DELIVERING DATA ENGINEERING AT SCALE

AI DATA INTEGRATION FRAMEWORK (AI DIF)

MAY 2026

WHY NOW

Data engineering needs an agentic upgrade

Pipelines are still 80% manual; AI is finally good enough to take over the metadata-driven repetitive work.

THE PROBLEM

80%

of pipeline build effort is repetitive metadata work — config, deploy, test, monitor

THE LANDSCAPE

4–8×

platform fragmentation: most enterprises run two or more of Snowflake / Databricks / dbt / Fabric in parallel

THE OPPORTUNITY

No vendor lock in

vendor lock-in tolerated by enterprises now that LLMs can generate target-specific code from one metadata model

The agentic shift

Single-responsibility AI agents — model · code · deploy · test · observe — collaborate over MCP and A2A under a Controller GenAI. Humans set goals and review; agents do the work. The result: data platforms in days, governed and observable from line one.

WHAT IS AI DIF

An advanced toolkit for deploying and managing metadata-driven ETL — across platforms

AI DIF integrates an agentic swarm, a metadata core, and target-specific code generators for Snowflake, dbt, Databricks-native, and DLT-meta.

01

Simplifies

Lowers the barrier to deploying enterprise-grade data platforms — agents handle the boilerplate.

02

Accelerates

Most data use cases reach production in days; mass-deployment of pipelines becomes practical.

03

Governs

Built-in metadata core, lineage, PII detection, RLS — observability and audit from day one.

04

No lock-in

Same metadata produces Snowflake, dbt, Databricks-native, and DLT-meta artefacts.

05

Frees teams

Critical engineers refocus on business value — the framework absorbs the repetitive work.

AI DIF combines: Windsurf / GitHub Copilot for code generation · custom plugins for extensibility · MCP servers exposing the target environment (Databricks, Snowflake, dbt) · a metadata-driven core with embedded governance and observability.

Reduces manual config and deployment effort by ~80% while keeping every artefact reviewable in Git.

THE AGENTIC SWARM

Controller GenAI orchestrates single-responsibility agents over MCP and A2A

Humans give intent; agents do the work; everything is governed, reviewable, and reversible.

CONTROLLER GenAI

"Create a cloud platform with ingestion from SAP, a Data Warehouse, DataMarts, and Power BI.
Scan the source for tables, choose models, here are credentials."
→ Scans the IT topology · discovers business context · takes architectural decisions · asks for fine-tuning when needed · executes via single-responsibility agents.

MODEL

Architect Agent

Reads infra and best practices; produces ADRs; drafts target architecture for the chosen platform.

MODEL

Modeller Agent

Scans source schemas via Unity Catalog / Cortex / connectors; suggests data products, mappings, PII flags.

BUILD

Code-gen Agent

Emits target-specific artefacts: Snowflake SQL/SP, dbt configs, Databricks wheels, DLT-meta YAML.

DEPLOY

Deployer Agent

Resolves dependencies (DAG), creates tables, deploys workflows and jobs — one-shot, idempotent.

VERIFY

Tester Agent

Auto-generates unit / integration / reconciliation tests; runs them post-deploy and gates promotion.

RUN

Observer Agent

Captures KPIs, freshness, anomalies; routes incidents to Teams / ServiceNow with proposed fixes.

Standards: MCP (Model Context Protocol — tools/data) · A2A (Agent-to-Agent — orchestration). All agents are reviewable in Git; human-in-the-loop on every critical decision.

METADATA-DRIVEN CORE

One metadata model — three time-scoped schemas — fed into every agent and every code generator

The single source of truth that makes mass-management of pipelines possible. Agents read it; code generators emit from it.

DESIGN-TIME

GOVERNANCE

Catalog of intent: data products, ETL definitions, lineage edges, table & column definitions, workflows, product dependencies. Loaded from DIF config files; the blueprint of the platform.

RUN-TIME

TECH_STATS

What actually happened: pipeline run registry, technical statistics, reconciliation, anomaly detection, data-quality KPIs. Populated automatically by every job execution.

