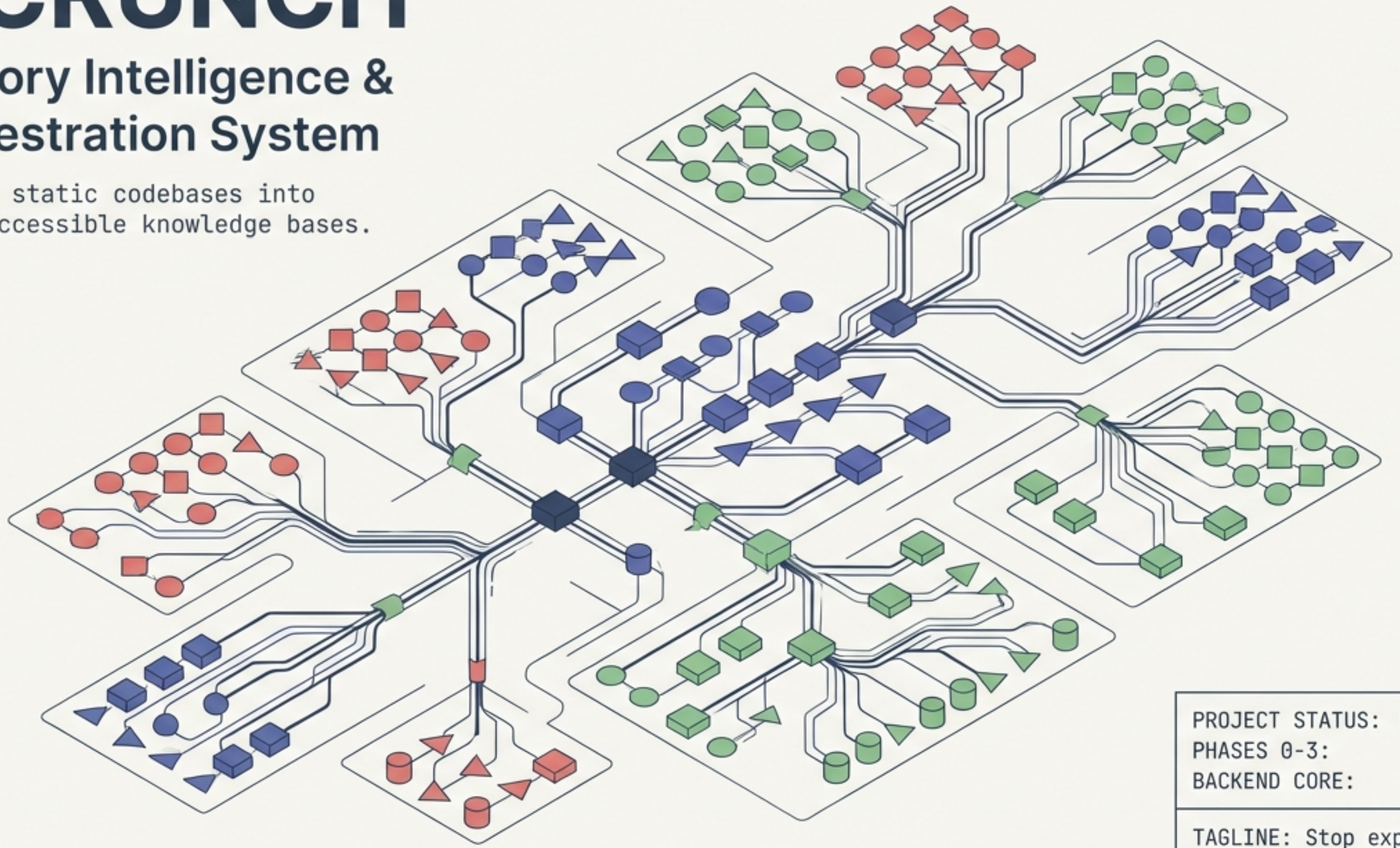
DOCRUNCH

Repository Intelligence & AI Orchestration System

Transforming static codebases into living, AI-accessible knowledge bases.

