The Path to Trusted Data: A Battle-Tested Observability Strategy

A revised and de-risked plan **for achieving ROI**, faster incident resolution, and enterprise-wide data trust.

We Are Optimizing for the Right Outcomes, in the Right Order

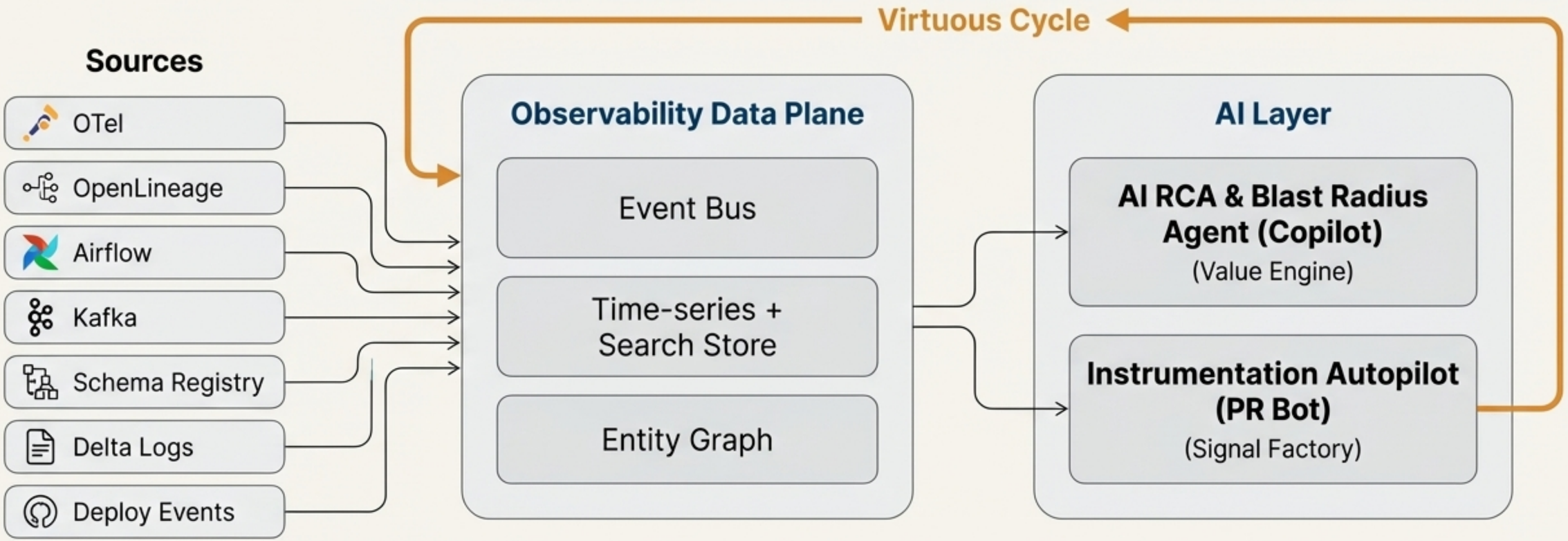
Our strategy is built on a mature understanding of value delivery. We will not boil the ocean. Instead, we will focus on a clear, prioritized sequence of business outcomes:

"This priority ordering shows maturity. Starting with demonstrable value rather than comprehensive coverage is the correct approach."

— *Critical Review Assessment*

Our Architecture Combines an AI “Value Engine” with a “Signal Factory”

The core of our plan is a powerful combination (B+A): an AI Copilot that delivers immediate value from existing signals, paired with an Instrumentation Autopilot that scales the data the Copilot needs to become even smarter. This creates a virtuous cycle of adoption and improvement.