DEBUG-TIME

LOGGING

Internal trace of how it happened: structured telemetry from stored procedures, captured to Snowflake / Databricks Event Tables for post-mortem and forensics.

HOW THE METADATA IS USED

Single source of truth, read by everything

- Code generators emit Snowflake / dbt / Databricks / DLT-meta artefacts from the same model
- Agents query metadata to make architectural decisions
- Deployment automation builds dependency DAGs from lineage edges
- Observability dashboards surface run history, quality, and reconciliation
- Onboarding a new table = inserting a metadata row, not writing code

FOUR PLATFORM VARIANTS

One metadata model → four sets of native artefacts

AI DIF is platform-agnostic at the metadata layer. Code generators emit target-specific artefacts that run as first-class citizens on each platform.

VARIANT A

Snowflake

- Config files for native objects
- Stored procedures (SP)
- or direct deploy via Cortex Code
- Native Snowflake security, RBAC, RLS

VARIANT C

Databricks-native

- Wheels & Databricks notebooks
- Workflows with DAG resolution
- Unity Catalog tables
- Jobs deployed via Databricks SDK

AI DIF

METADATA CORE

Governance
Tech_Stats
Logging

+ Agents
(architect, modeller, code-gen, deployer, tester, observer)

VARIANT B

dbt

- dbt project configs and models
- Cross-platform dbt Mesh
- Runs on dbt Cloud or dbt Core
- Emits SQL for any warehouse

VARIANT D

DLT-meta

- YAML/JSON metadata configs
- Spark Declarative Pipelines
- Templated DLT pipelines at scale
- Onboarding 1000s of tables

Customers run Snowflake AND Databricks AND dbt — AI DIF supports them all from one declarative model.

VARIANT A · SNOWFLAKE

Native Snowflake artefacts emitted from the metadata core

Three deployment modes — choose what fits the customer's existing Snowflake operating model.

MODE 1

Config-only

- Emits declarative configs (YAML/JSON)
- Customer's existing CICD applies them
- Most flexible · least invasive
- Recommended for established Snowflake estates

MODE 2

Stored procedures

- Generates Snowflake SQL stored procedures
- Includes ETL logic, parameter handling, error capture
- Deployed via Snowflake Tasks
- Proven, version-controlled, debuggable

MODE 3

Direct deploy

- AI DIF deploys SPs directly via Snowflake API
- Cortex Code integration where available
- Fastest path from metadata to running pipeline
- Ideal for greenfield Snowflake-only customers

Snowflake-native technology stack

Snowpark · Cortex Code · Snowpipe · Streams · Tasks · Stored Procedures · Snowflake Account Usage · Data Metric Functions · Network Rules + Secrets + External Access Integration · Notification Integration

Governance: Snowflake Horizon (object tagging, masking policies, row access policies). PII detection during code-gen, RLS at table-creation time.

VARIANT B · DBT

dbt projects generated from metadata — runs on every warehouse dbt supports

AI DIF is the metadata factory; dbt is the transformation runtime. Cross-platform dbt Mesh becomes practical at scale.

PROJECT GEN

Full dbt project from metadata

- models/, seeds/, macros/, tests/ folder structure
- sources.yml and schema.yml from source schemas
- Standard layered architecture (staging → intermediate → marts)
- Configurations target dbt Core or dbt Cloud

MODELS

Generated SQL with patterns

- Star-schema and Data Vault patterns
- Incremental and snapshot strategies
- Metric definitions for the dbt Semantic Layer
- Codified best practices applied automatically

TESTS

Tests and contracts at scale

- Source freshness checks
- Generic tests (unique, not_null, accepted_values, relationships)
- Custom singular tests for business rules
- Model contracts for stable interfaces

MESH

Cross-platform dbt Mesh

- Multiple dbt projects, one mesh — coordinate across warehouses
- Lineage stitched in dbt Explorer
- Iceberg tables enable shared physical layer
- Governed re-use across teams

Runs on: Snowflake · Databricks · BigQuery · Redshift · Postgres · DuckDB. Supports dbt Cloud, dbt Core, and Snowflake's Cortex Code dbt deployment.

