The Alert Is Firing. Now What?

LLM-Assisted Triage for Data Quality DAGs

Ashir Alam
Intuit Credit Karma –
Airflow Summit 2026

About Intuit Credit Karma

- **Mission**: Champion financial progress for 140M+ members worldwide.

- **Differentiator**: Pioneered free credit scores; services remain free.

- **Offerings**: Identity and Credit monitoring; credit cards; insurance; loans ; savings & checking.



About Me



Ashir Alam

Staff Software Engineer, Data at Intuit Credit Karma

Data pipelines, data quality, self-service platforms – and now LLM tooling for oncall

Agenda

01 Previously: alerts as configuration (2025 recap)

02 The challenge: every alert still needs a human

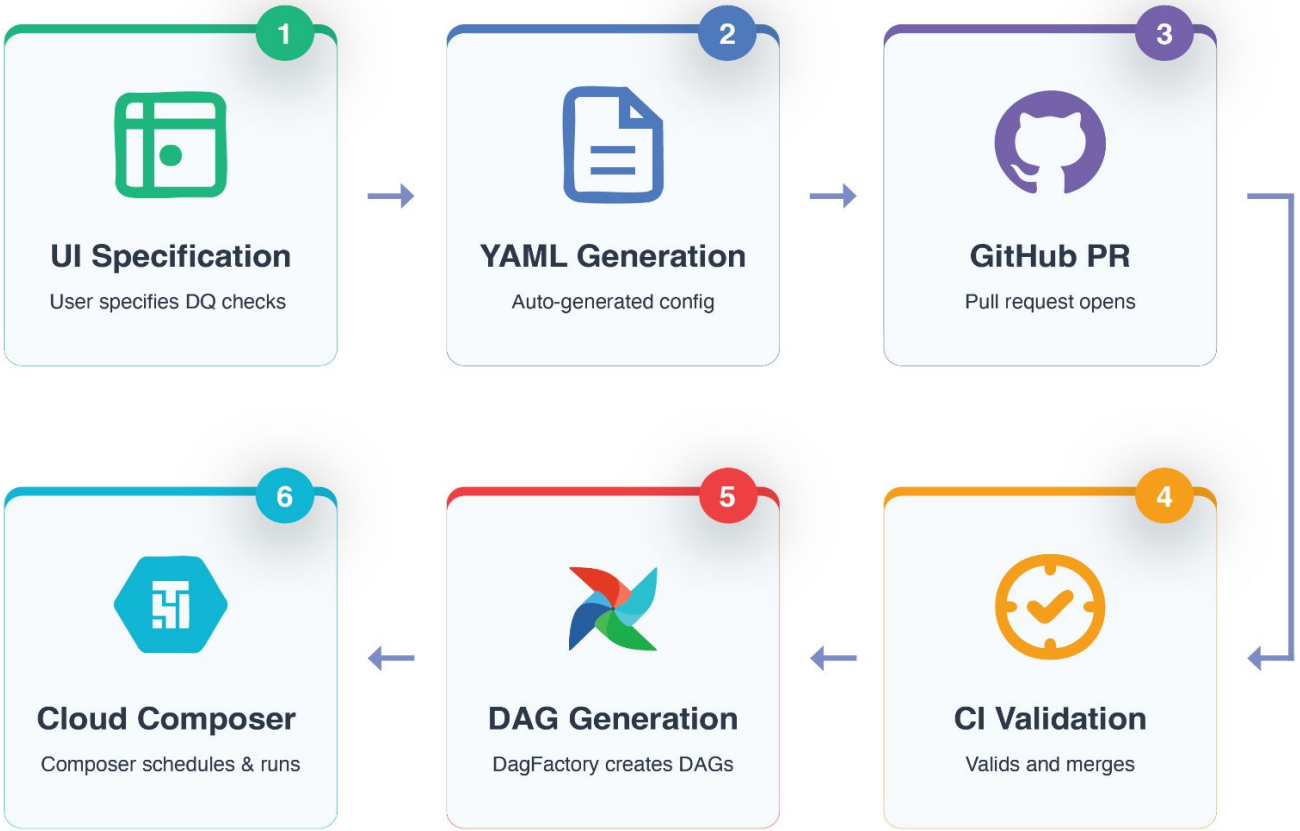
03 The solution: an LLM triage DAG

04 Demo: v1 → v2, building trust

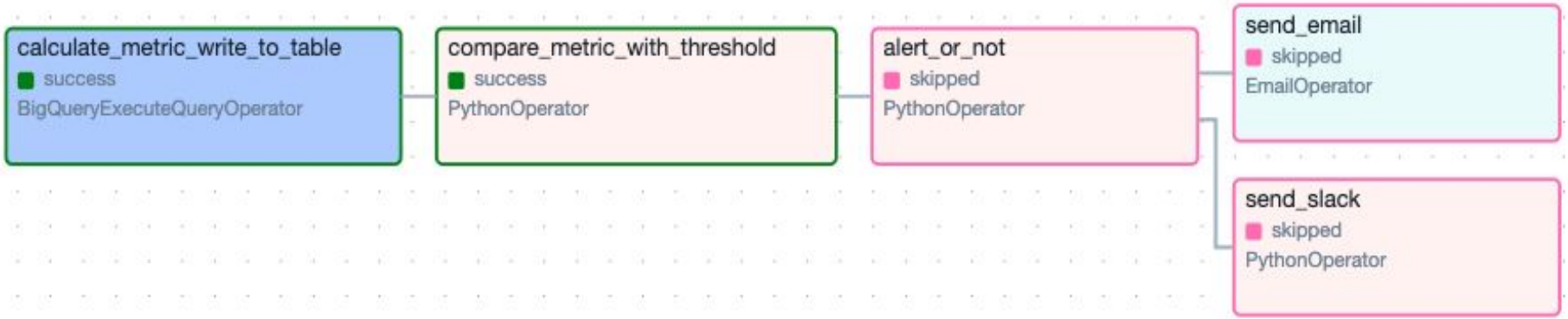
05 Impact & lessons learned

06 Q&A

2025 Recap: Alerts as Configuration



Where Part 1 Ended: The Alert Goes Out



We Automated Alert Creation, Not Judgment

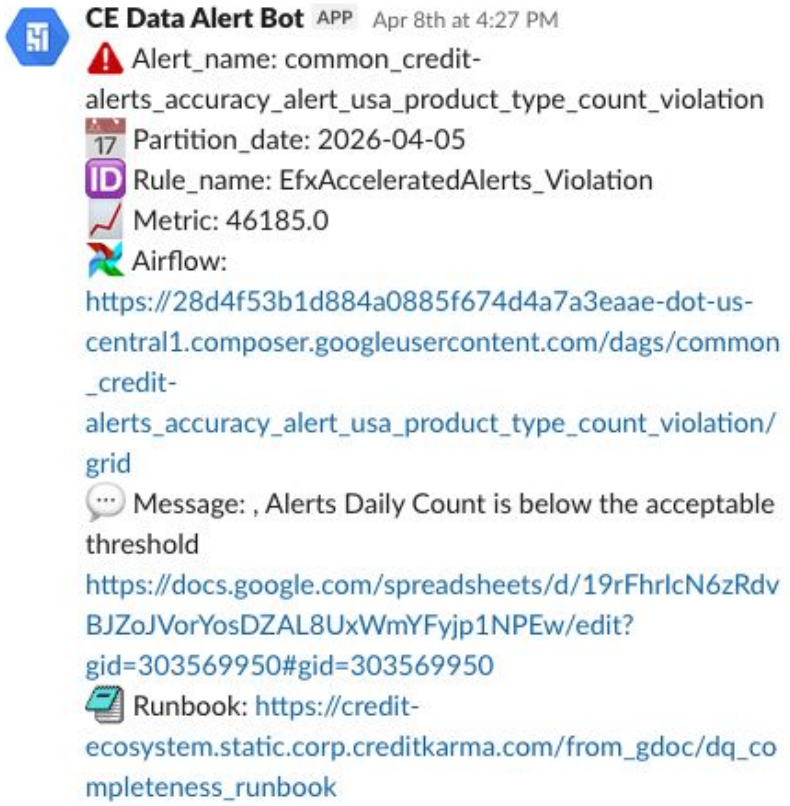
Self-service made checks cheap — alert volume grew with coverage

