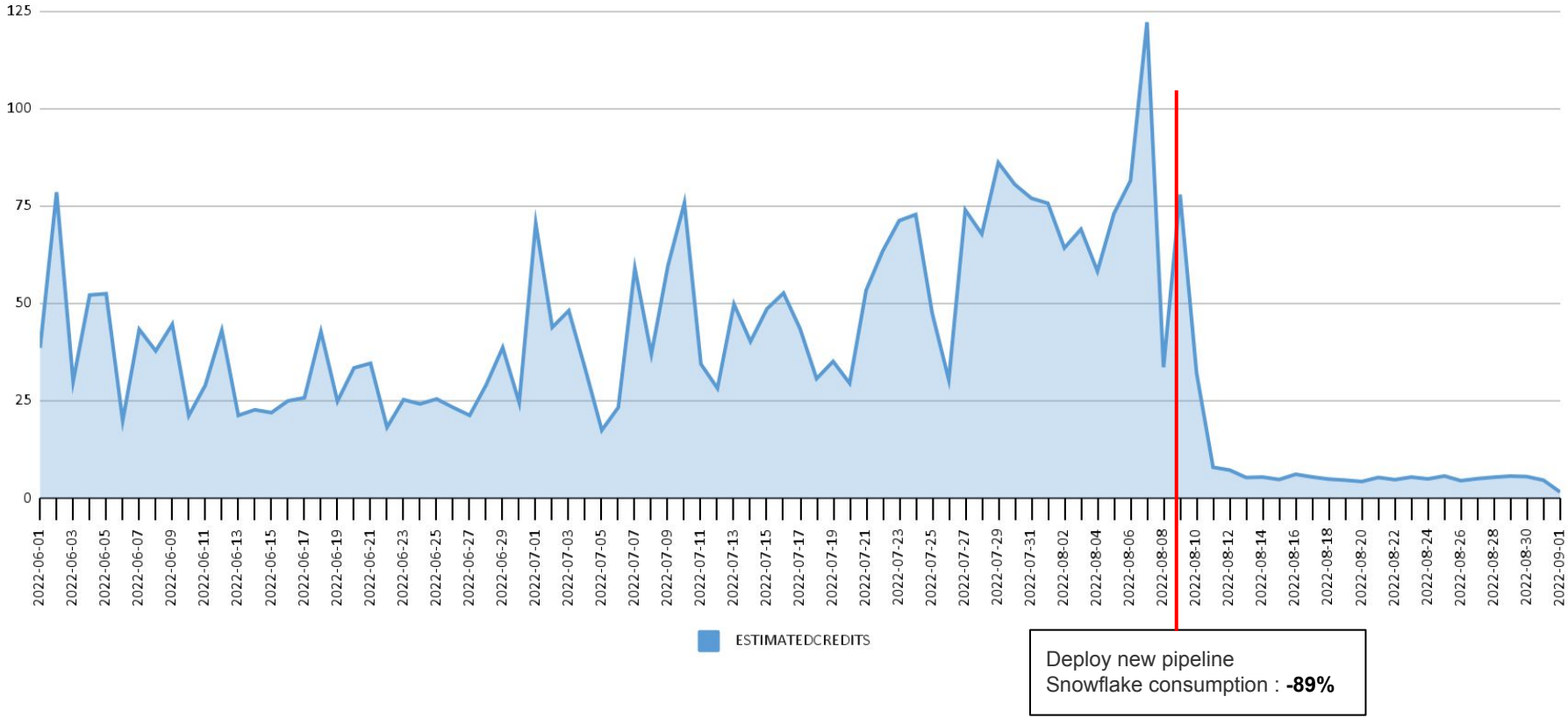
- Trazabilidad clara desde la fuente de los datos hasta su destino.
- 0 datos perdidos en el procesamiento.
- Cumplimiento diario de los SLA establecidos.
- Visualización clara de las tareas y DAG’s en Airflow.
- Confianza en los datos.
- Documentación clara.