VARIANT C & D · DATABRICKS

Two complementary modes for Databricks customers — native and DLT-meta

Native suits custom logic and existing wheels; DLT-meta suits declarative pipeline-templating at scale.

VARIANT C · DATABRICKS-NATIVE

Wheels, notebooks, workflows — full Databricks runtime

- Wrapper notebooks executing the DIF code core (one wheel)
- Configs embedded at deploy time — no runtime config reads
- Workflow generator builds DAGs from lineage edges
- Tables created in Unity Catalog with row-level security
- Unity Catalog volumes for landing; Delta for storage
- MCP server exposes Databricks to Windsurf for code-gen
- Genie + Cortex AI integrations for in-notebook reasoning
- First-class observability via Databricks Asset Bundles + ABC_METADATA

VARIANT D · DLT-META

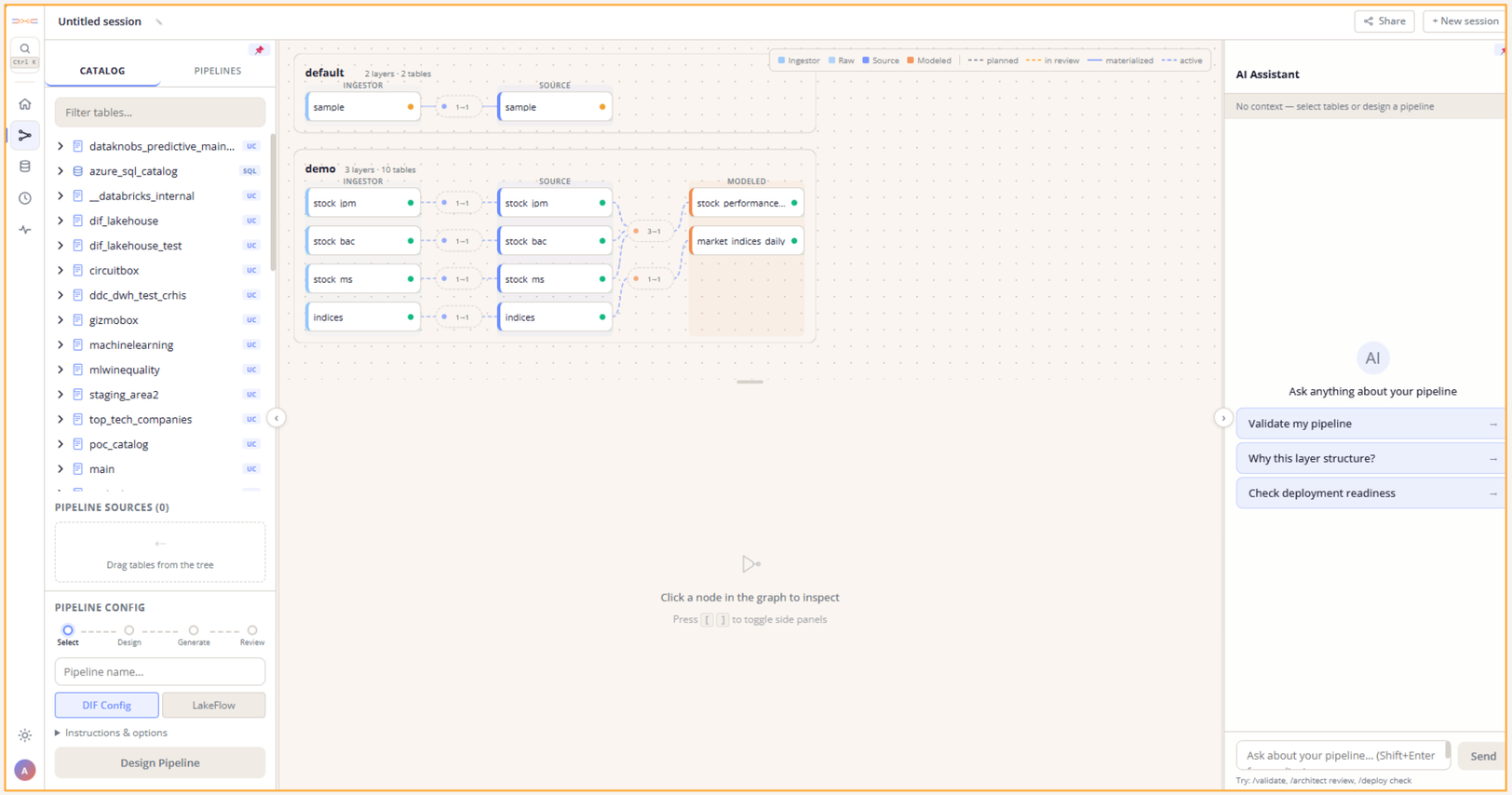
Declarative pipelines templated from metadata

