

Using Kedro to Power Reproducible Biomedical Knowledge Graphs for Agentic Systems

How Harvard's Zitnik Lab built OptimusKG on Kedro

Executive Summary

The Zitnik Lab, a research group at Harvard University, built OptimusKG, an open biomedical knowledge graph, on a reproducible Kedro-based data infrastructure that unifies ontologies, databases, drug resources, gene annotations, papers, and lab-specific artifacts into a single ontology-grounded graph.

The team treats their data infrastructure as a first-class engineering asset: every source, schema, and transformation is explicitly tracked, versioned, and reproducible, reflecting the commitment to auditability that biomedical research demands. Kedro's data catalog, custom dataset implementations, Kedro hooks, and a medallion architecture made that possible — resulting in a more trustworthy, observable, and portable knowledge layer.

 OptimusKG

 kedro

At a Glance

Team

[Zitnik Lab](#), a research group at Harvard University

Tool Integrated

[Kedro](#) (LF AI & Data)

Platform

[OptimusKG](#), an open biomedical knowledge graph developed by [Lucas Vittor](#) and the Zitnik Lab

Use Case

Reproducible knowledge graph infrastructure for agentic AI

Intended Users

Researchers and institutions building retrieval systems, scientific copilots, and agents

Source Formats Unified

APIs, FTP downloads, SQL dumps, OWL ontologies, ZIP archives, CSV, JSON, Parquet

Architecture

Medallion: Landing → Bronze → Silver → Gold

Transformation Engine

Polars DataFrames

Ontology Validation

BioCypher

Execution Modes

Sequential, parallel, and threaded, local or cloud

Distribution

Source on GitHub, Python client on PyPI, dataset on Harvard Dataverse

The Challenge: Biomedical Knowledge Without a Common Interface

The information an AI agent needs to answer a biomedical question is rarely in one place. Biomedical knowledge is scattered across ontologies, databases, drug resources, gene annotations, papers, and custom artifacts produced by individual labs, and each source carries its own identifiers, schemas, and release cycles.

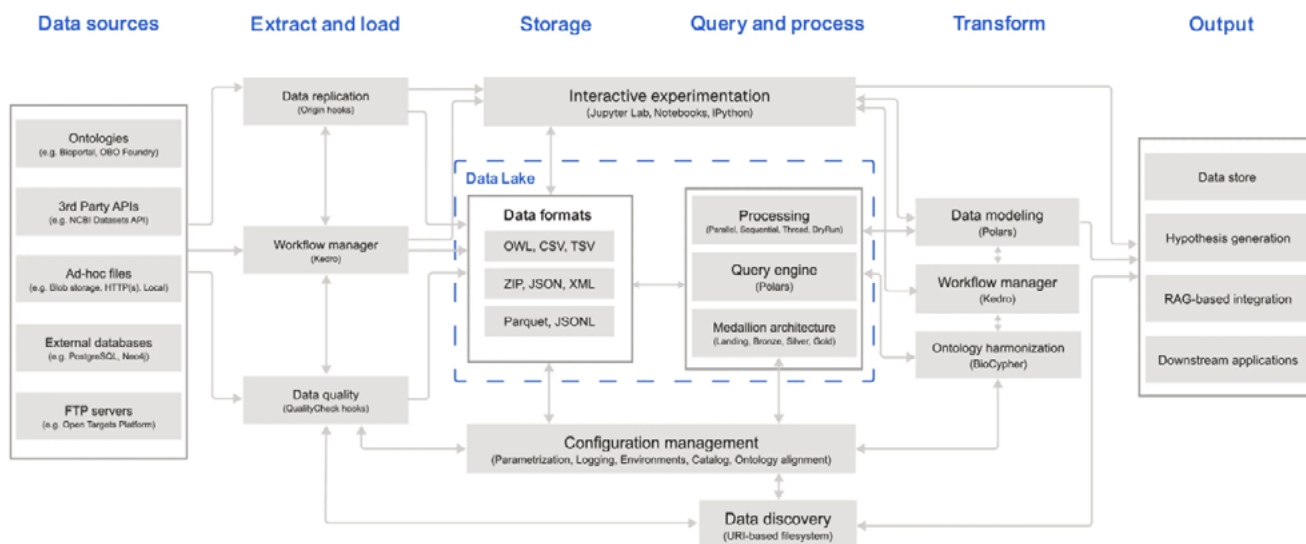
This is a core bottleneck for agents in science. They need to reconstruct what each entity means, trace where it came from, and reconcile conflicting evidence from different sources and time points. Agentic data infrastructure has to give agents predictable interfaces so that tool calls return consistent outputs. This means data cannot be treated as a preprocessing step that happens before the agent runs: it has to be a component of the system itself.

The Solution: Kedro as the Data Infrastructure Layer

The core architectural design choice behind OptimusKG was to build infrastructure that reliably recreates biomedical graphs, rather than producing a single static biomedical graph. That required a modular, provenance-aware pipeline that could orchestrate many different data sources while keeping every rebuild deterministic and auditable.

