RJL SOLUTIONS

OPERATORS' CORNER / DATA ENGINEERING AND OPERATIONS

Build It to Run

AI-assisted data engineering from first pipeline to steady state.

CONTRACT · RUN KIT · LOOP

Richard White
RJL Solutions · rjlsol.com

VERSION 1 · PUBLISHED EDITION
NOVEMBER 2026

Most of the data budget keeps yesterday's pipelines running.

PART ONE · FOR SENIOR LEADERS

Ask a data engineering leader where the team's week goes and the answer is rarely new work. It goes to pipelines that broke overnight, a source system that changed a field without notice, a report that looks wrong to finance, and a backfill nobody planned. The published surveys agree on the direction, even though each comes from a vendor with something to sell.

53%
of engineering capacity spent maintaining and troubleshooting pipelines, 500 enterprise leaders, Fivetran 2026 [1]

4.7
pipeline failures a month on average, each taking nearly 13 hours to resolve [1]

67
data incidents a month on average, about 15 hours each to resolve, Monte Carlo 2023 [5]

74%
say business stakeholders find data issues first, all or most of the time [5]

Read these as signals. They are self-reported, vendor-commissioned and measured in different ways. Two older surveys put the same work at 44 percent of engineering time [2] and 40 percent of the workday [4]. Your own number could be higher or lower, and the first job is to measure it.

The usual response splits the problem. One group builds pipelines, another runs them, and AI is bought separately for each: a coding assistant for the builders and an observability tool for the operators. Each side gets faster. Upkeep cost barely moves, because most of it is decided at build time. A pipeline shipped without a contract, tests, lineage, an owner and a runbook will be expensive to run whatever tool watches it.

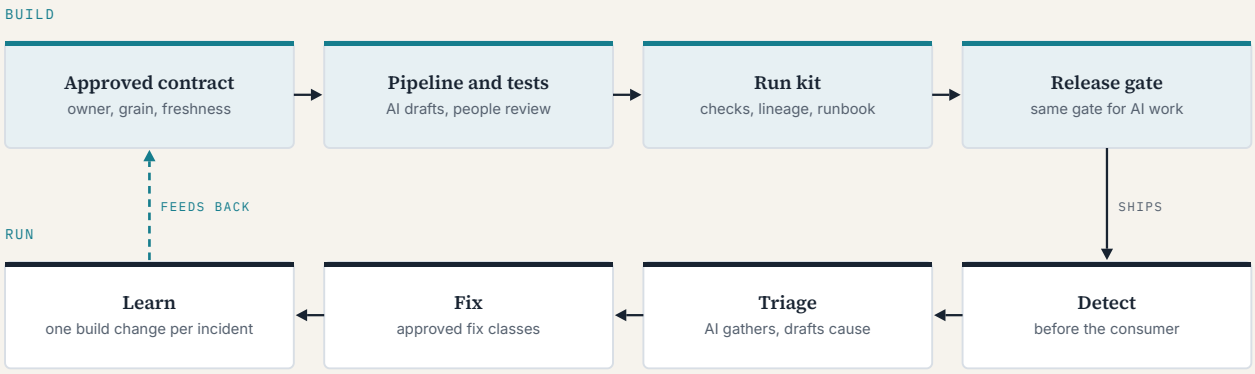


Exhibit 1. One loop. The build lane ships what the run lane needs, and each incident sends one change back into the build.

THE THESIS Treat build and run as one loop. The build ships what operations needs, and every incident changes the build. AI helps on both sides, inside the same release gates as any other change. Measure success as the share of engineering time that goes to new work.

02 / WHERE UPKEEP COMES FROM

Most breaks start upstream, and most of the cost is set at build time.

Pipelines rarely fail because the code was wrong on the day it shipped. They fail because something around them changed. In a 2021 survey of 543 data professionals, 98 percent said their pipelines break and 51 percent said it happens more than once a month. Those whose pipelines break daily named source availability and schema changes as the top causes, and 55 percent said a repair takes longer than one business day [3].