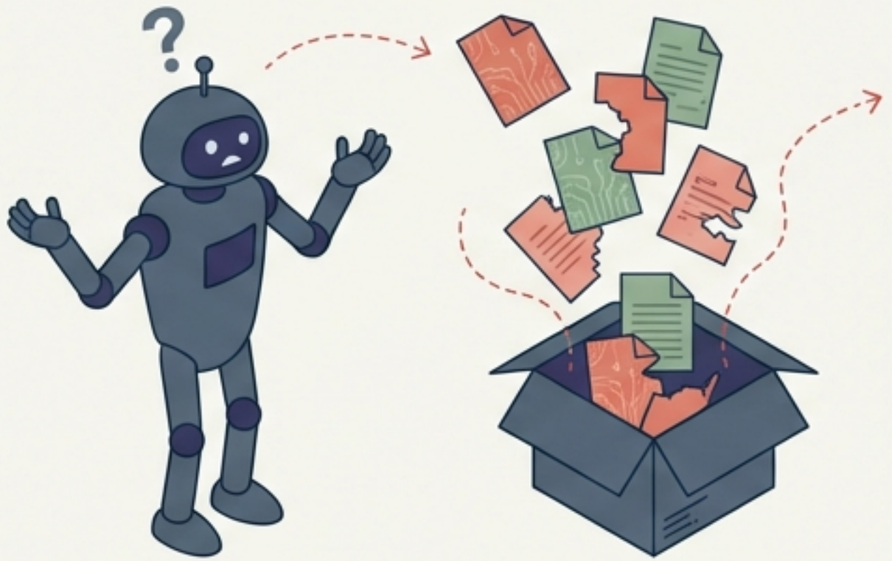


PROJECT STATUS:	MVP DELIVERED
PHASES 0-3:	COMPLETE
BACKEND CORE:	85% COMPLETE

TAGLINE: Stop explaining your architecture. Start managing it.

The Problem: The ‘Amnesiac’ Coding Agent

Without Docrunch (Ephemeral)

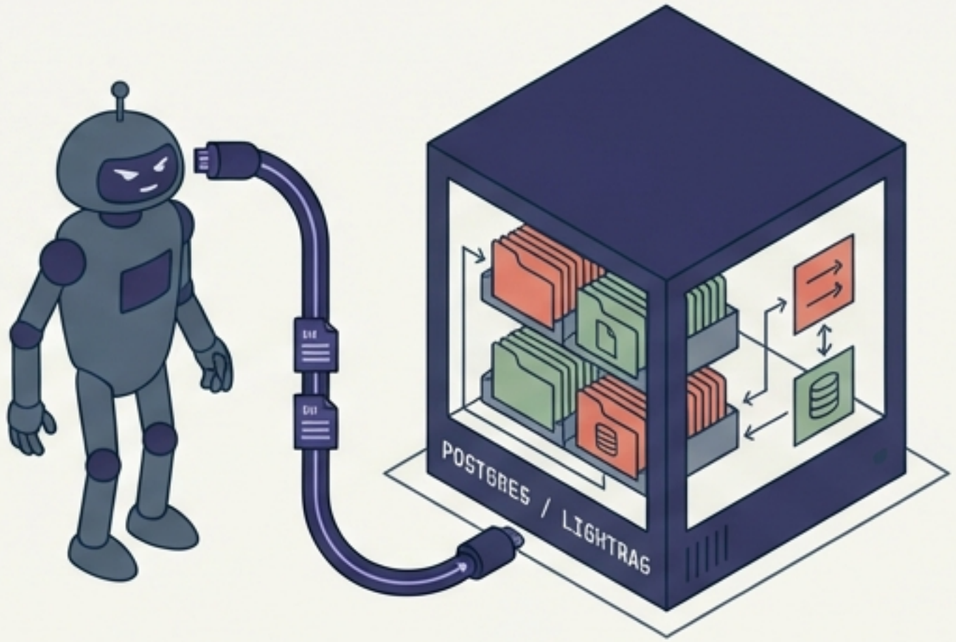


Every session is Day 1. Context is trapped in chat logs, leading to "drift" and repetition. Code becomes inconsistent with established patterns.

```
// Session 1: Auth implemented correctly
class Auth { ... }

// Session 2: Forgotten pattern
def reset_password(): # Inconsistent style
```

With Docrunch (Persistent)



Persistent memory layer. The system scans, indexes, and orchestrates context. Agents know "why" code exists.

```
// Agent queries context...
// Pattern retrieved: Use Auth class methods
Auth.reset_password()
```

A Living System Architecture

THE BRAIN
(Intelligence & Storage)

Core Engine (Scanner)