Consecuencias

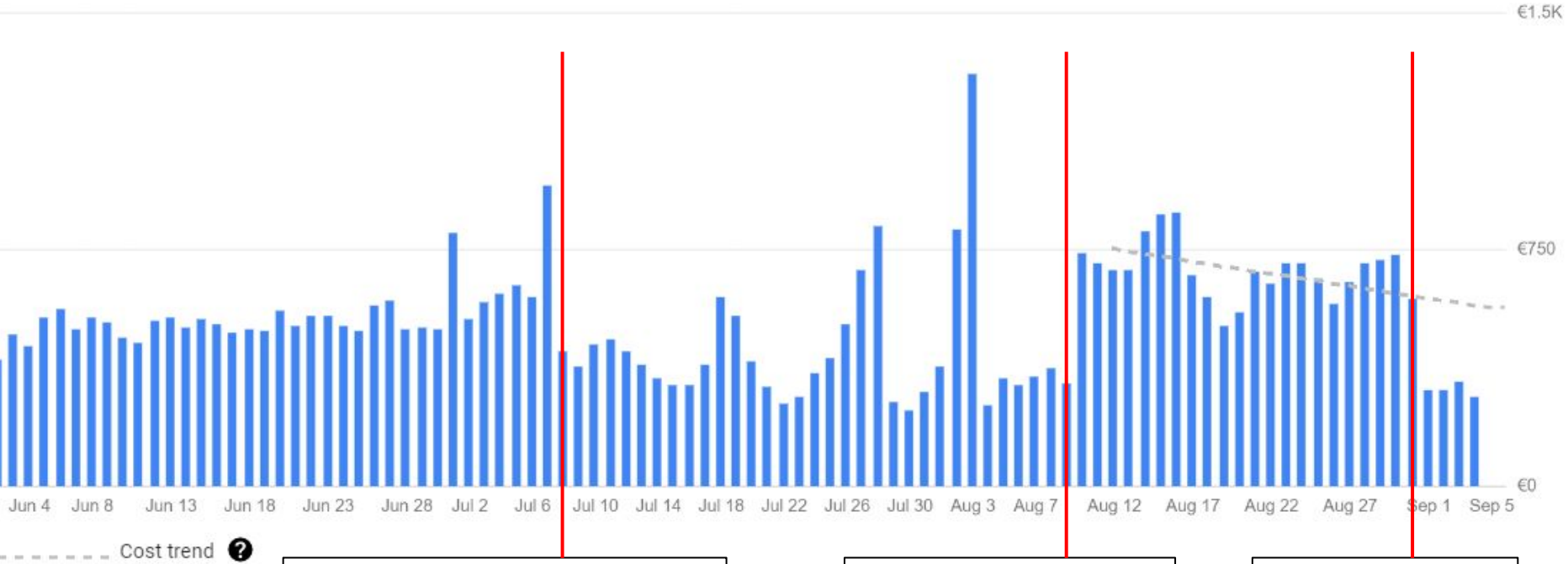
- DAG de extracción de datos 70% más rápido en el pipeline @daily y hasta 96% más rápido en el pipeline @hourly.