CAUSE OF UPKEEP	WHAT THE BUILD COULD HAVE SHIPPED	WHAT OPERATIONS DOES WITHOUT IT
A source adds, drops or renames a field	A contract test at the source boundary and a named source owner	Finds out from a failed job or a wrong report, then traces by hand
A file or feed arrives late or partial	A freshness and completeness check with an agreed clock	Answers a consumer's question about why today's numbers are missing
A business definition changes	A versioned definition with an owner and a change route	Edits logic under pressure, with no record of who decided
A consumer appears that nobody registered	Lineage and a consumer list kept with the pipeline	Breaks a downstream report during an otherwise routine change
Historical data needs a correction	A backfill plan and a replayable load	Writes a one-off script and hopes the totals match
Platform, library or credential changes	Pinned dependencies, upgrade tests and managed secrets	Patches each pipeline separately when something stops
Compute cost drifts upward	A cost budget and an owner who sees the bill	Finds out at quarter end

Every row has the same shape. A small piece of work at build time replaces a recurring piece of work in operation. That's why the loop matters more than either half. The 2021 Fivetran survey of 300 data leaders found that 80 percent had to rebuild pipelines after deployment [2]. Rebuilding is what happens when the first build didn't plan for change.

IMPLICATION Before buying anything for operations, sample twenty recent incidents and ask which row each one belongs to. The answer tells you how much of your upkeep is a build problem.

Separate the work a machine should do from the decisions people must make.

Google's site reliability engineering practice gives this work a name. Toil is work that is manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as a service grows. Google's SRE organization has an advertised goal of keeping toil below 50 percent of each engineer's time, because left alone it expands to fill all of it [7]. Data engineering has the same problem and rarely measures it.

KIND OF WORK	DATA ENGINEERING EXAMPLES	WHO SHOULD DO IT
Toil	Rerunning a failed load, re-pointing a renamed column, chasing a late file, regenerating documentation	Automation first, with AI drafting the cases a rule can't cover
Engineering	Designing a new pipeline, adding a contract test, building a replayable load, retiring a duplicate	Engineers, with AI drafting code, tests and documentation for review
Judgment	Deciding what open claim means, accepting a restatement, approving a new consumer of sensitive data	Named business and control owners. AI prepares the evidence and never decides

The categories matter because AI is very good at the first, useful for the second, and dangerous in the third. A model that can rewrite a mapping in seconds can also quietly change what a number means. The answer is to give it different authority for each kind of work.

Set a toil budget

Track, for each team, the share of time spent on toil, using incident tickets and a light weekly estimate. Set a ceiling, and treat a breach as a reason to fund build fixes. A budget turns upkeep from background noise into a number that competes for funding against new work.

DECISION Adopt a toil ceiling for data engineering, starting with the Google SRE goal of 50 percent and moving down as the loop matures. Review it with the same leaders who approve new data work.

04 / WHAT AI CHANGES

AI drafts on both sides of the loop. Production authority stays where it was.

AI is already part of data engineering. In dbt Labs' 2025 survey of 459 data practitioners and leaders, 70 percent of those who use AI daily use it for code development and 50 percent for documentation. Poor data quality remained the most reported challenge, at 56 percent [6]. Teams are writing code faster without a matching fall in quality problems.

SIDE OF THE LOOP	WHAT AI CAN DRAFT TODAY	WHAT STILL NEEDS A PERSON
Build	Pipeline code from a contract, tests from known cases, documentation, lineage notes, migration scripts	Approving the contract, reviewing the change, deciding what counts as correct
Run: detect	Summaries of alerts, grouping of related failures, plain-language status for consumers	Setting the objectives and deciding what pages a person at night
Run: diagnose	A first hypothesis from logs, lineage, recent changes and source status	Confirming the cause before any change
Run: fix	A proposed fix, a backfill plan, an impact list of affected consumers	Approving anything outside a pre-approved fix class
Learn	A draft incident record, a proposed new test, a runbook update	Deciding which build change to fund

The diagnosis evidence is promising and still early. A Microsoft study of more than 40,000 cloud incidents found that large language models could recommend plausible root causes and mitigation steps, and incident owners rated the approach useful in a human evaluation [8]. That was software infrastructure, with older models. Treat it as a reason to pilot, and measure the result in your own setting.