Postgres (Truth)

LightRAG (Vectors)

THE NERVOUS SYSTEM
(Connection)

MCP Server

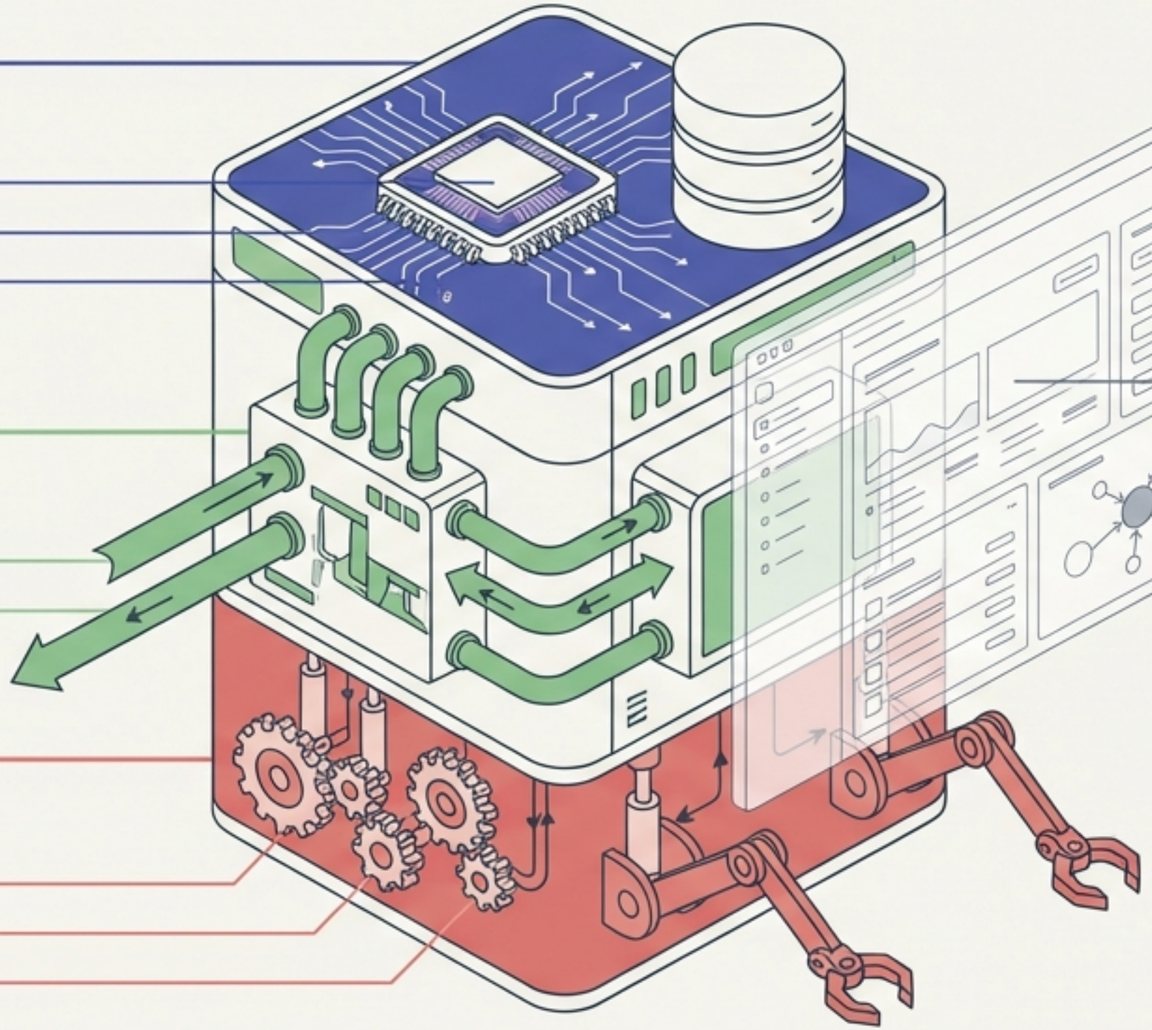
CLI Bridge

THE MUSCLES
(Orchestration)

Temporal (Workflows)

Redis (Fast Jobs)

Task System



THE FACE
(Visualization)

