
Shipping with Context:
Knowledge Graphs as the Backbone of AI-First Software Delivery

The Context Challenge in Enterprise AI

harness

Siloed Data & Tools

Information trapped in disparate systems, inaccessible to AI.

Lack of Domain Knowledge

Generic LLMs don't understand company-specific jargon or unique processes.

Risk of Hallucinations

Ungrounded AI can confidently generate incorrect, misleading information.

Security & Privacy Risks

Sharing sensitive data with external models poses significant compliance threats.

AI-Assisted DevOps vs. AI-Operational DevOps

harness

AI-Assisted DevOps

- AI helps *humans* write code, configs, runbooks
- Context injected manually via prompts
- Human remains the primary safety system

AI-Operational DevOps

- AI acts *within* the delivery system
- Pulls context automatically from a shared model
- Performs checks, simulations, and policy evaluation before acting

With Knowledge Graphs,

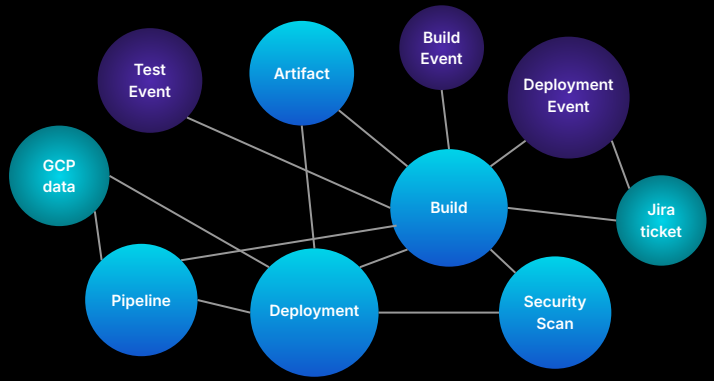
- Context becomes queryable, not conversational
- Safety moves from humans → system defaults

Anatomy of the Knowledge Graph i.e., “Context”

harness

Core Entities (The Nodes)

- Your data: Service, Pipeline, Build, Deployment, Incident, Policy, Cloud Cost, Owner, etc
- Your events: Build event, Deployment event, etc.
- 3rd-party data: GCP, GKE data