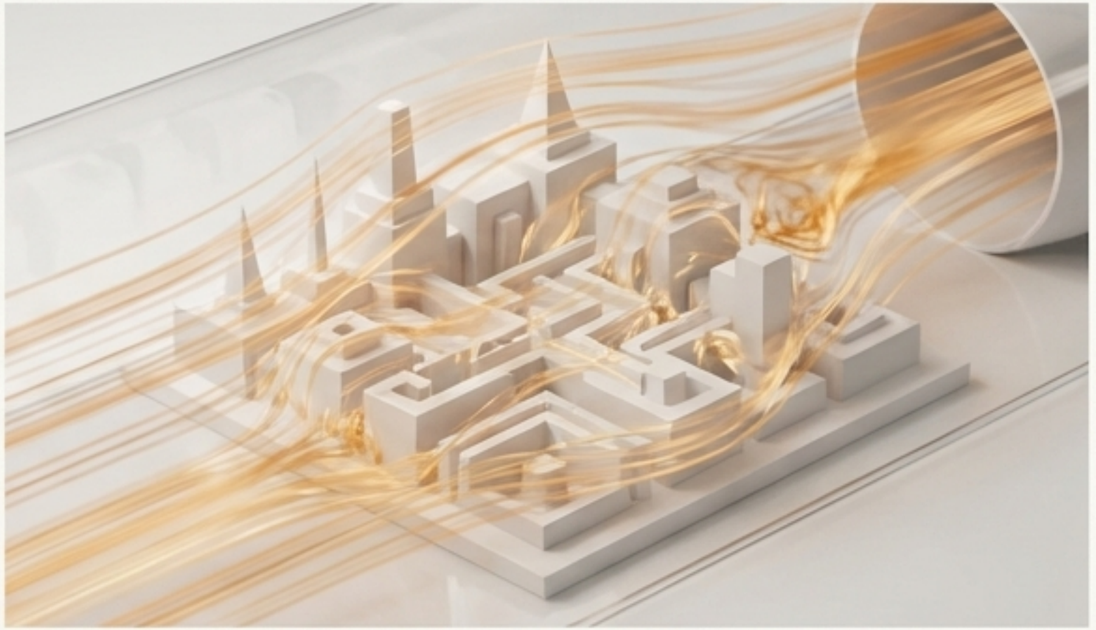


A Good Plan Survives First Contact with Reality

The Plan

The initial architecture is sound. But the best programs succeed not because of technical sophistication alone...

The Reality Check



...but because they relentlessly prepare for operational reality.

We conducted a candid, expert-led critical review to uncover the hidden assumptions and risks in our plan *before* execution.

What follows is not a new strategy, but a battle-hardened one—designed to survive contact with real teams, real incidents, and real organizational dynamics.

We Identified Four Thematic Gaps to Address

Our review surfaced challenges across technology, automation, process, and planning. Addressing these is the key to building a robust, trusted system.

Technical Foundation

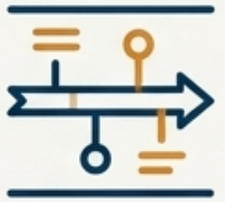
- **Risk:** The "Single Correlation ID" assumption is fragile across async boundaries (Kafka, Airflow, Spark).
- **Risk:** The plan for streaming Data Quality (DQ) is underspecified and treated as optional.

AI & Automation

- **Risk:** AI Copilot accuracy is undefined. If it's wrong 30% of the time, engineers will ignore it.
- **Risk:** Autopilot PRs could create review fatigue or break production, destroying trust.

People & Process

- **Risk:** The plan lacks organizational change management. Without it, adoption will stall.
- **Risk:** The Data Contract ownership model is vague, risking toothless or blocked agreements.



Execution Plan

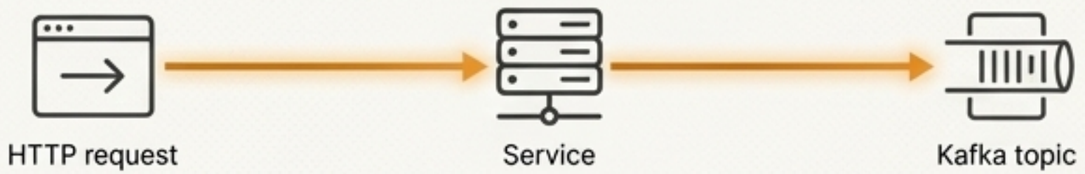
- **Risk:** The 30/60/90 day plan is aggressive and risks creating technical debt.
- **Risk:** The plan is missing explicit failure modes, circuit breakers, and rollback procedures.

Solving Correlation: A Hybrid Model for an Asynchronous World

Runtime trace propagation breaks at every async boundary. Relying on it alone is a recipe for failure. Our solution is a pragmatic, three-layer hybrid model that ensures the Copilot can always connect the dots.

Layer 1: Runtime Correlation (Where it works)

Scope: Sync calls (HTTP → Service → Kafka produce).
Mechanism: OTel + W3C trace context.