React Dashboard

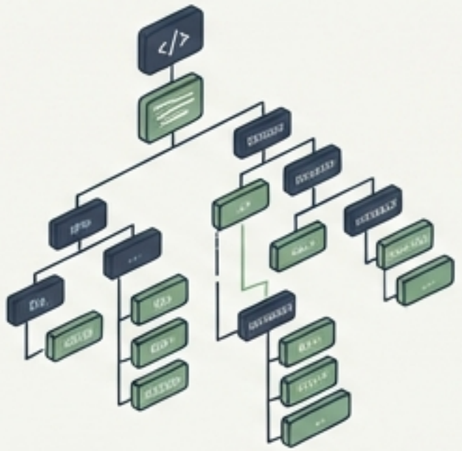
Graph Visualizers

The Brain: Polyglot Parsing & Knowledge Graph

Source Code

Python, TS, JS, TSX

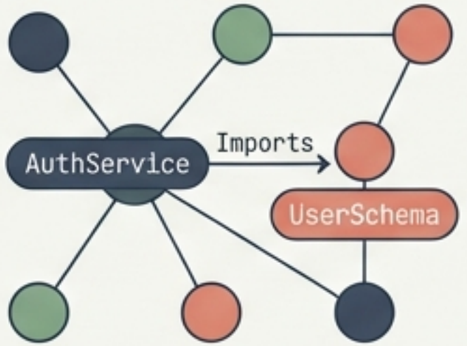
Tree-sitter Parsing



Abstract Syntax Tree (AST)

Extracts Classes, Functions, Imports.
Precise line spans, no arbitrary chunks.