Reducción de costos en Snowflake



Impacto de los costos en BigQuery



Clean of GCS historic files
Costs : **9%**

Deploy new pipeline
Costs : **+39.5%**

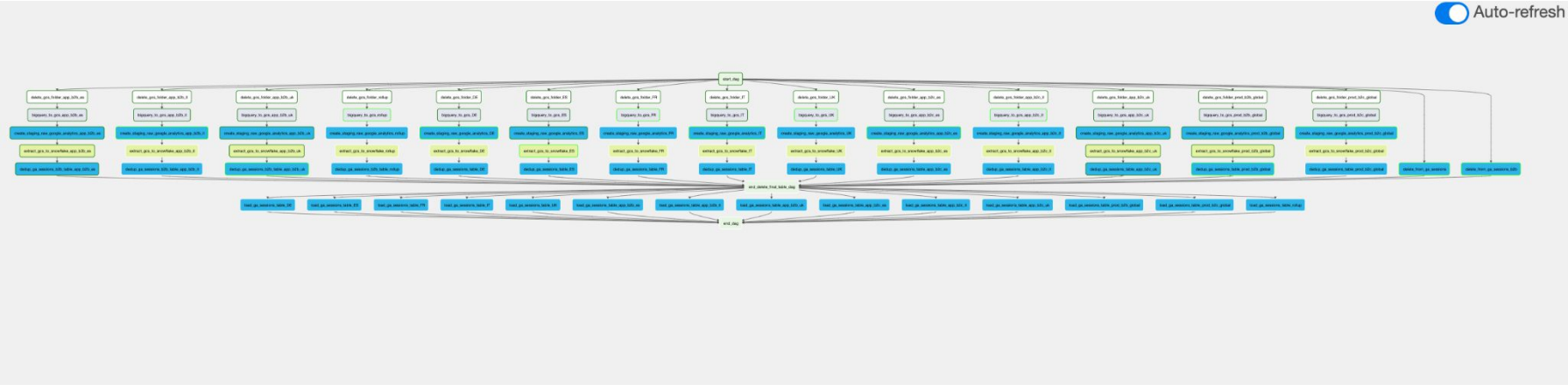
Whitelist events on GTM
Costs: **-28%**

Consecuencias

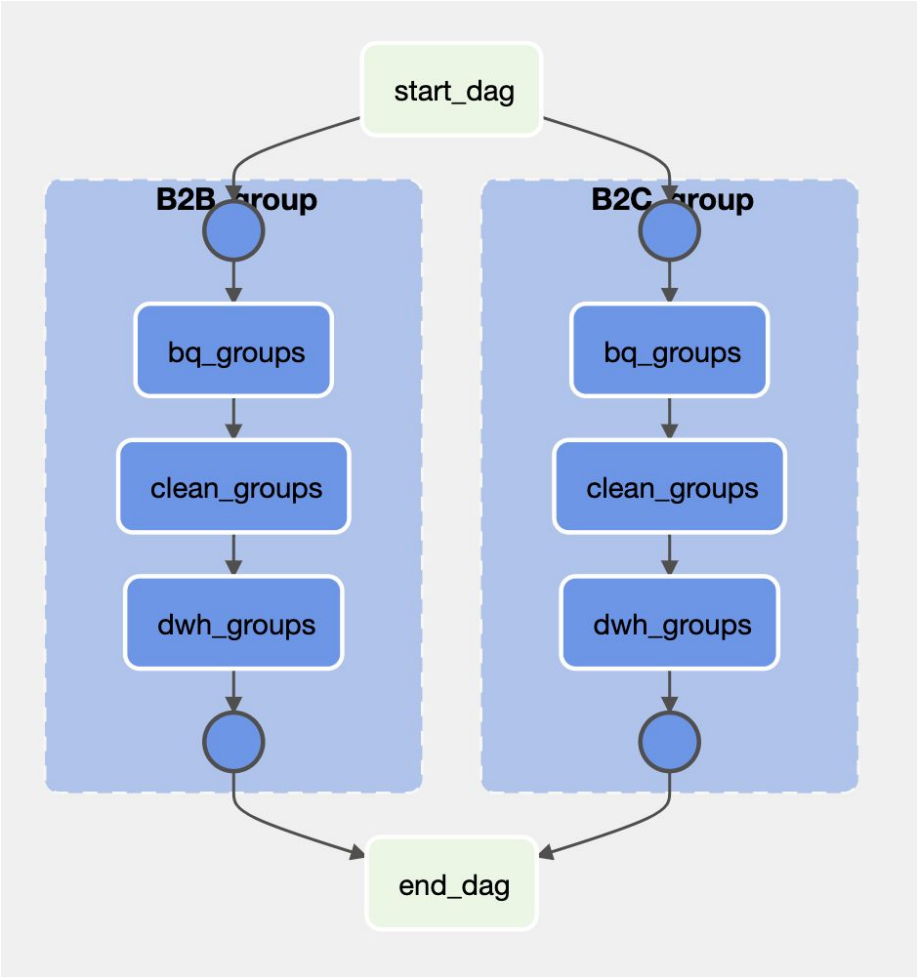
Reducción de fallos/time outs en un 95%.