Layer 2: Batch Correlation (Explicit Handoff)

Scope: Controlled async (Airflow → Spark).
Mechanism: Pass `dag_run_id` as a Spark parameter.

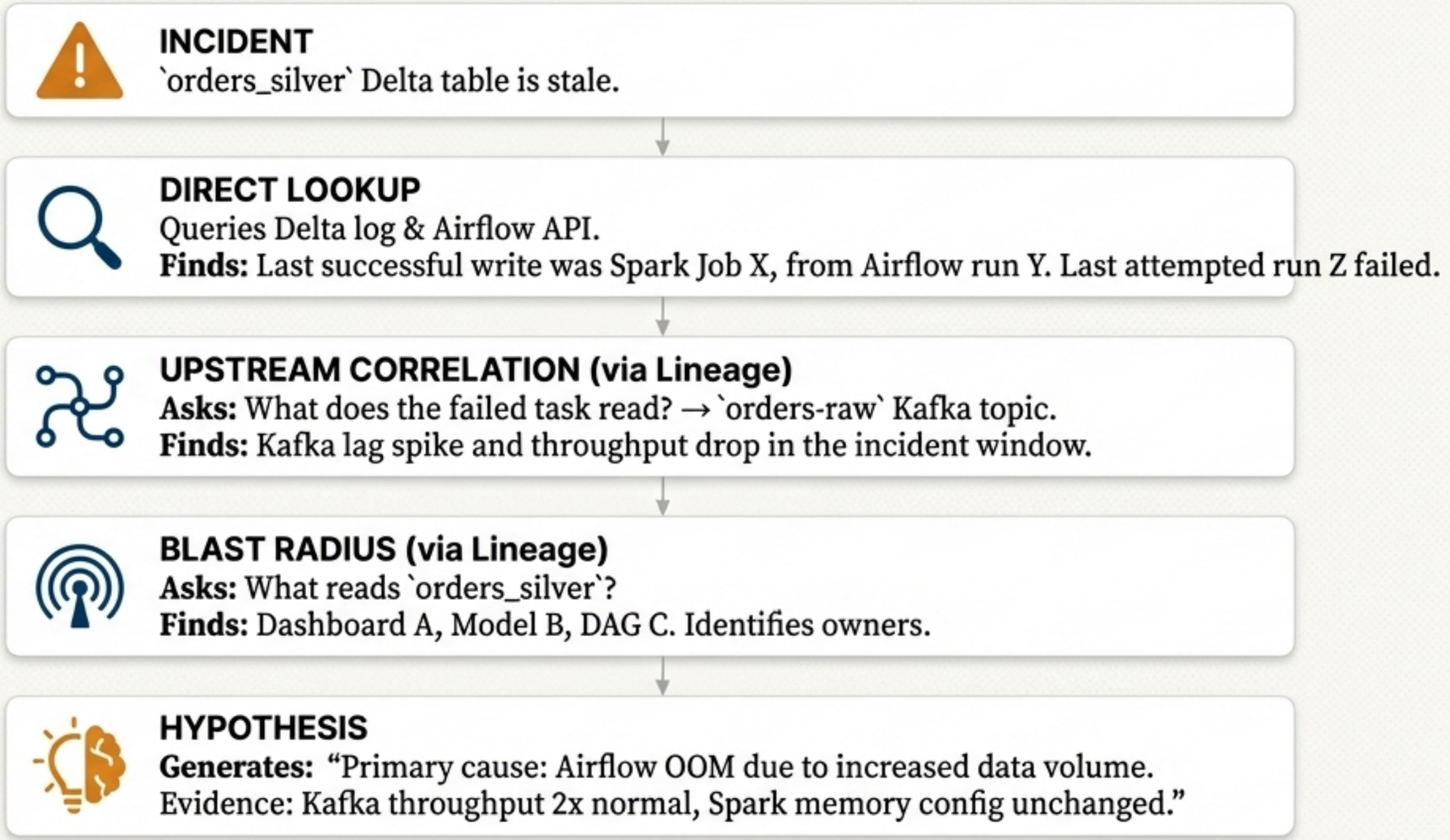
Layer 3: Lineage-Based Correlation (The Fallback)

Scope: State boundaries (Delta tables, S3 files).
Mechanism: Entity graph edges + time-window overlap.