Calibrate the productivity claim too. A 2025 randomized trial found experienced developers in familiar codebases took 19 percent longer with AI tools, while believing they were faster [12]. Google's DORA research describes AI as an amplifier of the practices a team already has [9]. A team with contracts, tests and runbooks gets more out of AI. A team without them gets faster at producing upkeep.

PRINCIPLE AI proposes, on both sides. Every production change, whoever drafted it, goes through the same review and release gate. A model's suggestion never becomes its own authorization.

05 / LEADERSHIP DECISIONS

Fund the loop, measure the share of new work, and stop counting pipelines.

Gartner predicts that through 2026, organizations will abandon 60 percent of AI projects that aren't supported by AI-ready data [10]. In Fivetran's 2026 benchmark, nearly 97 percent of leaders said pipeline failures had slowed analytics or AI programs [1]. Reliable pipelines are the floor every AI ambition stands on, so upkeep belongs on the leadership agenda.

Start or strengthen

- **A new-work share for every team.** The share of engineering time on new capability, reported monthly next to delivery.
- **A definition of done that includes the run kit.** No pipeline is released until someone who didn't build it can operate it.
- **One owner for build and run.** The team that builds a pipeline carries its incidents, or a named successor accepts it formally.
- **Pre-approved fix classes.** A short list of routine repairs AI-assisted operations may apply without waiting for a person, each with a limit.
- **Incident learning with a budget.** Every material incident produces one funded build change.

Stop

- Counting pipelines shipped as the measure of a data team.
- Buying separate AI tools for builders and operators with no shared record of contracts, lineage and incidents.
- Hero on-call: the same two people who know how everything works, fixing it by hand at night.
- Letting consumers be the monitoring system. If business users find most issues first, the checks are in the wrong place.

> FROM THE FIELD · COMPOSITE
>
> A data platform leader inherits 300 pipelines and a team of twelve. A two-week sample shows roughly half their time on reruns, late files and schema surprises. Instead of hiring for operations, the leader funds a run kit for the 25 pipelines that cause most incidents and pre-approves three fix classes. After two quarters the team reports the new-work share each month, and incident counts on those 25 pipelines are the first number reviewed.

DECISION Before approving the next data platform or AI tool, ask three questions. What share of engineering time goes to new work today? Which pipelines cause most incidents? What will the build ship so those incidents stop recurring?

06 / BUILD FROM A CONTRACT

An approved contract is the cheapest place to prevent an incident.

PART TWO · FOR ENGINEERING AND DATA LEADS

Start every new pipeline from a contract that the producer, the consumer and the data owner have agreed. The Open Data Contract Standard, maintained under the Linux Foundation's Bitol project, gives a portable format for one [11]. The format matters less than the fields and the approval.

CONTRACT FIELD	WHY OPERATIONS NEEDS IT
Owner and support route	Tells triage who decides when the data looks wrong
Grain and keys	Lets checks find duplicates and missing records
Schema and allowed values	Lets a contract test fail fast when a source changes
Freshness clock	States when data is late, in the consumer's terms
Definitions, with versions	Separates a code defect from a change in meaning
Consumers and purposes	Makes the impact list for every change and incident
Change policy	Says how much notice a breaking change needs and who approves it
Sensitivity and access	Keeps diagnosis and backfills inside the same permissions

AI earns its place here by drafting. Given an approved contract, a model can propose the pipeline code, the tests, the documentation and the lineage annotation together, and a reviewer checks them against one record of intent. That is the approach set out in Rewiring Data Management with AI, which follows a contract from request to retirement.

Don't let AI write the contract's meaning. It can propose definitions from existing code and reports, and flag where two teams use the same word differently. The owner approves the definition, and the approval is recorded with a version.

PRACTICE No contract, no build. For an urgent request, approve a minimal contract with an owner, grain, schema and freshness clock, and mark the rest as pending.

07 / SHIP THE RUN KIT

A pipeline is done when someone who didn't build it can run it.

