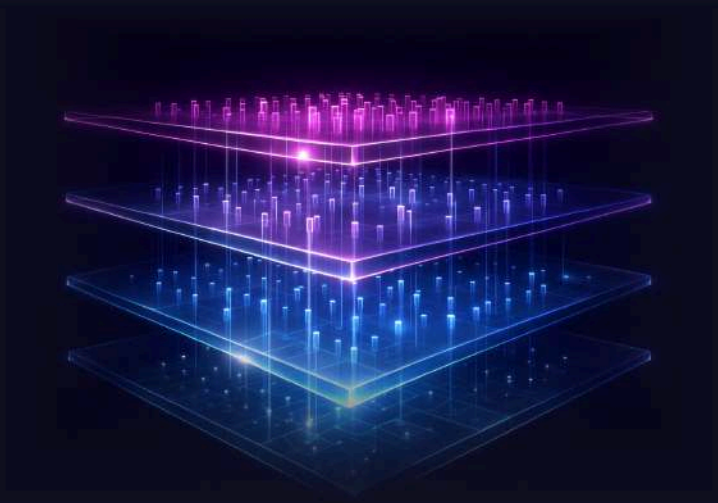
gable

CODE-LEVEL DATA LINEAGE

Field-level provenance from source code
to downstream consumers

gable

TECHNICAL PRODUCT FACT SHEET



What Gable does

Gable uses static analysis of application source code to produce field-level data lineage. It traces each data element from the code that creates or transforms it, through services and pipelines, to the models, regulatory reports, and balance sheet figures that depend on it.

Most lineage tools parse SQL, dbt, or Spark inside the data platform. That covers transformations between tables, but the code that produces and shapes data in the first place (Java services, Kotlin microservices, Python applications, event producers) is invisible to them. Gable starts where those tools stop: at the application code where data originates, so you can trace a field on a report or model input back to the service that created it.

Field-level lineage from source code

Static analysis maps every data element to its source, transformation, service, and downstream consumers. Lineage is at the column and attribute level, not just table-to-table.

Change-aware provenance

Lineage is reconstructed from the codebase and tied to commits and pull requests. When code ships, lineage updates. Teams see what changed, which fields are affected, and what depends on them before a merge lands.

Control and policy evidence per field

DLP, masking, encryption, and access policies attach to specific data elements. Evidence follows the field rather than living in a separate controls spreadsheet.

Evidence packaging

Lineage, change history, and control evidence export as a single packet for audits, procurement reviews, model risk assessments, or incident response.

gable

TECHNICAL PRODUCT FACT SHEET

How it works

Gable scans application repositories and detects data assets, egress points, and the code paths between them. It supports Java, Kotlin, TypeScript/JavaScript, Python, and selected PySpark workflows. The output is a map from application code through services to the data stores, models, and reports downstream, so you can answer how a specific field on a regulatory submission or model input was derived.

Data contracts define expected schema, semantics, and SLAs for each asset. CI/CD checks validate changes against those contracts before merge, so downstream breakage surfaces in the pull request, not in production.

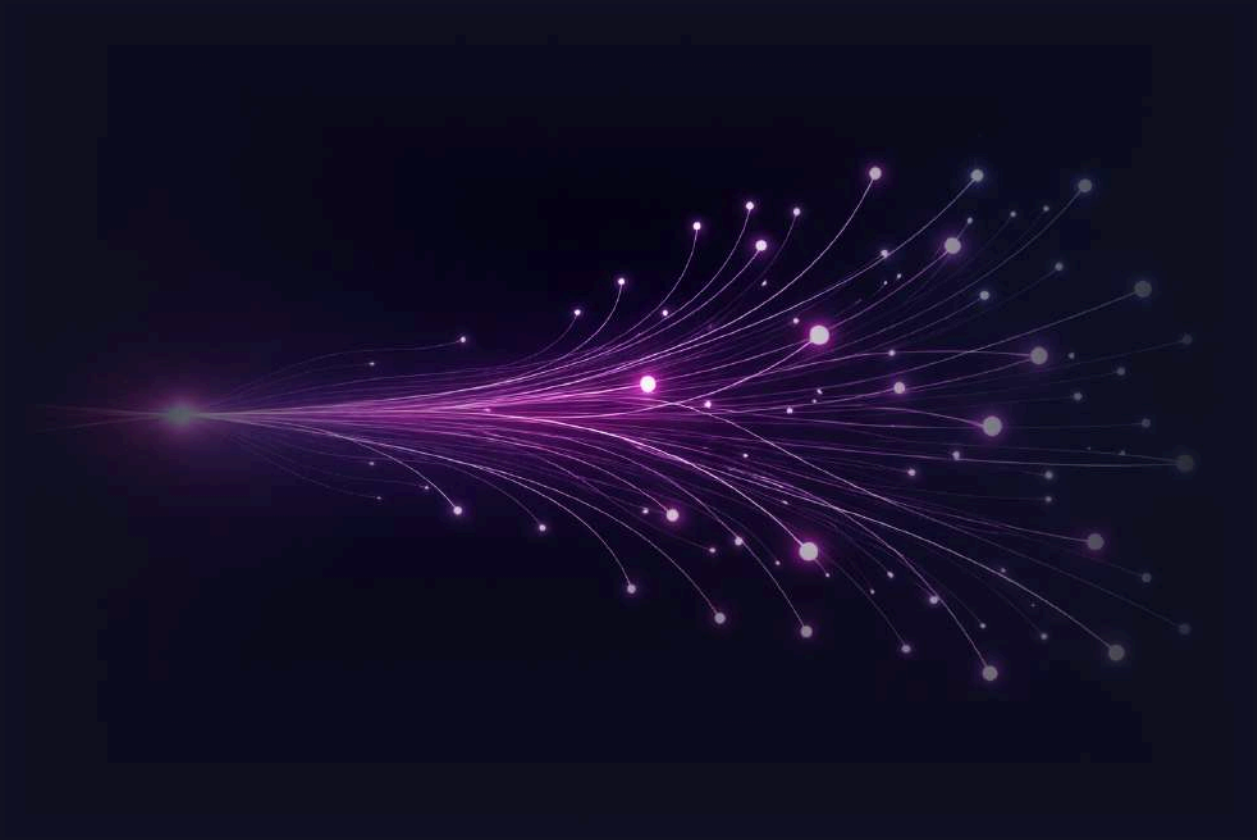
Where source-code lineage fits

Lineage tier	What it covers	Typical tools
Table / storage lineage	Table-to-table metadata relationships	Data catalogs, tag propagation
Database column lineage	SQL, dbt, Spark transformations inside the data platform	Atlan, dbt, warehouse-native lineage
Source-code lineage	Application code that produces, transforms, and emits data at point of origin	Gable

These tiers are complementary. Database column lineage explains transformations between tables. Source-code lineage explains what happens before data is stored, inside the services and applications that produce it. To trace a balance sheet figure or model input back to its origin, you need both.

gable

TECHNICAL PRODUCT FACT SHEET



See what's inside your code.

Gable is deployed in banking, with applicability across healthcare, insurance, and pharma. If your organization needs to trace how a field on a report, model input, or regulatory submission was derived from application code, request a demo to see source-code lineage on your stack.

Request a demo

gable.ai

info@gable.ai

gable