> "You cannot propagate a correlation_id through a Delta table. The table is a *state boundary*. This is where runtime correlation ends and lineage takes over."

How the Copilot Reasons: From Signal to Hypothesis

The RCA Copilot doesn't guess. It follows a repeatable, evidence-first algorithm, using the entity graph and time-window analysis to connect disparate signals into a coherent narrative.



Notice: No single `correlation_id` connects the whole chain. The Copilot uses asset-level lineage, time-window overlap, and causal reasoning.

The Future is Precise: The Power of Element-Level Lineage

To go from “which pipeline broke?” to “which transformation introduced the nulls?”, we need to move beyond asset-level lineage to the element (column) level. This unlocks a new class of diagnostic and preventative capabilities.

Question	Asset-Level Answer (Today)	Element-Level Answer (Our Goal)
Why is `revenue` null in the dashboard?	“Something in the orders pipeline.”	“Service B doesn’t populate order_total when currency is null; trace to upstream currency service.”
What breaks if I remove field `X`?	“Downstream jobs might fail.”	“Exactly these 7 fields in 4 downstream tables, used by 2 dashboards.”
Where did this PII field `Y` propagate to?	“To these 5 tables.”	“To these 12 specific columns, including 3 that are in non-PII certified systems.”

Our architecture is designed to support this from day one, with a phased implementation starting with our richest data source: Spark query plans.

Building Trust in AI & Automation Through Deliberate Guardrails

Automation without trust is just noise. We are implementing specific programs to ensure our AI Copilot is accurate and our Autopilot is a welcome contributor, not a source of fatigue.

For the AI RCA Copilot: The Accuracy Program

- **Ground Truth First:** We will define ground truth for 50 past incidents *before* going live.
- **Go/No-Go Threshold:** A hard accuracy target (e.g., top-3 hypothesis is correct ≥70% of the time) must be met before rollout.
- **Continuous Feedback Loop:** A lightweight feedback mechanism (Slack reactions, Jira button) will feed a weekly accuracy dashboard to drive improvement.

For the Instrumentation Autopilot: A Safe Rollout

- **Opt-In First:** Start with volunteer repos from our pilot domains, not mass automation.
- **Human Champions:** Require a named "observability champion" in each team to own PR acceptance and review.
- **Gated Rollout:** We will measure PR acceptance rate and time-to-merge. If these metrics degrade, we slow down.
- **Confidence Scoring:** Every generated PR will have a confidence score based on how well the repo matches known patterns.

Observability is a Practice, Not Just a Platform

Tools are only effective if they are integrated into daily workflows and supported by a culture that values data reliability. A dedicated Change Enablement workstream will ensure adoption is deep and sustainable.

Identify Champions: Proactively recruit 2-3 "Observability Champions" per domain *before* the pilot begins. They will be our partners and advocates.

Create Incentives: A simple certification or badge for teams that achieve "observability-ready" status. Make it a point of pride.

Ensure Visibility: Make observability metrics (freshness, quality, incidents) a non-punitive part of regular team health reviews.

Secure Sponsorship: We have secured explicit executive sponsorship with a monthly check-in cadence to absorb political pushback and ensure alignment.

Why might adoption stall at 30%? → Teams don't use the tools.
> *Why?* → Not integrated into their workflow.
>> *Why?* → Perceived as extra work.
>>> *Why?* → No training, no incentive, no accountability.
>>>> *Why?* → **No change management plan.** (This is what we are fixing.)

A Realistic 90-Day Plan Focused on Delivering Value Incrementally

Our original 90-day plan was ambitious. We've revised it to de-risk the initial delivery, focusing on establishing a solid foundation in one pilot domain first before expanding. This ensures we deliver high-quality capabilities, not rushed features with technical debt.

