arsalan@bruin:~/build-lab$ ./present.sh

SELF-HEALING PIPELINES

> agents that fix data incidents themselves

arsalan noorafkan - devrel, bruin

tue aug 18 2026 · 14:00 utc · google meet · 45m + q&a

join the bruin data community

slack · scan or click

bruin

what we'll cover

00:00–00:10	overview
00:10–00:25	context
00:25–00:45	workflow
00:45–01:00	recap & Q&A

by the end: you'll know which gaps in your stack to close for self-healing

why it matters

~40%

of the workday on quality [1]

68%

need 4+ hrs to detect [2]

74%

hear it from someone else [2]

* self-reported, vendor-commissioned, n=200-300 - the shape of the problem, not a measurement of yours

[1] monte carlo / wakefield 2022 (n=300)
[2] monte carlo / wakefield 2023 (n=200)

bruin

what is a self-healing pipeline?

bruin

how it works

context

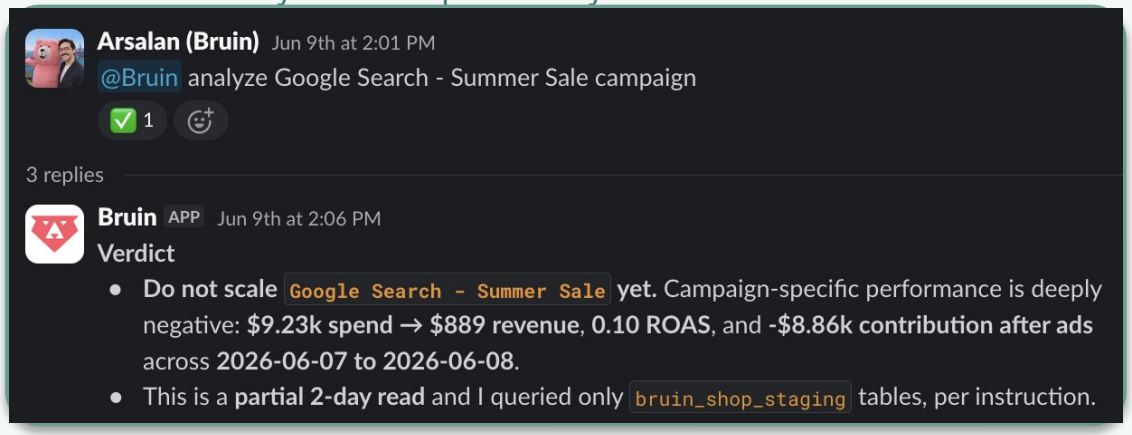
tools

guardrails

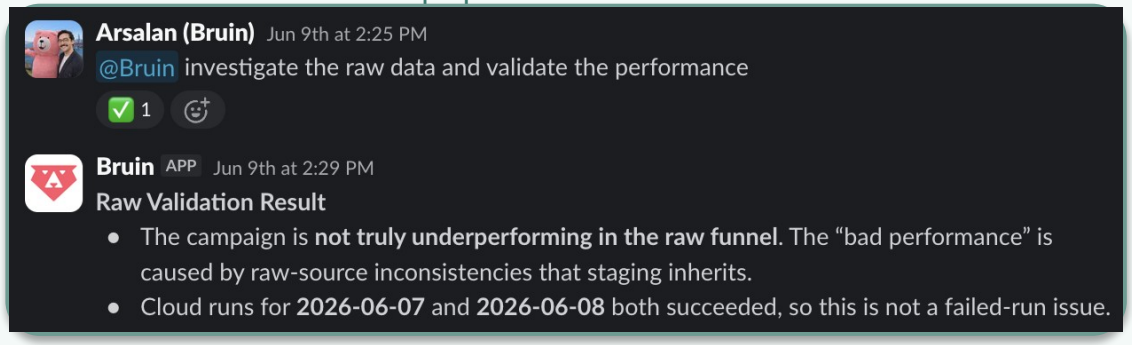
bruin

what missing context costs

access to only the report layer



access to the whole pipeline



bruin

the problem, and what fixes each one

incomplete	siloed	unmaintained
↓	↓	↓
enrich it	unify it	feedback loop

bruin

the modern stack - context in silos

fivetran

warehouse

dbt

airflow

bi & docs

what you already have

code

schemas

lineage

docs

bruin

enrich it - repo-level context

AGENTS.md

Bruin

```
# AGENTS.md

This file gives AI agents the repo-specific context they need to work in the
Harborside data platform: how the company and its pipelines are organized, the
standard every asset is held to, and the self-healing policy that says what an
agent may do on its own. Read it before opening any SQL.

Two more places hold context an agent should reach for:

- **`glossary.yml`** at the repo root — the shared semantic layer (entities and
  attributes) that asset columns extend.
- **`README.md` in each pipeline folder** — why that pipeline exists, why it is
  shaped the way it is, its lineage, and its designed incidents.
```


bruin

enrich it - shared definitions

glossary.yml

Bruin

```yaml
# glossary.yml
domains:
  commerce:
    description: Storefront orders, products, and one-time revenue.
    owners:
      - Dana (Finance)
    tags:
      - commerce
    contact:
      - type: email
        address: dana@harborside.example
```


bruin

enrich it - pipeline configuration

pipeline.yml

Bruin

```yaml
name: shop
schedule: daily
catchup: false
retries: 2
tags:
  - shop
domains:
  - finance
meta:
  team: finance
  cost_center: FIN-204
  description: One-time storefront revenue.
variables:
  anomaly_multiplier:
    type: integer
    default: 2
notifications:
  slack:
    - channel: "#data-alerts"
```


bruin

enrich it - pipeline intent