Every alert still lands on a human: is this real?

Median 1–2 hours of oncall time per alert

Alert fatigue: the more alerts, the less each one is trusted

A Real Morning: The Alert Fires



Is the data wrong — or is it just Sunday?

The Manual Triage Ritual

Weekend or seasonality? — compare the same weekday in prior weeks

Data late? — re-run the metric now

Fake or test users? — join violations against the fake-user table

Short file upstream? — manifest vs ingested row counts

Related alerts on the same partition?

Did the alert itself just change? — git history

Threshold stale? — history vs limit

The Insight: Split Triage Into Three Jobs

Evidence gathering is deterministic → an Airflow task per runbook question

Hypothesis testing is bounded exploration → an LLM with one query tool

Judgment stays human → the LLM drafts, oncall decides

The alert never changes – the thread under it gets smarter

A DAG That Triages Your DAGs

1. Alert fires
DQ DAG posts to Slack – unchanged

2. trigger_triage()
opt-in per alert; never breaks alerting

3. prepare
re-reads the breaching rows

4. collect × 8
parallel Airflow tasks, one per runbook question

5. investigate
LLM loop + guarded read-only BigQuery

6. publish
threaded verdict in Slack

One triage DAG, triggered by any alert DAG – pure Airflow, no new infrastructure

Eight Collectors, One Per Runbook Question

alert_definition — what does this alert measure? (YAML + SQL)

violations — the rows that fired

metric_history — 35 days, same-weekday patterns

replay_now — re-run the SQL now; no longer breaching → data was late

related_alerts — what else fired on this partition

recent_changes — did the alert itself just change? (git)

source_volume — upstream row counts, 21 days

file_ingestion — manifest vs ingested rows, per file

A failed collector becomes evidence marked UNAVAILABLE — never an abort

The Agent Loop: Bounded Autonomy

Input: evidence items E1…En from the collectors

Two tools only: run_bq_query and submit_verdict

Queries are guarded: read-only, partition filter required, byte-capped

Budgets: ≤6 queries, ≤10 turns, ≤360 seconds — then the verdict is forced

Every LLM query becomes new evidence — the trail is auditable

The runbook lives in the prompt: weekend rules, fake users, test the counterfactual

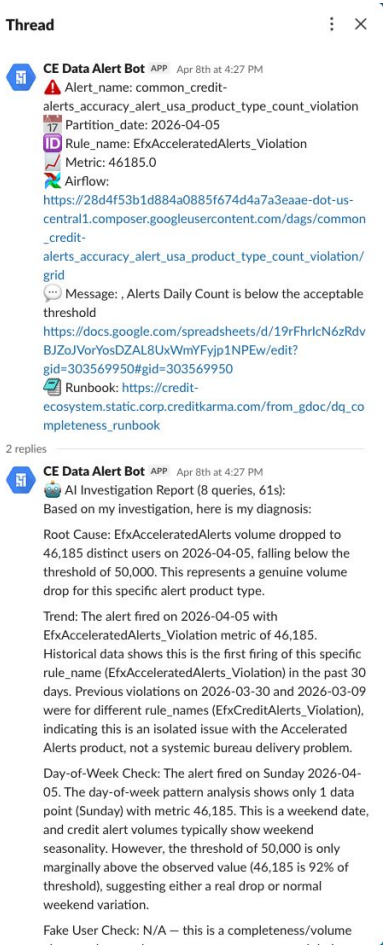
The Output Is a Form, Not an Essay

```
{
  "verdict":    "TRUE_POSITIVE | FALSE_POSITIVE | INCONCLUSIVE",
  "category":   "data_delay | weekend_seasonality | fake_users | …",
  "confidence": "high | medium | low",
  "headline":   "one sentence, ≤160 chars",
  "evidence":   [ { "evidence_id": "E3",
                    "quote": "verbatim value from that evidence",
                    "supports": "what this shows" } ],
  "ruled_out":    [ … ],    "not_checked": [ … ],
  "recommended_actions": [ … ],
  "engineer_action": "none | rerun_after_data_lands | tune_threshold | …"
}
```