Days 0-30: Prove Value in a Single Domain

- Finalize Event Schemas & Canonical IDs
- RCA Copilot MVP (scoped to *one* pilot domain)
- Autopilot ready for one repo type (e.g., Kafka service)
- Establish Copilot Accuracy Baseline

Days 31-60: Expand and Harden

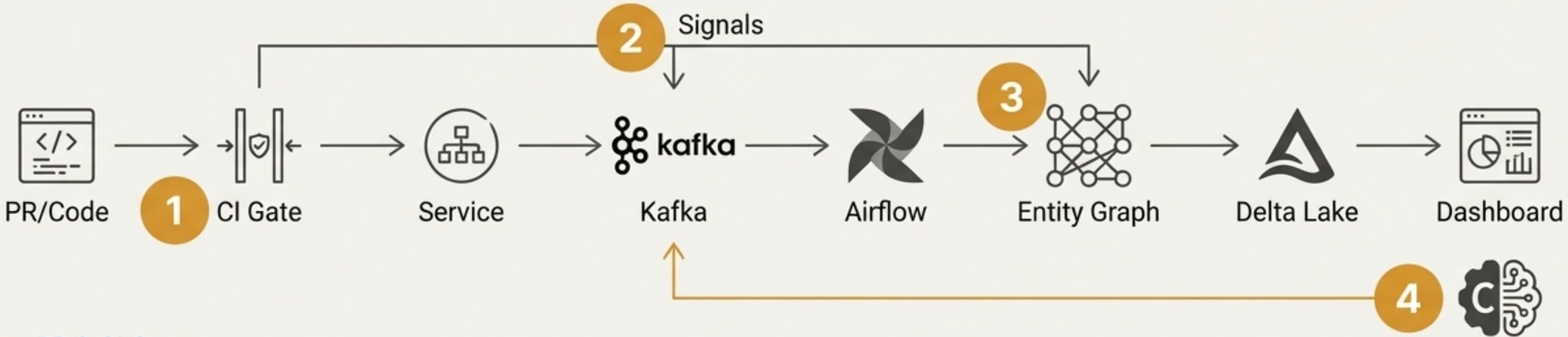
- Onboard second pilot domain
- Roll out Autopilot for Spark/Dask repos
- CI/Migration Gate (v1) designed and tested
- Copilot Blast Radius computation goes live

Days 61-90: Scale Coverage and Capabilities

- Expand Autopilot to more repo types
- Full CI/Migration Gate rollout
- Copilot adds "Similar Incident Retrieval"
- Draft initial SLOs for pilot domains

Here’s How It All Works: The “Orders” Pipeline, End-to-End

Let’s trace a real incident. At 8:00 AM, the executive dashboard is stale. Here’s how our system diagnoses the root cause in minutes, not hours.



1. **PR & CI Gate**
   A developer submits a PR with a non-backward compatible schema change for the `orders_enriched` topic. **The synchronous CI check fails the build instantly**, preventing the incident. *(This is prevention)*.
2. **Runtime Signals**
   Let's assume the change gets through. A `deploy event` is ingested. The `schema_change_event` is logged. Kafka metrics show consumer lag spiking. Airflow metadata shows the
3. **Entity Graph**
   The graph connects all assets: `orders-enricher` service → `orders_enriched` topic → `orders_daily_agg` DAG → `orders_gold_daily` Delta table → `Exec Dashboard`.

4. **RCA Copilot in Action**
   - **Input**: Freshness breach alert on `orders_gold_daily`.
   - **Blast Radius**: “Impacts: Exec Orders Dashboard, finance_daily_snapshot.”
   - **Hypothesis (High Confidence)**: “Schema change introduced a compatibility issue.”
   - **Evidence**:
     - Schema version 42 published at 1:10 AM.
     - DLQ rate for consumer increased 0 → 35/min.
     - Consumer logs show deserialization failures.
     - Consumer lag spiked at 1:15 AM.