Most teams already have a definition of done for code: reviewed, tested, merged. Add a run kit, released with the pipeline and checked at the release gate. AI can draft most of it from the contract and the code. A person confirms it’s right.

RUN KIT ITEM	WHAT IT CONTAINS	HOW AI HELPS
Owner and on-call	Named owner, escalation route, hours of support	Fills from the contract and team roster
Checks	Schema, volume, freshness, null and range checks, with thresholds from profiling	Proposes thresholds from historical profiles
Lineage	Sources, transformations and consumers, emitted as the job runs	Annotates code and fills gaps for review
Runbook	Known failure modes, first checks, safe actions and who to call	Drafts from code paths and past incidents
Replay and backfill plan	How to reload a window safely, and how to verify totals afterwards	Drafts the plan and the reconciliation queries
Fix classes	Which repairs may run without a person, with limits	None. People define and approve these
Cost budget	Expected compute and storage, alert at a set overrun	Estimates from test runs
Consumer notice	How consumers are told about delays and corrections	Drafts the plain-language message template

Lineage is worth emitting at run time, from the job itself, with an open standard such as OpenLineage, also hosted by the Linux Foundation [11]. Lineage documented once by hand goes stale quickly. Lineage emitted on every run gives triage an accurate map the moment something breaks.

Google’s SRE teams use the same idea before they agree to support a service: a production readiness review that checks monitoring, capacity, failure handling and ownership before launch [7]. The run kit is that review, written for data. It is cheaper to hold the review at release than to discover the gaps during the first incident.

PRACTICE Add the run kit to the release checklist and make it a gate. A pipeline can launch with a partial kit only when its owner accepts the gap in writing and dates the fix.

Write the tests that production failures would have needed.

Unit tests on transformation code catch a small share of what goes wrong in data pipelines. The expensive failures are at the edges: a source that changed, a join that multiplied rows, a total that no longer matches the ledger. Test at several levels, and match the depth to the consequence of the data.

TEST LEVEL	WHAT IT CATCHES	WHEN IT RUNS
Contract test at the source boundary	Added, dropped or retyped fields, new values outside the allowed set	On every load, before transformation
Transformation tests	Logic errors on known cases, including the awkward ones	On every change, in CI
Grain and join checks	Duplicates, fan-out from joins, missing keys	On every change and every load
Reconciliation	Totals that drift from an authoritative source	On every load for financial or regulatory data
Consumer tests	A change that breaks a registered report, model or agent	Before release, against each registered consumer
Replay test	A backfill that would double-count or skip a window	Before any backfill reaches production

AI is good at writing the second and fifth kind once a person has named the cases. Ask it for the awkward cases too: the reopened claim, the reversed payment, the record migrated from a retired system. Then keep those cases as permanent fixtures. Data Is Not Just Code sets out the full layered approach and why a data rollback differs from a code rollback.

Every incident should add at least one test that would have caught it. Over a year, the test suite becomes a record of how the pipeline actually fails, which is worth more than any coverage percentage.

PRACTICE Put contract tests and reconciliation first. They catch the failures consumers notice, and they give AI-assisted triage a clear starting point.

09 / DETECT BEFORE THE CONSUMER

Set objectives in the consumer's terms, and page only for what they would notice.

If business users find most problems first, monitoring is watching the wrong things. In Monte Carlo's 2023 survey, 74 percent of data professionals said stakeholders identify issues first all or most of the time, up from 47 percent the year before, and respondents estimated 31 percent of revenue was affected by data quality issues [4][5]. Uptime checks pass while numbers are wrong. Data needs its own objectives.

OBJECTIVE	STATED IN THE CONSUMER'S TERMS	EXAMPLE THRESHOLD
Freshness	Data for the prior day is available by the time the report is used	Claims data loaded by 6:00 a.m. Eastern on 99 percent of business days
Completeness	All expected sources and partitions arrived	Every regional feed present, and row counts within the profiled range
Validity	Values fall within agreed rules	No unknown status codes, no negative paid amounts
Consistency	Totals agree with the system of record	Paid amount reconciles to the ledger within an agreed tolerance

Google's SRE Workbook makes the same point for data processing pipelines. Availability and latency are weak measures of whether a pipeline is serving its users. Freshness, how quickly data is processed, and correctness, how often processing goes wrong, come closer [13].