Every claim must cite an evidence id and a verbatim quote

Demo v1: The First AI Diagnosis

Real alert, real diagnosis — 8 verification queries, 61 seconds, threaded under the alert



Thread

CE Data Alert Bot APP Apr 8th at 4:27 PM

Alert_name: common_credit-alerts_accuracy_alert_usa_product_type_count_violation
Partition_date: 2026-04-05
Rule_name: EfxAcceleratedAlerts_Violation
Metric: 46185.0
Airflow: https://28d4f53b1d884a0885f674d4a7a3eaae-dot-us-central1.composer.googleusercontent.com/dags/common_credit-alerts_accuracy_alert_usa_product_type_count_violation/grid
Message: , Alerts Daily Count is below the acceptable threshold
https://docs.google.com/spreadsheets/d/19rFhrIcN6zRdvBJZoJVorYosDZAL8UxWmYFyjp1NPEw/edit?gid=303569950#gid=303569950
Runbook: https://credit-ecosystem.static.corp.creditkarma.com/from_gdoc/dq_completeness_runbook

2 replies

CE Data Alert Bot APP Apr 8th at 4:27 PM

AI Investigation Report (8 queries, 61s):
Based on my investigation, here is my diagnosis:

Root Cause: EfxAcceleratedAlerts volume dropped to 46,185 distinct users on 2026-04-05, falling below the threshold of 50,000. This represents a genuine volume drop for this specific alert product type.

Trend: The alert fired on 2026-04-05 with EfxAcceleratedAlerts_Violation metric of 46,185. Historical data shows this is the first firing of this specific rule_name (EfxAcceleratedAlerts_Violation) in the past 30 days. Previous violations on 2026-03-30 and 2026-03-09 were for different rule_names (EfxCreditAlerts_Violation), indicating this is an isolated issue with the Accelerated Alerts product, not a systemic bureau delivery problem.

Day-of-Week Check: The alert fired on Sunday 2026-04-05. The day-of-week pattern analysis shows only 1 data point (Sunday) with metric 46,185. This is a weekend date, and credit alert volumes typically show weekend seasonality. However, the threshold of 50,000 is only marginally above the observed value (46,185 is 92% of threshold), suggesting either a real drop or normal weekend variation.

Fake User Check: N/A — this is a completeness/volume alert on alert product counts, not an accuracy violation on individual record values. The metric counts distinct users receiving each alert product type.

Correlated Alerts: None found — no other accuracy alerts for the common partner fired in the last 24 hours.

Verdict: INTERMITTENT — The alert fired on a weekend with a volume only 8% below threshold. This could be normal weekend seasonality in alert volumes, or a minor delivery fluctuation. The isolated nature (first EfxAcceleratedAlerts_Violation in 30 days) and marginal threshold breach suggest this may be a threshold tuning issue rather than a true data quality failure.

Recommended Action:

1. Per the runbook section \"Low Alert Volume from bureau\", check New Relic dashboards for Credit Alerts Pipeline services (Webhooks, EFGA gateway, CAS) to confirm no CK-side ingestion failures occurred on 2026-04-05.