- Emits onboarding files (YAML/JSON) for the DLT-meta framework
- Bronze/silver onboarding flows from one template
- Spark Declarative Pipelines run on Databricks Lakeflow
- Onboarding 1,000s of tables with one metadata commit
- Schema evolution and CDC handled declaratively
- Quality expectations injected from the metadata model
- Built on Databricks Labs DLT-meta open-source project
- Ideal for high-volume migration and ingestion-heavy estates

RICH UI · DESIGN REVIEW

Browse, edit, and approve generated artefacts before they hit production

The AI DIF UI surfaces every agent decision — humans review, accept, or steer.



WHAT THE USER SEES

Design review

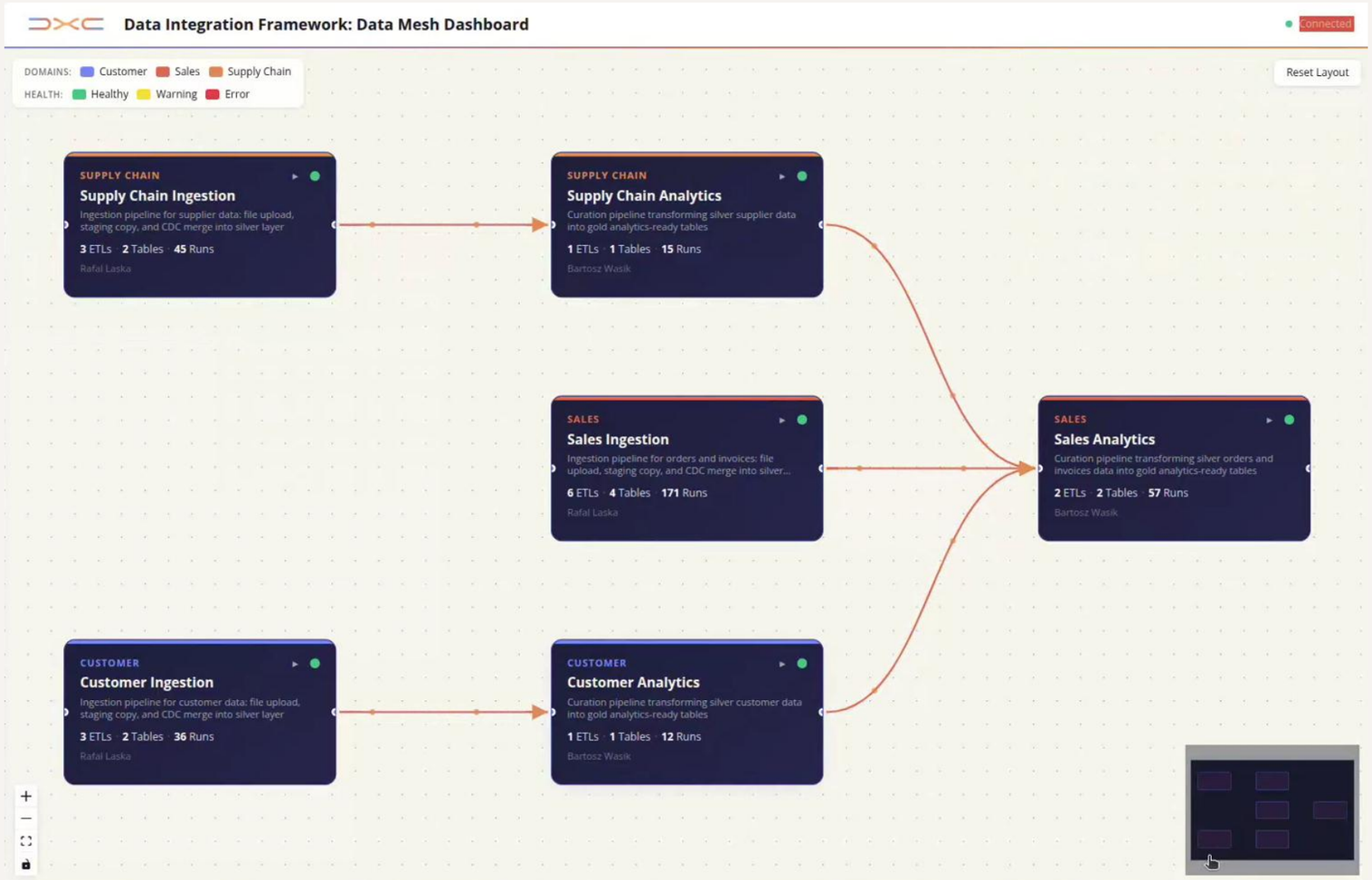
Every agent proposal — schema, model, code, deploy plan — is shown side-by-side with the current state. Reviewers approve, edit, or push back.

