PRODUCTION AI ENGINEERING

Knowledge Graph vs. Context Graph

What your enterprise **knows**— vs. what your agent **needs to decide**, right now.

HOW TO READ THIS

- Knowledge graph — every entity that exists
- Context graph — what this decision needs
- Permission gate — access checked first

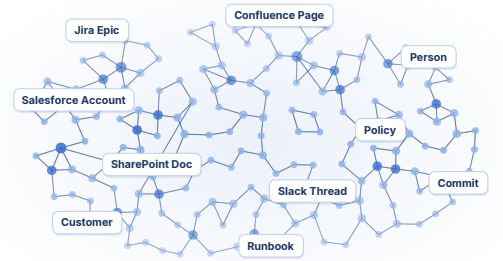
The knowledge graph knows what exists.

~2.4M nodes across every connected system. Governed, versioned, always-on enterprise memory.

THE QUESTION

"What's the status of the Q3 DataMesh migration, and who owns the remaining blockers?"

one question → one context graph



A complete graph is a haystack. The agent still has to search it.

RETRIEVAL

CONTEXT ASSEMBLY LAYER

runs per question, at request time

1 ANCHOR — Resolve the question to seed entities — entity resolution, not cosine similarity. `6 seed nodes`

You built precise structure. Then queried it probabilistically, across all of it.

2 TRAVERSE — Typed, bounded expansion along the ontology — whitelisted relations, hop limit. `1,840 in scope`

Permission-first, not filter-after.

Most stacks retrieve everything, then filter. By then the unauthorised node has already shaped the answer.

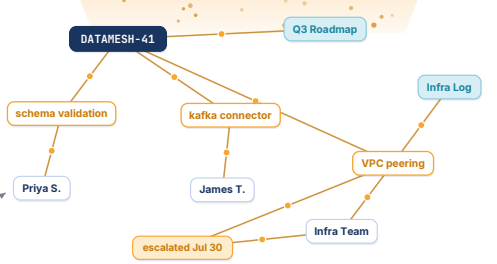
3 PERMISSION GATE — Check the asker's rights before retrieval, not after. Never retrieved = never leaked. `610 permitted`

Scopes the graph to what this decision actually needs.

4 RANK & BUDGET — Hybrid score + rerank, then prune to the token budget the model can actually use. `38 nodes`

The agent gets grounded context.

38 nodes. The facts this decision needs, plus the event traces behind them — shaped to this user's access.



Small enough to reason across, not search through.

Small subgraph → Reason → Act → Cite sources → Write trace back

KNOWLEDGE GRAPH		CONTEXT GRAPH
What is permanently true		What is true for this decision, in this moment
Entities and relationships		Relevant facts plus the decision traces behind them
Stable and versioned		Ephemeral — built and thrown away per question
Domain-scoped		Decision-scoped
Curated offline by data teams		Assembled at runtime by the agent
Governed at the store		Gated at assembly, before retrieval
Optimised for coverage		Optimised for precision and token budget
Millions of nodes		Dozens of nodes
Fails by being incomplete		Fails by being the wrong subgraph

Same question, two users → two different context graphs. Both are correct.

THE SEMANTIC SPINE

four layers — each one answers a question the layer above it cannot

LAYER 01

Ontology

Defines the types, relations and constraints. Governed by humans, changes rarely. Without it, traversal has no rules to follow.

LAYER 02

Knowledge Graph

Stable, governed enterprise memory. Every entity that exists and how it connects. Complete — and far too big to reason over.

LAYER 03

Context Graph

The right subgraph, at the right moment, for the right person. Built per question, thrown away after. This is the layer most teams skip.

LAYER 04

AI Agent

Reads the traces, decides, acts, and writes the decision back — so today's context becomes tomorrow's memory.

KAIZEN SOFTWARE SYSTEMS

Enterprise AI, LLM engineering & software products · Charlotte, NC

Meridian — permission-first enterprise knowledge AI Read the full guide

Answers with citations. Access checked before retrieval. kaizensoftwaresystems.com/blog.php