Each objective gets an error budget: one minus the objective, or the amount of failure the consumer has agreed to tolerate in a period [7]. When the budget runs low, the team shifts time from new work to fixes for that pipeline. That makes reliability a planned trade the business can see.

Alert less, and better

Page a person only when a consumer-facing objective is at risk and a person needs to act. Everything else becomes a ticket or a daily summary. AI helps by grouping related alerts into one incident and writing a short status a consumer can read. A team woken by twenty alerts for one late file will start ignoring alerts.

MEASURE Track the share of incidents found by checks before a consumer reports them. It is the clearest single test of whether the build shipped the right checks.

Let AI gather the evidence and draft the cause. People confirm it.

Triage is where AI saves the most time with the least risk, because most of the work is reading. An engineer paged at 6:10 a.m. spends the first half hour finding out what changed, where the data came from and who is affected. A model with read access to the run kit can assemble that in minutes.

01 **ALERT** A freshness objective is at risk for the claims feed.

02 **GATHER** AI reads lineage, the last successful run, recent code and contract changes, source status and open incidents upstream.

03 **DRAFT** AI states its best hypothesis, the evidence for it, what it ruled out and how sure it is.

04 **IMPACT** AI lists affected consumers from the contract and lineage, and drafts the consumer notice.

05 **CONFIRM** The on-call engineer confirms or rejects the hypothesis. A rejected one is recorded, with the reason.

06 **ACT** A fix in an approved class may run. Anything else waits for approval.

An authority ladder for operations

LEVEL	AI MAY	CONDITION TO MOVE UP
1. Read and explain	Read logs, lineage and run history. Draft hypotheses and notices	Hypotheses confirmed in most sampled incidents
2. Propose	Draft a fix, a backfill plan or a contract change for review	Proposals accepted with few corrections
3. Apply approved fix classes	Rerun a failed load, retry a late file, apply a known schema mapping within limits	A clean record at level 2 for that fix class, and an owner’s sign-off
Never	Change a definition, approve a restatement, widen access or delete history	Stays with people

PRACTICE Start every team at level 1. Give AI-assisted operations the same permissions as the on-call engineer it supports, and no more. Record each AI step in the incident, so a reviewer can see what it read and proposed.

11 / SCHEMA CHANGE

Turn the most common break into a routine change.

Source schema changes are among the most common reasons pipelines break [3]. They are also predictable. A field is added, renamed, retyped or removed, and the questions that follow are always the same: does any consumer depend on it, what does the new field mean, and how much history needs to change?

01 **DETECT** The contract test at the source boundary fails, before any transformation runs. Downstream data stays as it was.

02 **CLASSIFY** AI compares the new schema with the contract and classifies the change: additive, rename, retype or removal.

03 **PROPOSE** For a rename or additive change, AI drafts the mapping update, tests and an impact list from lineage.

04 **DECIDE** The contract owner approves. If the change alters meaning, the definition owner decides separately.

05 **RELEASE** The change goes through the normal gate, using expand, migrate, contract so consumers move on their own schedule.

06 **LEARN** The source owner is asked to give notice next time, and the contract's change policy is checked.

Additive changes and confirmed renames are good candidates for a pre-approved fix class once the team has a clean record with them. Retypes and removals are not, because they can change what a number means. The model’s classification can be wrong, so the class covers only cases where the evidence is unambiguous and the impact list is empty or confirmed.

The bigger gain is upstream. Share the contract with the source team, and ask for a change notice period. A source team that can see who depends on its fields usually starts announcing changes. That is a build fix for a run problem, and it costs a meeting.

PRACTICE Measure schema-change incidents separately. If the count isn’t falling after two quarters, the contract isn’t reaching the source owner.

Every material incident buys one change to the build.