Formalizing Trust at Scale with Data Contracts

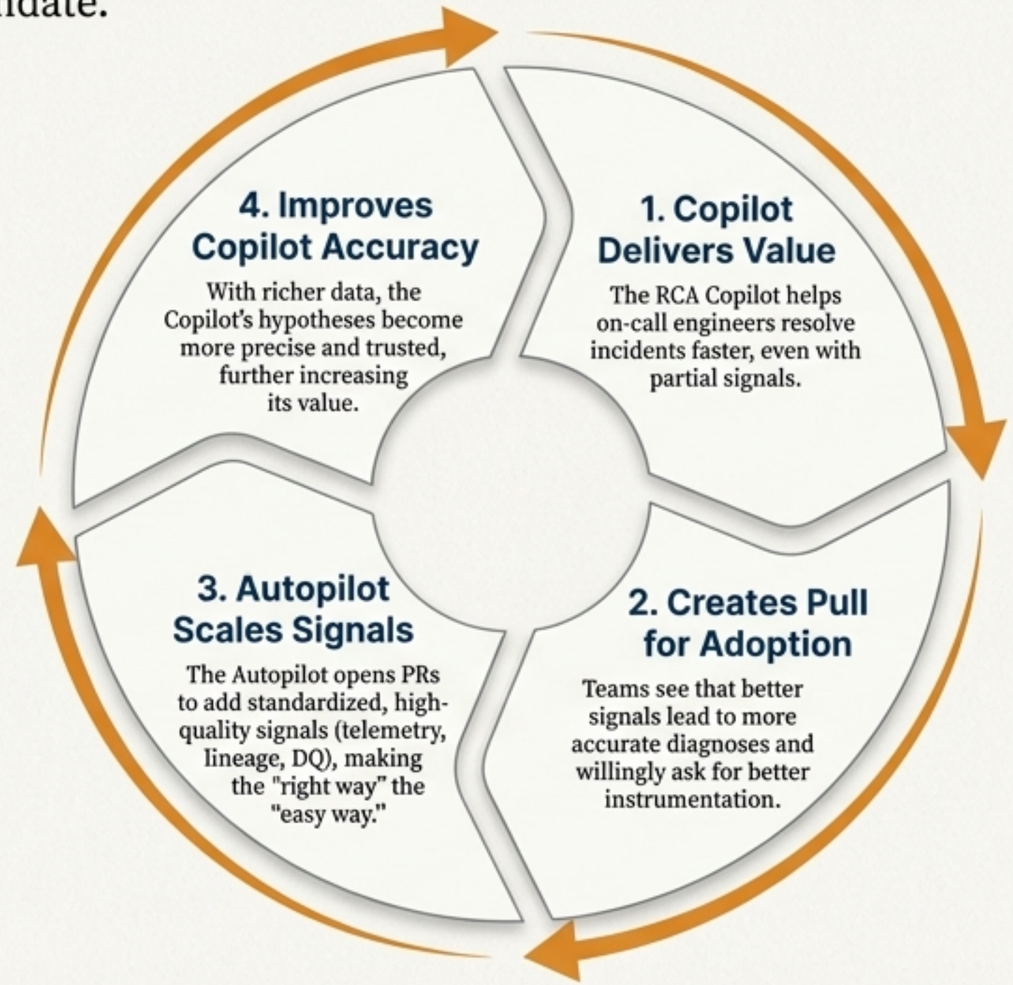
As we mature, we will move from informal SLOs to formal Data Contracts—machine-readable agreements defining the reliability guarantees between data producers and consumers. This makes accountability explicit and enforceable.

```yaml
data_product: urn:dataset:delta:orders_silver
tier: 1
owners:
  primary: orders-data
  oncall: pd://orders-data

# How different physical versions become comparable
compatibility:
    compare_mode: canonical_superset
    canonical_schema:
       fields:
         - { name: order_id, type: string }
         - { name: total_amount, type: double }
         - { name: discount_code, type: string, nullable: true, default: null }

# Guarantees monitored by our platform
slos:
    freshness_minutes_max: 60
    volume:
        min_per_day: 10_000_000
    dq_lite:
      - name: not_null(order_id)
      - name: range(total_amount, 0, 50000)

migration_policy:
    dual_write_window_days: 14
    # Enforced by CI/migration gates
    default_consume: canonical_view
```

Our observability platform will automatically compile these contracts into monitors, alerts, and CI gates.

The Flywheel of Adoption and Improvement

Our strategy is designed to create a self-sustaining cycle. The Copilot's value creates a natural "pull" for teams to adopt better instrumentation, and the Autopilot makes that adoption nearly frictionless. This is how we scale excellence without a mandate.



Our Commitment: The First Five Actions We Will Take

This plan is grounded in immediate, decisive action. We are ready to begin execution with a focus on de-risking our most **critical assumptions.**

1 **Validate** the **correlation strategy** across async boundaries with a manual trace-through of our pilot pipelines.

2 Define Copilot **accuracy targets** and measurement ***before*** a single line of code is shipped.

3 Establish the **Change Enablement workstream** with named champions and confirmed executive sponsorship.

4 **Lock in** the revised **30/60/90-day plan** and communicate it to all stakeholders.

5 **Document failure modes and rollback procedures** for both the AI Copilot and the Instrumentation Autopilot.