2. If CK pipeline is healthy, use the runbook query to collect Equifax Accelerated Alerts stats: SELECT partition_date, count(distinct bureauAlertId) FROM prod-ck-abl-data-53.ck_bureau_ingestion_etl_0_2.credit_alert_usa WHERE bureau='Equifax' AND alertProduct='EfxAcceleratedAlerts' AND partition_date >= '2026-03-20' GROUP BY 1 ORDER BY 1 DESC to determine if this is a bureau-side volume drop or CK processing issue.

3. Monitor the next weekday (2026-04-07) volume to confirm whether this is weekend seasonality. If weekday volumes return above 50,000, recommend threshold adjustment to account for weekend patterns.

Confident Prose Is Not Evidence

- Numbers with no way to check where they came from
- Root causes stated, not shown — hallucination reads the same as truth
- 14 runs on one alert produced 3 different verdicts
- "Weekends see 50–70% lower volume" — started with zero supporting evidence
- The costliest sentence the bot can say is "ignore this" — when it is wrong

Demo v2: Every Claim Carries Its Receipt

Same alert class — now every number cites the query that produced it



Thread

CE Data Alert Bot APP 37 minutes ago

[TRIAGE TEST - replayed alert]
Alert_name: common_credit-alerts_accuracy_alert_usa_product_type_count_violation
Partition_date: 2026-11-02
Rule_name: TuCreditAlerts_Violation
Metric: 116851.0
Airflow: https://28d4f53b1d884a0885f674d4a7a3eaae-dot-us-central1.composer.googleusercontent.com/dags/common_credit-alerts_accuracy_alert_usa_product_type_count_violation/grid
Message: Alerts Daily Count is below the acceptable threshold
https://docs.google.com/spreadsheets/d/19rFhrlcN6zRdvBJZoJVorYosDZAL8UxWmYFyjp1NPEw/edit?gid=303569950#gid=303569950
Runbook: https://credit-ecosystem.static.corp.creditkarma.com/from_gdoc/dq_completeness_runbook

1 reply

CE Data Alert Bot APP 37 minutes ago

AI triage: 🔴 **TRUE POSITIVE** (real data issue) · confidence **high**

> TuCreditAlerts fell to 116,851 distinct users on Nov 2, below the 300,000 minimum, and remains low after replay.

Why: The count remains 116,851 after replay, so late arrival did not resolve it. Excluding fake users leaves 116,851 real users and zero fake users. The count was only 12.4% of the prior Monday's 940,974, ruling out ordinary Monday seasonality. Total source volume was 59.9% of the prior Monday, but available evidence does not establish upstream or CK-side ownership.

Evidence
- The breach persisted when replayed on Nov 4. `metric_now:116851.0,still_breaches:true` *(replay)*
- The low count affects real users and is not caused by fake accounts. `real_tu_users:116851,fake_tu_users:0` *(query)*
- TuCreditAlerts users fell to 12.4% of the immediately prior Monday. `nov02_tu_users:116851,oct26_tu_users:940974,tu_ratio:0.124` *(query)*
- Overall source volume fell to 59.9% of the prior Monday. `nov02_source_rows:3729408,oct26_source_rows:6224858,source_ratio:0.599` *(query)*

Ruled out
- Late-arriving data: `metric_now:116851.0,still_breaches:true` *(replay)*
- Fake users caused the breach: `real_tu_users:116851,fake_tu_users:0` *(query)*
- Normal Monday seasonality: `nov02_tu_users:116851,oct26_tu_users:940974,tu_ratio:0.124` *(query)*

Suggested: Investigate manually
1. Investigate the Nov 2 TuCreditAlerts delivery and pipeline stages for missing or truncated inputs.
2. Compare manifests, ingestion totals, and target counts against Oct 26 before escalating ownership.
3. Assess downstream impact to users missing expected TuCreditAlerts records.

Not checked
- Per-file or upstream delivery evidence was unavailable, so ownership of the shortfall is unknown.
- AlertProduct-level pipeline stages before the target table were not inspected.

