

REVISIÓN DE UN FLUJO / RESUMEN

Revisión de fiabilidad del flujo de pedidos

Un flujo en BigQuery y Composer/Airflow

Esta muestra ilustra los hallazgos y decisiones de una revisión enfocada. La empresa, el flujo y los registros son ficticios. Las pruebas locales proceden del ensayo en Python adjunto; no son resultados de un cliente, una medición en la nube ni una promesa de ahorro.

DECISIÓN RECOMENDADA

Corregir los reintentos y la extracción antes de confiar en esta fuente de informes. La solución local trata correctamente las versiones suministradas. El cambio en producción debe esperar a que el responsable confirme las reglas del origen y la solución supere las pruebas en el entorno real de BigQuery y Composer.

El problema del informe

Una tabla diaria de pedidos alimenta un informe operativo. La tarea incremental filtra por fecha del pedido y añade el resultado al destino. Repetir un lote crea un duplicado; se pierden los cambios de pedidos antiguos. Que la tarea termine correctamente no revela estos errores.

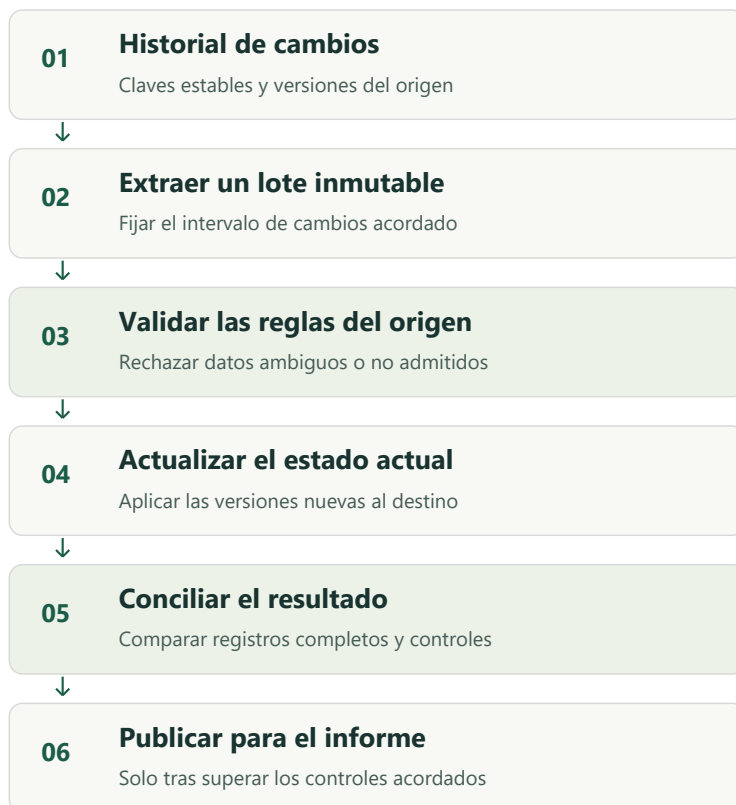
RESULTADO LOCAL	LÓGICA INSEGURA	REFERENCIA Y CANDIDATO
Filas en la tabla de destino	5	4
Claves de pedido distintas	4	4
Importe de control	1.300,00 GBP	1.020,00 GBP
Correcciones tardías de A y B	Ausentes	Presentes

Los importes son controles de conciliación ficticios. Su diferencia no mide ingresos perdidos, reducción de costes ni impacto comercial.

02 / ALCANCE DE LA REVISIÓN

Alcance y flujo

La revisión sigue un flujo de pedidos desde los cambios en origen hasta su estado actual. El diagrama muestra el recorrido previsto; el ensayo reproduce su comportamiento localmente.



Alcance del ejemplo

Un flujo diario, dos lotes de cambios, una tabla destino y cuatro claves de pedido. La instantánea inicial y los lotes contienen seis versiones del origen.

Supuestos del origen

Claves estables, fechas de actualización en UTC y versiones ordenadas. Cada versión contiene un registro completo. Esta solución **no admite borrados**.

Referencia independiente

El estado esperado se define de forma independiente en **expected.json**. No lo calcula la función de actualización propuesta.