README.md

Bruin

```
# shop — one-time storefront revenue

The storefront side of Harborside's revenue: one-time orders (as opposed to the
Home Box subscription, which lives in the `stripe` pipeline). This pipeline
conforms raw orders and products, then publishes the daily revenue tables Finance
and the exec dashboard read.

**Consumers:** the exec revenue dashboard and Finance ad-hoc analysis.
**Owner:** Dana (finance), `dana@harborside.example`. Cost center `FIN-204`.

## Why this pipeline looks like this

- **Dedup lives in the stage layer, once.** Raw orders are append-only and can
  carry repeated `order_id`s (retries, connector replays). `shop_stage.orders_clean`
  is the single place that resolves them, so every report is duplicate-free
  without each one re-implementing the rule.
```


bruin

enrich it - table identity

name, type and description

Bruin

```yaml
name: shop_report.orders_enriched
type: bq.sql
connection: gcp-default
uri: bigquery://harborside-analytics.shop_report.orders_enriched   # cross-pipeline address

description: >
  Order-grain fact enriched for analysis: one row per order with its product
  category, whether the buyer is a billing customer, and how they were acquired.
```


bruin

enrich it - table metadata

owner, tags, domains and meta

Bruin

```yaml
owner: dana@harborside.example
tags:
  - harborside
  - shop
  - report
  - reference
domains:
  - commerce
  - finance
meta:
  layer: report
  tier: gold
  team: finance
  cost_center: FIN-204
  sla: "06:30 UTC"
  grain: one row per order
```


bruin

enrich it - column-level definitions and checks

columns, keys and checks

Bruin

```yaml
columns:
  - name: order_id
    extends: Order.OrderId          # pull type/description from glossary
    type: VARCHAR
    description: Storefront order identifier; the grain.
    primary_key: true
    nullable: false
    checks:
      - name: not_null
      - name: unique
```


bruin

enrich it - custom checks and unit tests

custom_checks and unit_tests

Bruin

```yaml
custom_checks:
  - name: no_non_positive_amounts     # custom SQL check: query result must equal value
    description: Every order must have a positive amount.
    query: SELECT COUNT(*) FROM {{ this }} WHERE amount_usd ⩽ 0
    value: 0
    blocking: true

unit_tests:                           # one order in → assert its category
  - name: order_gets_its_product_category
    inputs:
      - asset: shop_stage.orders_clean
        rows:
          - {order_id: O1, product_id: P1, order_date: "2026-05-01"}
      - asset: shop_stage.products_clean
        rows:
          - {product_id: P1, category: home}
    expected:
      rows:
        - {order_id: O1, category: home}
```