8/8 checks · every citation reviewed (3 fixed on re-review)
2 verification queries · 57.2s · gpt-5.6-sol (sev-2) · triage `fb98373814ed`
React ✅ if this verdict is right or ❌ if it is wrong.

The Trust Pipeline: Verdicts Are Audited

1. Deterministic check — every quote must exist in its cited evidence
2. LLM judge — does the quote prove the claim? verdict and cause judged too
3. One repair round — problems go back to the investigator
4. Still unsupported → dropped; verdict downgraded toward INCONCLUSIVE

The Slack reply lists what was removed — the bot shows its corrections

Asymmetric Errors Get Asymmetric Compute

A wrong TRUE POSITIVE costs an hour; a wrong FALSE POSITIVE hides an incident

Default: fast model (GPT-5.4) via the Intuit AI Gateway

FALSE POSITIVE or INCONCLUSIVE → re-run on the stronger model (GPT-5.6)

Sev-2 alerts skip straight to the stronger model; disagreements shown in-thread

Real case: the fast model said FALSE POSITIVE 3× — the strong model overruled it twice

Rollout: Shadow First, Opt-In Always

Shadow mode — verdicts post to a test channel; the real alert is untouched

Per-alert opt-in via one Airflow Variable

Triage can never break alerting: the trigger never raises, failures never page

✅ / ❌ reactions on every verdict → a labeled dataset for free

That dataset gives us override rate, false-negative rate, and an eval set

Impact So Far

1–2 hours → under 10 minutes
median time to a triage decision

45 s median
from alert firing to posted verdict – it's waiting before oncall even looks

100%
verdicts accepted so far (✅, none overridden)

~20k tokens / triage
projected under $100 / month for all alerts

Oncall starts from evidence, not from scratch

Lessons Learned

- Hallucinated causes → citation validation beats better prompting
- Context bloat → budget every evidence item; excerpts, not dumps
- Cost → cheap model first; escalate only the risky verdicts
- Corporate reality – proxies, gateways, auth – took half the engineering time
- Once, the judge itself was wrong – the verifier needs verifying
- The triage path must never page, never retry-duplicate, never block the alert

What's Next: From Diagnosis to Action

Act with approval — one-click rerun / snooze / backfill from the thread

Eval harness — replay labeled alerts on every prompt or model change

Cross-alert correlation — one root cause, many alerts, one thread

Runbook knowledge base — every resolved thread feeds the next triage

Our ✅/❌ labels feed agent evals — the very next session's topic

Thanks! Questions?

Ashir Alam

ashir_alam@intuit.com

linkedin.com/in/ashir-alam

Part 1 (2025): Dynamic DAGs & Data Quality using DagFactory – airflowsummit.org

Backup: How We Got Here

1. A runbook doc — humans follow the steps
2. An interactive AI skill — engineer drives, assistant helps
3. v1: inline ReAct agent (Claude Haiku) — free-text report, no citations
4. v2: a triage DAG (GPT-5.4 / 5.6) — collectors, cited verdict, judge, repair

The verification design mattered more than which model we used

Backup: Real Runs, Real Saves

Fast model said FALSE POSITIVE 3× on one alert – the strong model overruled it twice

A verdict blamed the bureau while its own evidence showed CK wrote MORE rows than were delivered

The judge dropped "ruled out data delay" – the quote cited didn't prove it

One repair round fixed 2 weak citations in a single run

And once the judge itself was wrong – the verifier needs verifying too

Backup: Guardrails & Budgets

SQL guard: single SELECT only, keyword blocklist, airlock project only, dry-run byte check

Byte caps per query class (2–20 GiB); replay capped at one scheduled run's cost

Agent: ≤6 queries, ≤10 turns, ≤360 s; repair adds 2 queries / 120 s

DAG: 20-min timeout, 4 concurrent triages, publish never retries (no duplicate threads)

Isolation: opt-in Variable, shadow channel, failures never page