Operations teams write incident reviews. Build teams rarely read them. Close that gap with a simple rule: every material incident ends with one funded change to the build, assigned to the owner and tracked to completion.

WHAT THE INCIDENT SHOWED	BUILD CHANGE IT SHOULD PRODUCE
The check that would have caught it didn't exist	A new contract, grain or reconciliation test
Triage took long because evidence was scattered	Run-time lineage, or a runbook entry for this failure mode
The fix was manual and repeatable	A candidate fix class, proposed for approval
The source changed without notice	A change policy agreed with the source owner
Consumers found out first	A consumer-facing objective and a status message template
Two teams disagreed about what the data meant	A definition decision, versioned in the contract

Google's published example error budget policy sets a useful threshold. If a single incident consumes more than 20 percent of the error budget over four weeks, the team holds a postmortem, and the postmortem must include at least one top-priority action to address the root cause [14]. Adopt the same rule for data, and write the reviews blameless. Google's postmortem practice focuses on the conditions that allowed the failure, so people report problems early [7].

AI can draft the incident record from the timeline it already helped assemble, and propose the build change. A person picks the change worth funding. Over time, track repeat incidents by cause. A repeat means a build change was skipped or didn't work.

Review the loop quarterly

- New-work share by team, against the toil ceiling.
- Incidents by cause, and repeats by cause.
- Share of incidents found by checks before consumers.
- Fix classes in use, their success rate and any that should be withdrawn.
- Build changes funded from incidents, and the ones still open.

PRACTICE Put build and run metrics on one page for the same leader. A loop with two owners and two dashboards stops being a loop.

Count released capacity honestly, and keep it apart from cash.

The economic case for this work is released engineering capacity, plus fewer business disruptions. Fivetran's 2026 benchmark puts average monthly business exposure from pipeline downtime and disruption at about $3 million for very large enterprises [1]. That is a survey-based estimate from a vendor and shouldn't go into a business case as a saving. Build the case from your own numbers.

LINE	ILLUSTRATIVE VALUE	HOW TO GET YOURS
Data engineers in scope	12	Headcount on the pipelines in scope
Share of time on upkeep today	50%	Two-week time sample plus incident tickets
Upkeep hours a year	12 × 1,800 × 50% = 10,800	Engineers × working hours × upkeep share
Share of upkeep the loop can remove	30%	Incident sample: share caused by missing build items or covered by fix classes
Capacity released	3,240 hours, about 1.8 engineers	Upkeep hours × removable share
Cost of the work	Run kits for 25 pipelines, AI tooling, review time	Estimate after the incident sample

Every value in the table is illustrative. Replace each one with a measured figure before using it. The removable share is the assumption most likely to be wrong, so test it against a real incident sample and report a range.

Capacity released isn't cash saved. It becomes cash only if the organization avoids a hire, cuts contractor spend or retires a tool. Otherwise it shows up as more new work, which is often the better outcome and should be reported as such. AI Token Economics sets out the method for separating capacity from cash and for counting the AI cost per accepted task, which here means cost per incident triaged and per fix accepted.

DECISION Report three numbers: capacity released, business disruption avoided, and cash actually realized. Fund the next phase on the first two only if the third has a named owner.

One claims feed, from build through its first three incidents.

This example is illustrative. The organization, data and timings are synthetic. A mid-sized insurer builds a daily claims feed for the finance and claims operations teams.

Build

The claims data owner, finance and operations agree a contract: one row per claim per day, status codes from an allowed list, data by 6:00 a.m. Eastern, and open claim defined as version 1 of the operations definition. AI drafts the pipeline, the tests and the run kit from the contract. An engineer reviews them, adds three awkward cases from profiling and releases the feed.