Relationship Analysis

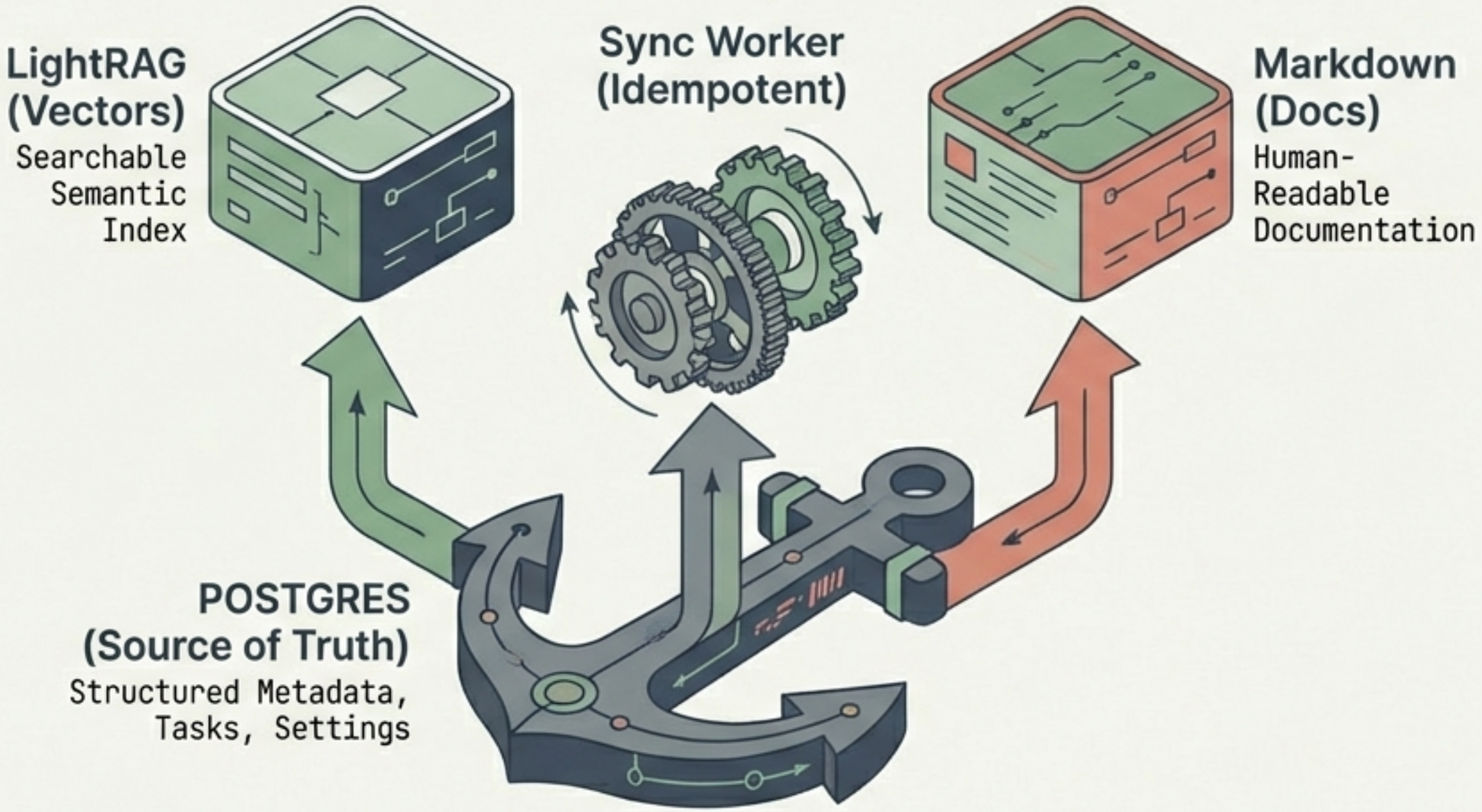


Knowledge Graph

Unlike simple RAG, Docrunch maps relationships.
Result: Precise, non-hallucinated context injection.

Task-012: Relationship Analyzer - COMPLETED

Data Consistency: The 'Source of Truth' Anchor



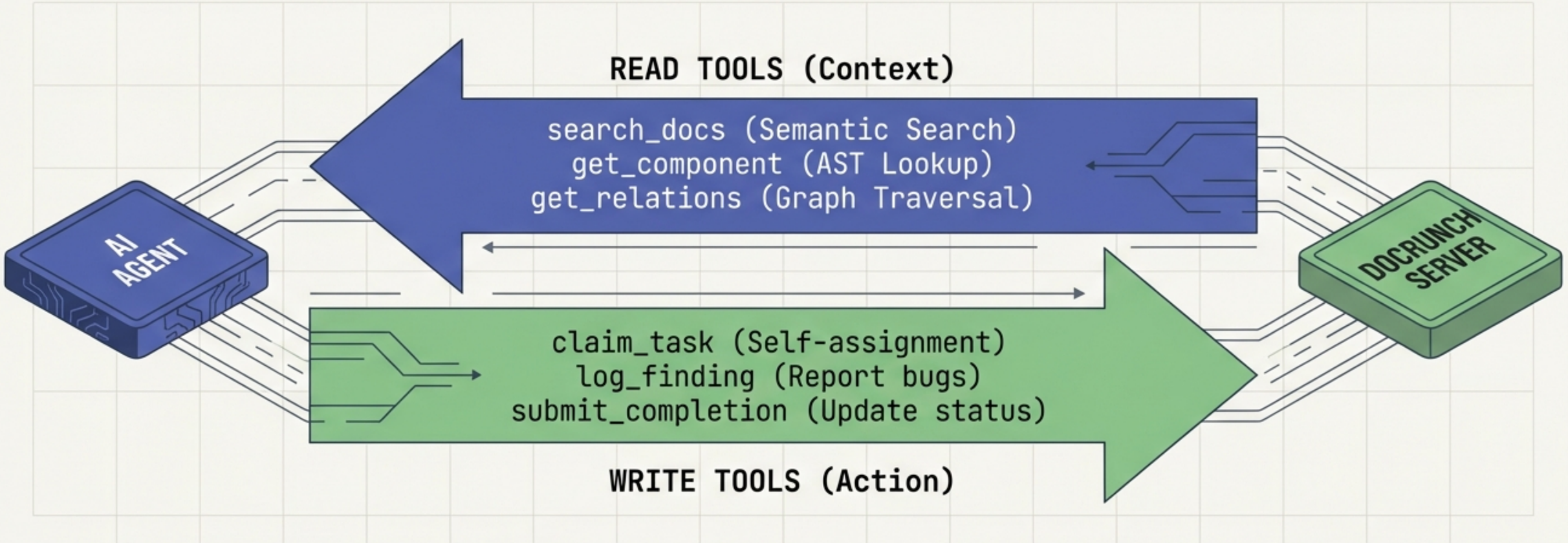
The Outbox Pattern

1. Writes to Postgres emit sync_events.
2. Workers process events to update Projections.
3. Failures in projections do not corrupt the database.