OptimusKG addressed this architecture by building a reproducible, Kedro-based biomedical data pipeline that unifies heterogeneous structured and semi-structured resources into an ontology-grounded knowledge graph for AI agents.

Kedro's [data catalog](#) gave the team the primitives to support that architecture, functioning as the source of truth for every dataset: where it comes from, how it is stored, which schema it follows, and what metadata it carries. The team extended Kedro's dataset abstraction with custom implementations for Parquet, JSON, ZIP, SQL dumps, XML, OWL, and CSV, isolating source-specific loading logic so downstream transformations could operate on a consistent Polars DataFrame interface regardless of source format.



OptimusKG also implements Kedro [Hooks](#) to handle the cross-cutting concerns that would otherwise be duplicated across nodes. The Origin Hook manages provider resolution and data downloads, the Checksum Hook detects when an upstream source has changed between runs, and the Quality Checks Hook validates naming conventions and relation values before data moves to the next stage. Configuration management separates environment settings from transformation logic, meaning the same build runs locally, in the cloud, sequentially, or in parallel without modifying code.

The data infrastructure is organized as a [medallion architecture](#) with Landing, Bronze, Silver, and Gold layers, so raw inputs are progressively normalized, aligned, harmonized, and exported as typed Parquet node and edge tables with traceable provenance. The result is a more trustworthy, observable and portable knowledge layer for retrieval systems and scientific copilots.

Landing	Bronze	Silver	Gold
<i>Kedro-managed replication</i> Heterogeneous sources ingested in their original formats; Origin Hook resolves providers and downloads or placeholders missing data.	<i>Normalization</i> Raw files converted into normalized, machine-readable columnar tables using Polars.	<i>Harmonization</i> Schema alignment, entity harmonization, and cross-source joins construct an intermediate graph representation.	<i>Export</i> Final graph exported as Apache Parquet: separate node and edge tables with typed, nested properties.

Configuration and parametrization (filters, source-specific constants, ontology settings, logging, catalog entries, and environment-specific paths) are kept separate from transformation code. The same graph build runs locally or in the cloud, sequentially or in parallel, without changing a line of code.

The Role of Open Source

Open source is central to why OptimusKG's infrastructure is trustworthy and replicable rather than a one-off internal build.

- Kedro is a mature, open source pipeline framework, originally developed at QuantumBlack and now a project of the LF AI & Data Foundation, bringing software engineering practices like modularity, reproducibility, and separation of concerns to data and AI pipelines.
- The Gold-layer output is exported as standard Apache Parquet node and edge tables, so it is not locked to any single graph database or retrieval framework.
- Every part of the stack is open: the pipeline source code is on GitHub, a Python client is published on PyPI for programmatic access, and the dataset itself is distributed via Harvard Dataverse.

Lessons Learned

- Data cannot be treated as a preprocessing step before the agent runs. It has to be designed as part of the system itself, subject to the same engineering discipline: versioning, testability, reproducibility, and clear interfaces.
- Establishing the pipeline framework early made the rest of the work tractable. The dataset interface gave the team a clean extension point for new source formats, without changing shared code. The hook system gave them a place to add cross-cutting checks without scattering logic across nodes. Both patterns worked because the framework provided stable abstractions to build against.
- The medallion architecture, combined with the data catalog, also proved to be an effective model for biomedical data. Because provenance and schema are declared at the catalog level, every intermediate layer remains traceable back to its source. In biomedical work, knowing which version of a database produced a given entity is part of what makes the result auditable.

Next Steps

OptimusKG is positioned as a foundation rather than a finished product. The Zitnik Lab, a research group at Harvard University, continues to develop the project as part of a broader effort to build open, reproducible infrastructure for AI, making biomedical knowledge more accessible to researchers and institutions building retrieval systems, scientific copilots, and agentic applications. The team has invited feedback, contributions, and collaboration from the community, framing reliable biomedical knowledge infrastructure as a collective effort.

Get Started with OptimusKG and Kedro

Explore the OptimusKG documentation, read the research paper describing the methodology, or install the Python client to work with the graph programmatically. Kedro itself is a project of the LF AI & Data Foundation.

GitHub: [OptimusKG on GitHub](#)

PyPI: [OptimusKG Python client](#)

Dataset: [Harvard Dataverse](#)

Kedro: [kedro.org](#)

Modern agentic systems are only as trustworthy as the data infrastructure beneath them — which is why data can no longer be treated as a preprocessing step that happens before the agent runs. It has to be built into the system itself, engineered with the same rigor as everything else, so every agent gets the predictable, consistent ground truth it needs to reason well.

What Is Kedro?

Kedro is an open source Python framework for structuring and authoring production-ready data and AI pipelines. It brings software engineering best practices, including modularity, reproducibility, and separation of concerns, to help teams build maintainable, scalable workflows and move from experimentation to production faster.

Originally developed at QuantumBlack to address the challenges of real-world data science projects, Kedro is now a project of the LF AI & Data Foundation, supported by a growing open source community.