Composer/Airflow orquestaría estos pasos en un entorno real. El ejemplo no despliega Airflow ni ejecuta SQL en BigQuery.

Evidencias necesarias del entorno real

Empezar con código anonimizado, definiciones de tablas, reglas del origen y metadatos de las ejecuciones seleccionadas. Relacionar las tareas con sus trabajos de BigQuery. La facturación, región, necesidad de actualización y controles del negocio determinan si el cambio merece la pena.

El cliente puede facilitar exportaciones autorizadas. Si hace falta acceso directo, acordar su finalidad, alcance, caducidad y retirada. La consulta inicial no requiere credenciales ni registros confidenciales. Una revisión no autoriza escrituras en producción.

Quedan fuera la plataforma completa, el resto de DAG, la herramienta de informes y la seguridad de la organización de Google Cloud.

03 / QUÉ CORREGIR PRIMERO

Hallazgos y acciones recomendadas

F01

Un reintento duplica un pedido

PRIORIDAD
ALTA

E03 contiene dos copias del pedido C al añadir dos veces el mismo lote. T01 detecta el resultado incorrecto. Un reintento no debe crear un registro de negocio nuevo por haber perdido la confirmación de la ejecución anterior.

Acción recomendada: Actualizar el estado por order_id y source_version tras validar el lote. T03, T07 y T15 comprueban repetición, duplicados idénticos y recuperación simulada tras perder la confirmación. Falta probar SQL, orquestación y concurrencia reales.

EVIDENCIA E03 / PRUEBAS T01, T03, T07, T15

F02

El filtro por fecha omite correcciones tardías

PRIORIDAD
ALTA

En E01, A y B cambian después de la fecha original del pedido. Filtrar solo por esa fecha omite las versiones nuevas. E02 define los importes corregidos esperados y T05 confirma que ambos aparecen en la solución propuesta.

Acción recomendada: Elegir un cursor fiable según las reglas del origen: fecha de actualización o posición de captura de cambios. Usar intervalos explícitos y lotes inmutables. Acordar retrasos, solapamiento y conciliación; una ventana retrospectiva sin evidencias no garantiza completitud.

EVIDENCIAS E01, E02 / PRUEBA T05

F03

Una tarea correcta no garantiza datos correctos

PRIORIDAD
ALTA

El flujo puede terminar con claves duplicadas o importes obsoletos. El ejemplo compara filas completas, claves e importes; un total agregado sin cambios no puede ocultar por sí solo un registro incorrecto.

Acción recomendada: Definir periodo de referencia, responsables y aceptación antes del cambio. Mantener disponible la salida publicada anterior hasta superar las pruebas. Las discrepancias y las reglas desconocidas del origen bloquean el despliegue.

EVIDENCIAS E02, E04, E05 / PRUEBAS T02, T18

04 / PRUEBAS Y LÍMITES

Resultados de validación

20 / 20

PRUEBAS LOCALES SUPERADAS

Comprobaciones locales en Python
Validación en la nube pendiente

La salida esperada se define independientemente de la función de actualización. Los resultados comprueban el comportamiento acordado para estos casos; no demuestran validez para cualquier origen o carga de producción.

PRUEBAS	RESULTADO	COMPORTAMIENTO COMPROBADO
T01 a T02	OK	Detectar el resultado inseguro y coincidir con la referencia independiente.
T03 a T07	OK	Repetición, versiones antiguas, correcciones tardías, orden inverso y duplicados idénticos.
T08 a T12	OK	Bloquear versiones contradictorias, claves nulas, fechas inválidas, borrados y cambios de esquema.
T13 a T16	OK	Lotes vacíos, fallo antes de publicar, confirmación perdida y reversión local.
T17 a T20	OK	Límites de intervalo, conciliación exacta, validación del lote completo y orden temporal.