INCIDENT	WHAT HAPPENED	WHO DECIDED	BUILD CHANGE
1. New status code	The source added code RO for reopened. The contract test failed before transformation, so no report showed wrong numbers. AI classified it as an additive value change and drafted the mapping	Data owner confirmed RO counts as open under version 1	RO added to the allowed list and to the test fixtures
2. Late file	A regional file arrived 90 minutes late. The freshness objective fired. AI found the upstream job's delay in the source status log and drafted a consumer notice	On-call engineer confirmed the cause. A retry ran as a pre-approved fix class	Retry window added to the runbook. Source owner asked for a delay signal
3. Definition request	Finance asked to exclude claims reopened only for a document request. AI listed four affected consumers and drafted the change	Finance and operations owners agreed version 2, effective next quarter	Definition versioned. Both versions kept for comparison during the change

None of the three incidents reached a consumer as a wrong number. The first was caught by a contract test the build shipped. The second was routine because the run kit defined a safe action. The third was a decision about meaning, and AI's role was to make that decision quick and well informed.

WHAT TO NOTICE Each incident left the pipeline easier to run. That is the loop working. If your incidents leave nothing behind, you are paying for the same failure more than once.

Measure first, then fix the pipelines that cost the most.

WEEKS	WORK	EVIDENCE AT THE END
1 to 3	Sample engineering time for two weeks. Classify the last 60 days of incidents by cause. Rank pipelines by incident hours	A baseline new-work share and a top-25 list
4 to 8	Write or complete contracts and run kits for the top 25. Turn on AI-assisted triage at level 1 for those pipelines	Contracts approved, run kits passing the gate, triage hypotheses logged
9 to 13	Approve two or three fix classes. Start the incident-to-build rule. Hold the first quarterly loop review	Fix classes in use with limits, build changes funded, first trend lines

Measures to report from week one

MEASURE	DEFINITION
New-work share	Share of engineering time on new capability, by team
Incident hours	Time from detection to restored objective, by pipeline and cause
Found by checks first	Share of incidents detected before a consumer reported them
Repeat incidents	Incidents with the same cause on the same pipeline within 90 days
AI triage accuracy	Share of AI hypotheses the engineer confirmed, sampled weekly
Fix class success	Share of automatic fixes that restored the objective without follow-up

Keep the first phase narrow. Twenty-five pipelines with full run kits will show more than three hundred with partial ones. Lean teams can start with five: the ones that wake someone up most often.

THE GOAL A data engineering team that spends most of its time on new work, finds problems before its consumers do, and lets AI take the reading and the routine repairs while people keep the decisions.

Two checklists to copy into your first release.

Release checklist: run kit

- Approved contract with owner, grain, schema, freshness clock and consumers
- Contract test at the source boundary
- Grain, join and reconciliation checks with thresholds from profiling
- Run-time lineage emitted from the job
- Runbook: known failures, first checks, safe actions, escalation
- Replay and backfill plan with verification queries
- Fix classes listed, with limits and approver
- Cost budget and alert
- Consumer notice template
- Named owner who accepts the incidents

Incident record: fields

- Pipeline, objective breached, detected by check or by consumer
- Timeline, including each AI step and who confirmed it
- Cause category: source, schema, timing, definition, consumer, platform or cost
- AI hypothesis, confirmed or rejected, with the reason
- Fix applied, fix class if any, and who approved it
- Consumers affected and notices sent
- Build change chosen, owner and due date
- Repeat of an earlier incident: yes or no

Both lists work as fields in a ticketing system or as a section in the pipeline's repository. Keep them next to the code, so a reviewer sees the run kit in the same change as the pipeline.

COMPANION PAPERS

Rewiring Data Management with AI covers the contract lifecycle in depth. Data Is Not Just Code covers layered testing and release. AI Token Economics covers cost per accepted task and the capacity-to-cash bridge.

Sources and how they were used.

Prepared by Richard White. Version 1, November 2026. Survey figures are reported as their sources state them. Most come from vendor-commissioned surveys, which are useful for direction and weak for precise comparison. The field note is a composite with details changed. The worked example and the economics are illustrative. Sources checked 10 November 2026.