Las mejoras visuales en Airflow

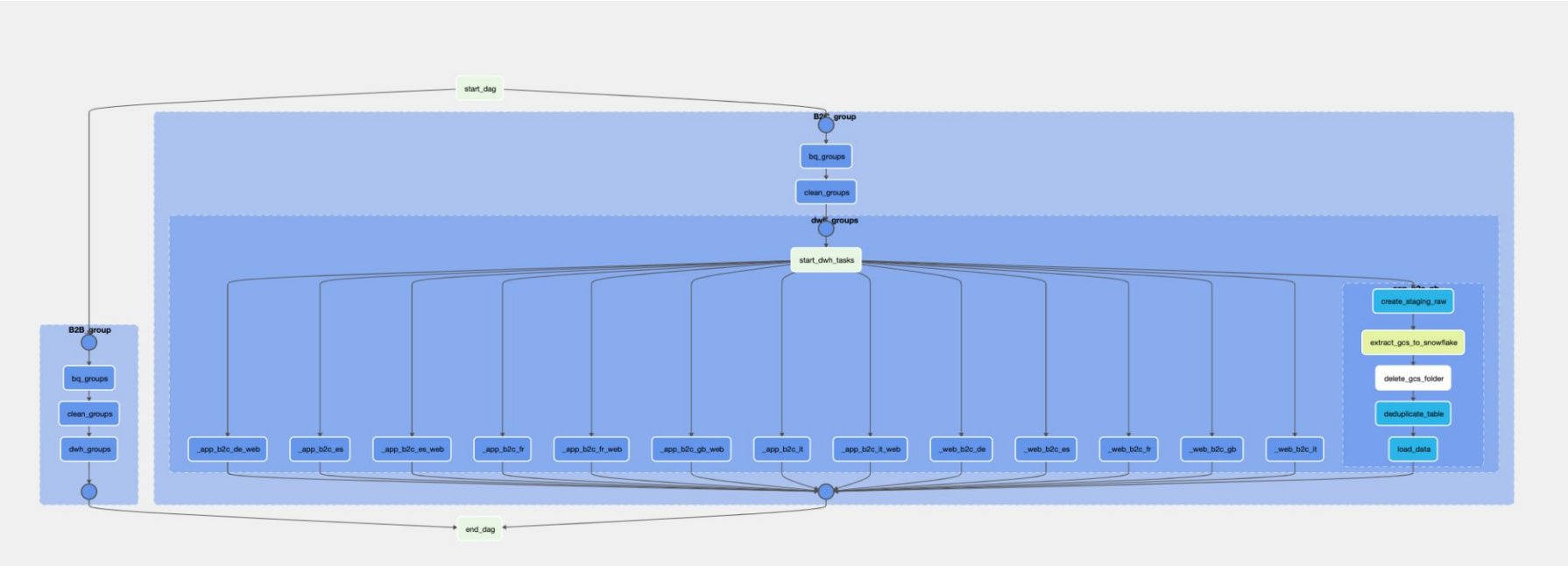
Pasamos de esto:



A esto:


start_dag
B2B group
bq_groups
clean_groups
dwh_groups
B2C group
bq_groups
clean_groups
dwh_groups
end_dag

Zoom in


start_dag
B2B_group
bq_groups
clean_groups
dwh_groups
B2C_group
bq_groups
clean_groups
dwh_groups
start_dwh_tasks
_app_b2c_de_web
_app_b2c_es
_app_b2c_es_web
_app_b2c_fr
_app_b2c_fr_web
_app_b2c_gb_web
_app_b2c_it
_app_b2c_it_web
_web_b2c_de
_web_b2c_es
_web_b2c_fr
_web_b2c_gb
_web_b2c_it
create_staging_raw
extract_gcs_to_snowflake
delete_gcs_folder
deduplicate_table
load_data
end_dag

Conclusión

- “Divide y vencerás”

- En muchos casos tomarse el tiempo para mejorar los pipelines legacy puede ahorrarte tiempo, dinero y dolores de cabeza.

El Grumpy Data Pipeline

ya no fallaba casi nunca

y cumplia con sus SLA’s

Y el data team vivió feliz

para s~~iempre~~ un rato más...

Fin.

Gracias por su atención

@ValeryBriz