Plan de aceptación en producción

- 01 Confirmar las reglas del origen y una referencia independiente representativa, incluidos datos tardíos y política de borrados.
- 02 Ejecutar el SQL adaptado en un dataset aislado del cliente. Comparar claves y valores, repetir lotes y probar entradas inválidas con límite de coste.
- 03 Probar la versión instalada de Composer/Airflow, intervalos fijos, lotes inmutables y confirmaciones perdidas. Usar un solo escritor o validar la concurrencia acordada.
- 04 Ensayar la restauración conjunta de datos, punto de control y código estable. Identificar aprobación, responsable de recuperación, publicación y periodo de observación.

Pendiente de probar: SQL en BigQuery, planificación en Composer/Airflow, extracción real, escritores concurrentes, restauración en producción, tiempos y cargos. Los SQL son plantillas de diseño. Una implementación real requiere alcance y aceptación propios.

05 / TRAZABILIDAD

Evidencias y responsables

El anexo técnico permite revisar el razonamiento local. Cada evidencia se identifica abajo; los roles proponen responsabilidades para un encargo real.

ID	ARCHIVO O REGISTRO	QUÉ DEMUESTRA
E01	fixtures.json	Instantánea ficticia inicial y dos lotes inmutables de cambios.
E02	expected.json	Estado independiente: A 110,00; B 210,00; C 300,00 y D 400,00 GBP.
E03	evidence/baseline.json	Duplicado y valores obsoletos reproducidos con la lógica insegura.
E04	evidence/candidate.json	Salida candidata contrastada con la referencia independiente.
E05	evidence/checks.json	Resultado y significado de cada una de las 20 pruebas locales.
E06	Plantillas SQL y RUNBOOK.txt	Plan de validación y recuperación en la nube, aún sin ejecutar.

Responsabilidades propuestas para un encargo real

RESPONSABLE	DECISIÓN O RESPONSABILIDAD
Responsable del origen	Confirmar claves, versiones, actualizaciones, borrados y datos tardíos.
Responsable del flujo	Validar la referencia, el entorno de pruebas y el traspaso operativo.
Responsable del despliegue	Autorizar el cambio concreto en producción, su ventana y reversión.
AgileDataWarehouse	Entregar la revisión o cambio acordado, evidencias, límites y documentación.

Ejecutar el ejemplo local

Extraer el anexo y ejecutar **python run_rehearsal.py** con Python 3.10 o posterior. Solo utiliza la biblioteca estándar. No requiere cuenta en la nube ni credenciales.

06 / LA SIGUIENTE DECISIÓN

Del informe al siguiente paso acotado

Tu equipo puede utilizar el informe sin comprometerse a un proyecto mayor. La revisión es útil cuando resuelve una incertidumbre relevante para la siguiente decisión.

01 Implementar con tu equipo

Utilizar las evidencias, la corrección propuesta y el plan de aceptación para un cambio ejecutado por vuestro equipo.

02 Solicitar una implementación acotada

Acordar la corrección, pruebas, accesos, autorización de producción, reversión y entrega mediante una propuesta independiente.

03 Aclarar primero las reglas del origen

Detener la implementación si hay dudas sobre actualizaciones o borrados. Aclararlas con el responsable del origen.

El apoyo continuado es opcional. Puede cubrir flujos, controles y mejoras dentro de la capacidad reservada. No se incluye automáticamente en una revisión ni sustituye unas guardias 24/7.

Interpretación y método

Seis versiones sintéticas no permiten estimar ahorro, mejora de tiempos, nivel de servicio ni frecuencia de incidencias. Un MERGE sin filtrar el destino puede leer muchos datos. Para recomendarlo por eficiencia hacen falta planes de consulta, cargas representativas y facturación real.

Airflow recomienda resultados repetibles entre reintentos. La atomicidad de MERGE en BigQuery no hace atómicas la extracción ni la orquestación. El entorno y las versiones reales determinan el diseño final.

[Diseño de tareas en Airflow](#) / [Referencia MERGE en BigQuery](#)

Revisemos un flujo

Describe qué se ejecuta, qué falla y qué primer resultado sería útil. Podemos empezar por correo o mediante el formulario web. Antes de trabajar acordamos alcance, accesos, precio y fechas por escrito. Los datos profesionales se facilitan en privado antes de contratar.

[Describe tu flujo en agiledwh.com](https://agiledwh.com)