The reviewer never has to write boilerplate — only judge it.

DATA MESH · PRODUCT LEVEL

Top-down view of every data product on the platform

AI DIF natively supports data mesh: products are first-class citizens with owners, domains, and dependencies.



PRODUCT-LEVEL NAVIGATION

Mesh top view

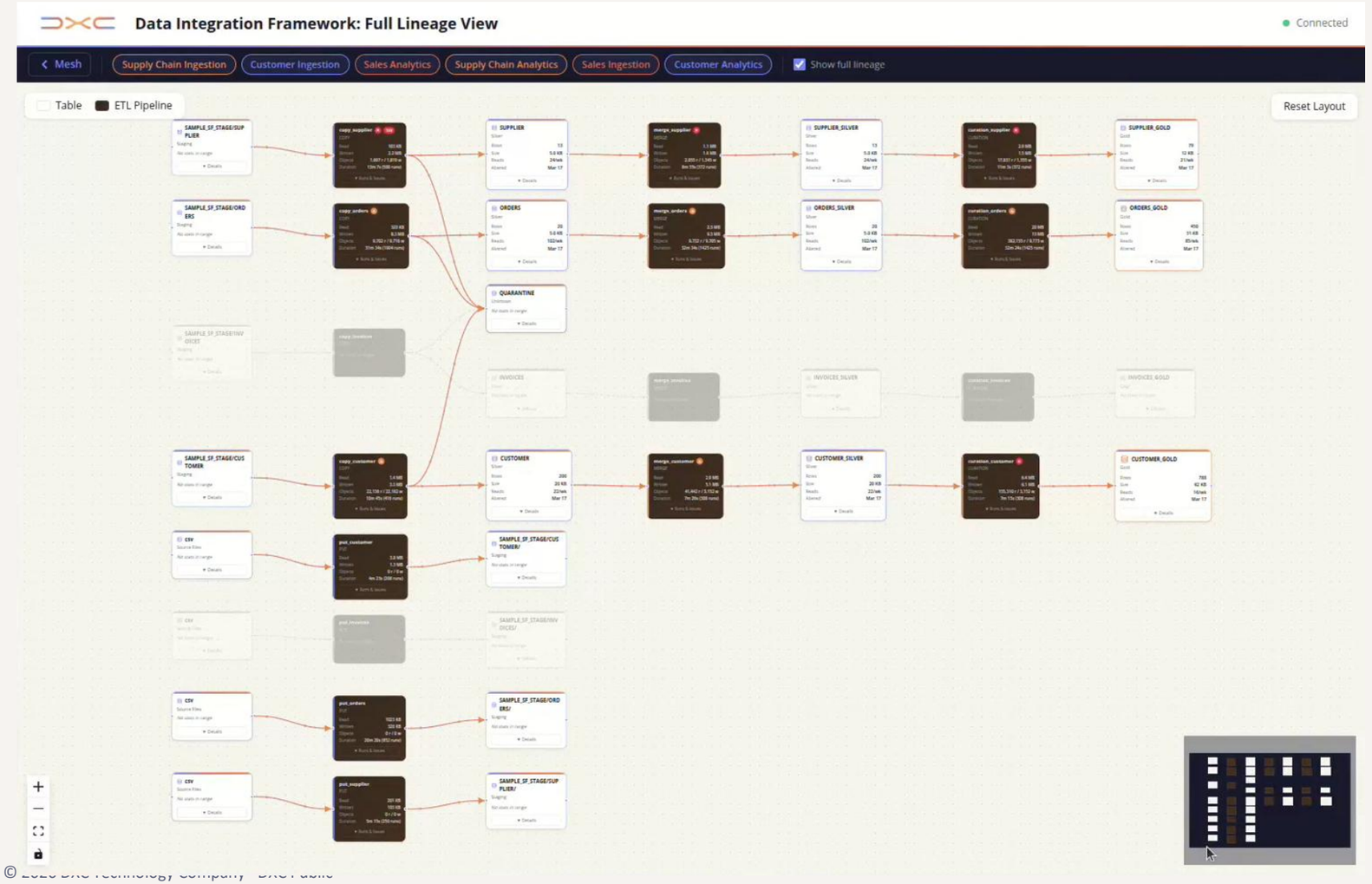
Every data product is registered in the metadata core with: domain, owner, tags, dependencies on upstream products, and downstream consumers.