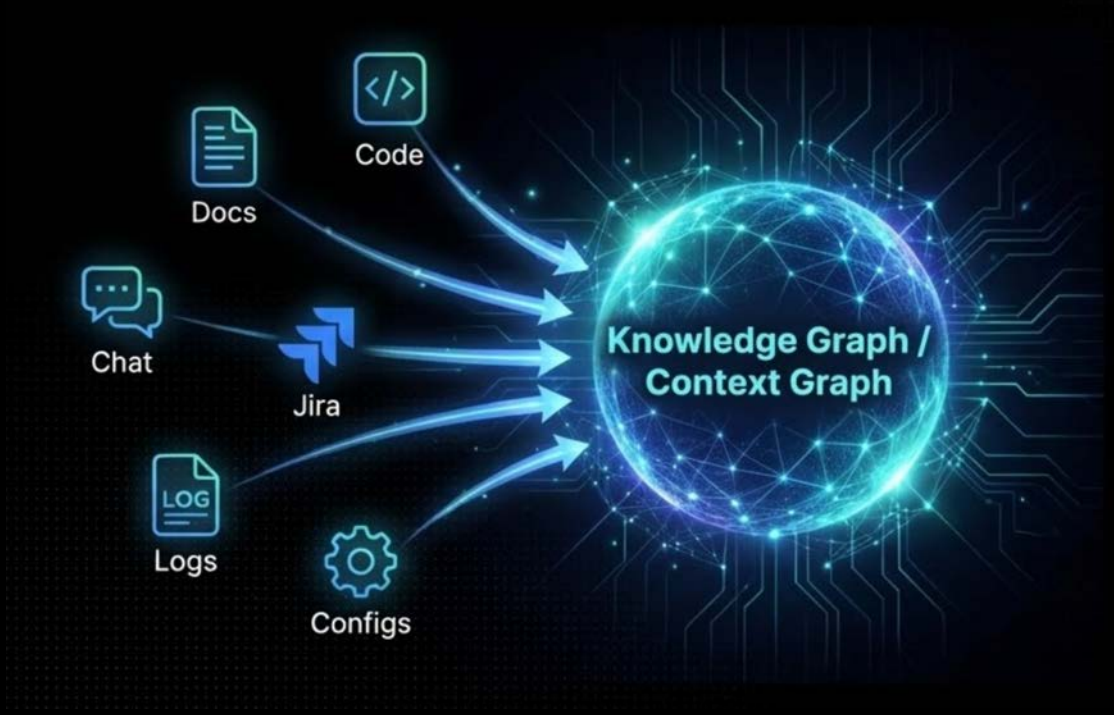


Critical Relationships (The Edges)

- Service -> Depends On -> Service (Blast radius)
- Deployment -> Triggered By -> Commit (Root cause)
- Service -> Owned By -> Team (Accountability)

Building the Knowledge Graph

harness



Extract entities
(services, users, incidents)

↓

Enrich with metadata
(owners, versions, policies)

↓

Link relationships
(who owns what, what depends on what)

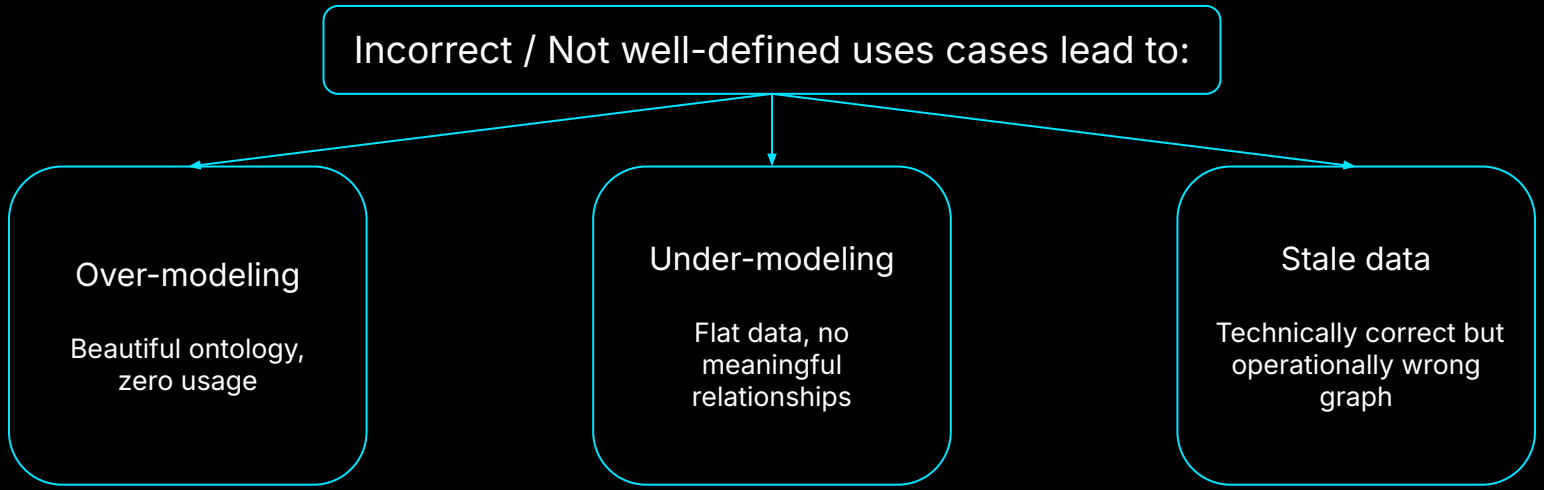
↓

Expose via Graph API for AI agents to query

If context is the new code, knowledge graph is the compiler

Where can Knowledge Graphs fail?

Primary reason for knowledge graph to fail: Incorrect use cases



Demo