bruin

enrich it - lineage and materialization

dependencies, materialization, late data

Bruin

```yaml
depends:                                                    # 3 dependency kinds:
  - shop_stage.orders_clean                                 # regular (blocking)
  - asset: shop_stage.products_clean                        # symbolic (lineage only)
    mode: symbolic
  - uri: bigquery://harborside-analytics.stripe_stage.customers   # external (cross-pipeline)

materialization:
  type: table
  strategy: time_interval          # rebuilds one interval per run
  incremental_key: order_date
  time_granularity: date
  partition_by: order_date
  cluster_by:
    - country

interval_modifiers:
  start: -2h                       # source runs late: shift start back 2h
```


bruin

unify it - one place the agent can trust

A

buy a context layer

or glue it together yourself

atlan

datahub

B

one platform

nothing to unify

bruin

fivetran + dbt

databricks

bruin

give the agent tools

bruin

dbt

airflow

fivetran

skills

AGENTS.md

bruin

the workflow

same agent, same context - more access at each step

bruin

1 · local - start here

clone & connect → investigate → propose fixes

a dev environment · a schema prefix

bruin

example - one failure, five steps

completed → fulfilled

1 detect

a check fails

→

2 diagnose

the app repo

→

3 fix

match the rename

4 verify

pass in dev, open a pr

→

5 feedback loop

ask why, write it down

bruin

2 · ci/cd and production - the access you grant

on every pull request

- validate
- run it in dev
- impact assessment

in production

- run logs
- actions
- the repo
- a channel

where does your loop stop?

detect	checks and run status
diagnose	run logs, lineage, the app repo
fix	a dev environment and a pull request
verify	the ability to rerun
learn	somewhere to write it back

bruin

the learn stage - a real feedback loop

a correction	→	into the glossary
a silent failure	→	into a quality check

the repo, not agent memory

bruin

automate the loop

scheduled

goes looking

triggered

fires on failure

the morning report

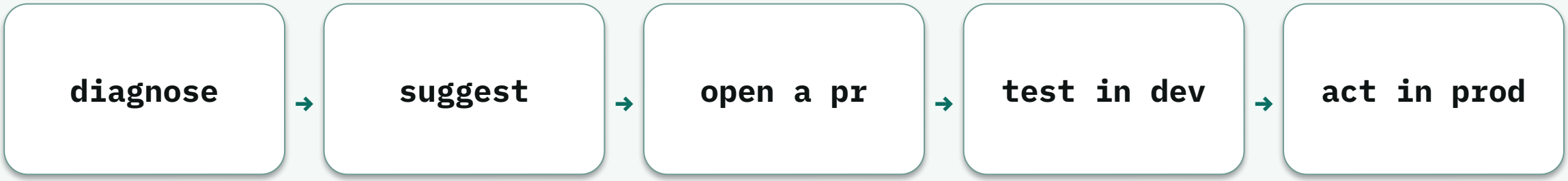
what failed

what it fixed

what needs you

bruin

how far you let it go



more oversight

▲
most teams are here

more autonomy

bruin

recap

context

tools

guardrails

loop

```
1   map your context
2   enrich and unify it
3   run locally, in dev
4   ci/cd, then production
```


bruin

references & further reading

bruin docs

- bruin mcp
- cloud mcp
- quality checks
- environments
- ci/cd github action
- scheduled agents
- bruin academy

other tools

- dbt mcp
- airflow mcp (astronomer)
- fivetran mcp
- greptile
- coderabbit

context layers

- atlan
- datahub

sources

- [1] monte carlo 2022 survey
- [2] monte carlo 2023 survey
- [3] anthropic on self-service data analytics

platform consolidation

- fivetran + dbt labs merger june 2026

community

- bruin community on slack

bruin

questions?

- the bruin data community
- getbruin.com/learn
- getbruin.com/docs

join our slack community

thank you

bruin