From this top-level view a reviewer can see the full landscape — and double-click into any product to inspect its ETLs, tables, and run history.

DATA MESH · ETL LEVEL

Double-click into a product to see every ETL, every table, every run

Two levels of detail in one navigation model — product overview, ETL deep dive.



ETL-LEVEL DEEP DIVE

Inside a product

Each product unfolds into the ETLs it contains: source-to-target mappings, generated code, deployed workflow status, KPIs, and the most recent runs.

Lineage is preserved across the mesh — an upstream change ripples to every dependent product, with impact assessment baked in.

END-TO-END AUTOMATION

From source to running pipeline — fully agentic, fully governed

Six stages, fully automated, with human checkpoints at design-review and deployment.

1
Connect
Register external sources in Unity Catalog / Snowflake / dbt sources.

2
Discover
Modeller agent scans schemas via MCP; flags PII; proposes data products.

3
Generate
Code-gen emits target-specific configs, models, SPs, wheels, or DLT-meta YAML.

4
Deploy tables
Deployer agent creates tables, applies RLS, embeds configs in Workflows / Tasks.

5
Deploy workflows
Dependency DAG resolved; jobs scheduled or run on demand; one-shot, idempotent.

6
Run & observe
Wrapper notebooks/SPs execute SQL; KPIs and quality metrics flow back to ABC_METADATA.

WHAT'S TRUE AT EVERY STAGE

Every artefact is reviewable in Git · every agent decision is logged · every change is reversible. Configs are embedded at deploy time — no runtime config reads. Each stage's output is the next stage's input, so the pipeline is fully replayable.

GOVERNANCE & OBSERVABILITY

Built into the metadata core — not bolted on later

Compliance, security, and operability come for free — they are properties of the metadata model.

DETECT

PII detection during code-gen

Modeller agent flags PII columns when scanning sources; tags applied automatically; masking policies generated for target platform.

ENFORCE

Row-level security at table creation

RLS rules generated alongside DDL; Snowflake row access policies, Unity Catalog row filters, dbt row-level filters.

TRACE

Cross-platform lineage

Object- and column-level lineage stitched across Snowflake, Databricks, dbt, and DLT-meta via shared metadata identity.

MEASURE

Run-level observability

Run registry, quality results, reconciliation, and KPIs in ABC_METADATA — single dashboard regardless of platform.

ALERT

Alerting & incident routing

One alert across all variants — routes to Microsoft Teams, email, Azure Event Grid; ServiceNow incidents created with proposed fixes.

OPTIMISE

FinOps from query history

Per-pipeline cost attribution from query history and warehouse metering; expensive jobs surfaced for optimisation.

IMPOSSIBLE. DELIVERED.