[1] Fivetran, *The Enterprise Data Infrastructure Benchmark Report 2026*, press release 26 March 2026. Survey of 500 senior data and technology leaders at enterprises with more than 5,000 employees, fielded Q4 2025. Vendor-commissioned. fivetran.com/press

[2] Fivetran and Wakefield Research, "Data and Analytics Leaders Report Wasting Funds on Bad Data," 17 November 2021. Survey of 300 data and analytics leaders at companies with revenue above $100 million. 44 percent of engineering time on pipelines, 80 percent rebuilding pipelines after deployment. fivetran.com/press

[3] Fivetran, "New Survey Finds More Than Two Thirds of Companies Leave Valuable Data Untapped," 11 March 2021. Survey of 543 data professionals. Break frequency, causes of daily breaks, repair time. fivetran.com/press

[4] Monte Carlo and Wakefield Research, data quality survey, 9 August 2022. Survey of 300 data professionals. 40 percent of the workday on data quality, 47 percent saying stakeholders are affected by issues the team misses. businesswire.com

[5] Monte Carlo and Wakefield Research, *State of Data Quality*, 2 May 2023. Survey of 200 data professionals, fielded March 2023. 67 incidents a month, 15 hours to resolve, 74 percent found first by stakeholders, 31 percent of revenue estimated affected. businesswire.com

[6] dbt Labs, *2025 State of Analytics Engineering*, April 2025. Survey of 459 data practitioners and leaders, fielded 8 October to 27 December 2024. getdbt.com

[7] Beyer, B. et al. (eds.), *Site Reliability Engineering*, O'Reilly and Google, 2016, chapters 4 ("Service Level Objectives"), 5 ("Eliminating Toil"), 15 ("Postmortem Culture") and 32 ("The Evolving SRE Engagement Model," production readiness reviews). Definition of toil and the 50 percent goal. sre.google/sre-book

[8] Ahmed, T. et al., "Recommending Root-Cause and Mitigation Steps for Cloud Incidents using Large Language Models," ICSE 2023. More than 40,000 Microsoft incidents, with a human evaluation by incident owners. arxiv.org/abs/2301.03797

[9] DORA (Google Cloud), *State of AI-assisted Software Development*, September 2025, and the DORA AI Capabilities Model. Survey-based associations. cloud.google.com

[10] Gartner, "Lack of AI-Ready Data Puts AI Projects at Risk," 26 February 2025. Prediction through 2026. gartner.com

[11] Bitol project, Open Data Contract Standard, and OpenLineage, both hosted by the LF AI & Data Foundation. bitol-io.github.io; openlineage.io

[12] Becker, J., Rush, N. et al., METR, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity," July 2025. 16 developers, randomized by task. METR later redesigned the study. metr.org

[13] Beyer, B. et al. (eds.), *The Site Reliability Workbook*, O'Reilly and Google, 2018, chapter 13, "Data Processing Pipelines," and Google Cloud Monitoring guidance on service level indicators for data processing. Freshness and correctness as pipeline measures. sre.google/workbook

[14] *The Site Reliability Workbook*, appendix B, "Example Error Budget Policy." The 20 percent single-incident threshold and postmortem rule. sre.google/workbook

Proposed operating guidance. Examples describe no client engagement or client result.

ABOUT

Richard White

I was global firm chief data officer at McKinsey & Company, owning data, AI and platforms end to end, and before that chief data officer at New York Life, where I built the chief data office and reported data risk to risk committees at multiple levels. Earlier I led data at the Federal Reserve Bank of New York and enterprise data governance at Citigroup. I founded RJL Solutions in 2024. My book, *The Day After the Demo*, is free at rjlsol.com/book.

RJL Solutions connects strategy, architecture, engineering, governance and operation through Frame, Prove, Build, then Operate and transfer. Each phase is separately scoped and priced in a written proposal before you commit. The first sixty-minute working session is free.

LIFECYCLE **Rewiring Data Management with AI**
Approved contracts and bounded agents, from request to retirement.

DELIVERY **Data Is Not Just Code**
Context, layered testing and a release pipeline matched to consequence.

ECONOMICS **AI Token Economics**
Cost per accepted task, and capacity kept apart from cash.

RJL SOLUTIONS

WEB rjlsol.com

EMAIL rwhite@rjlsol.com

Examples are illustrative and describe no client engagement or client result. The field note is a composite with details changed. Survey and study figures are reported as their sources state them.