Sync Latency: Report events < 60s | Scan events < 5 min

Bidirectional MCP: The Nervous System

From Passive Reading to Active Reporting



Value: Documentation evolves AS the code changes, not after.

The CLI Bridge & Proxy

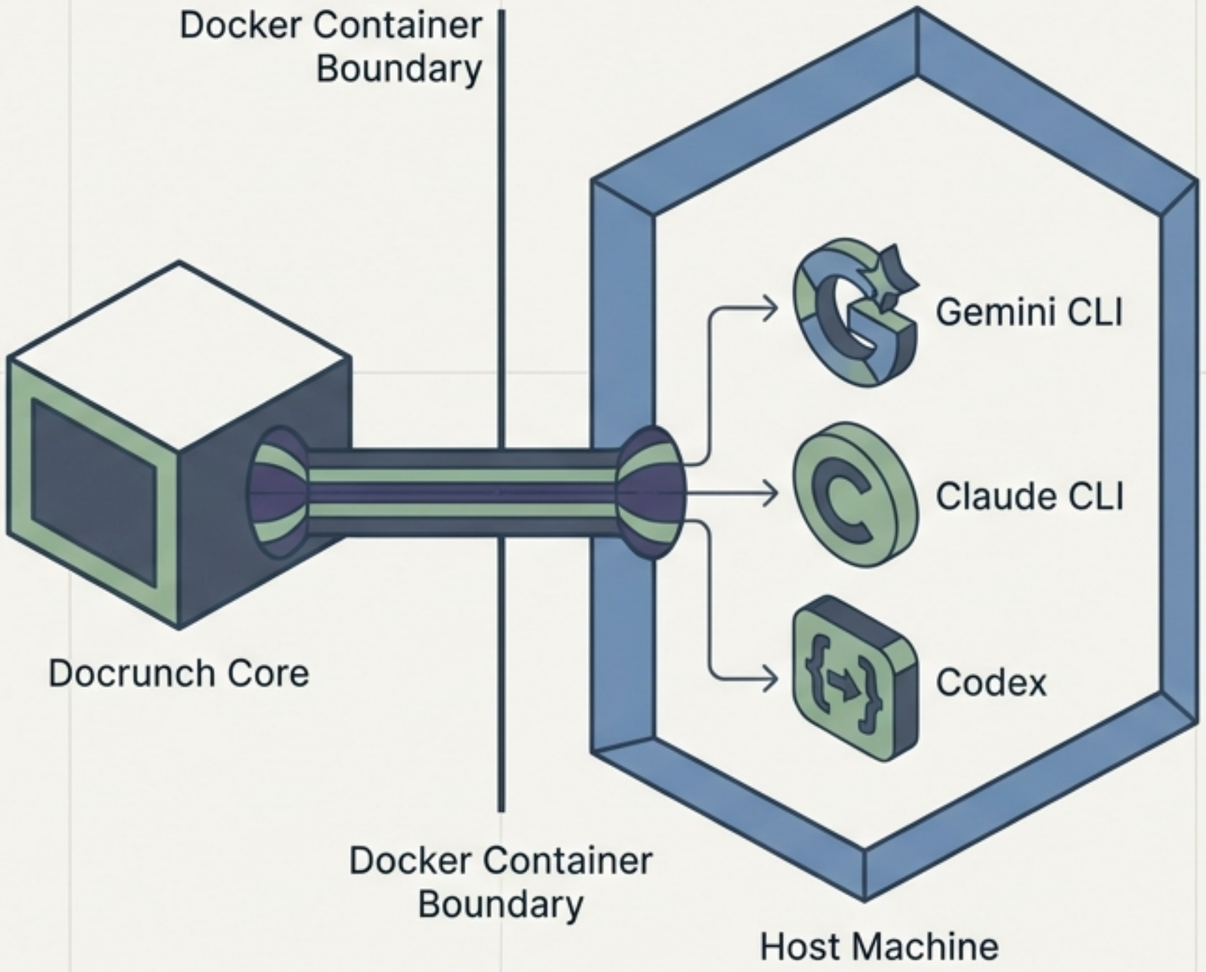
Challenge: Docrunch runs in Docker, but authenticated developer tools run on Host.

Solution: Task-010A CLI Bridge.

Mode A: Local Subprocess (Direct Execution)

Mode B: HTTP Proxy (Tunnel POST /execute)

Benefit: API keys stay local. Flexibility to use preferred tools.

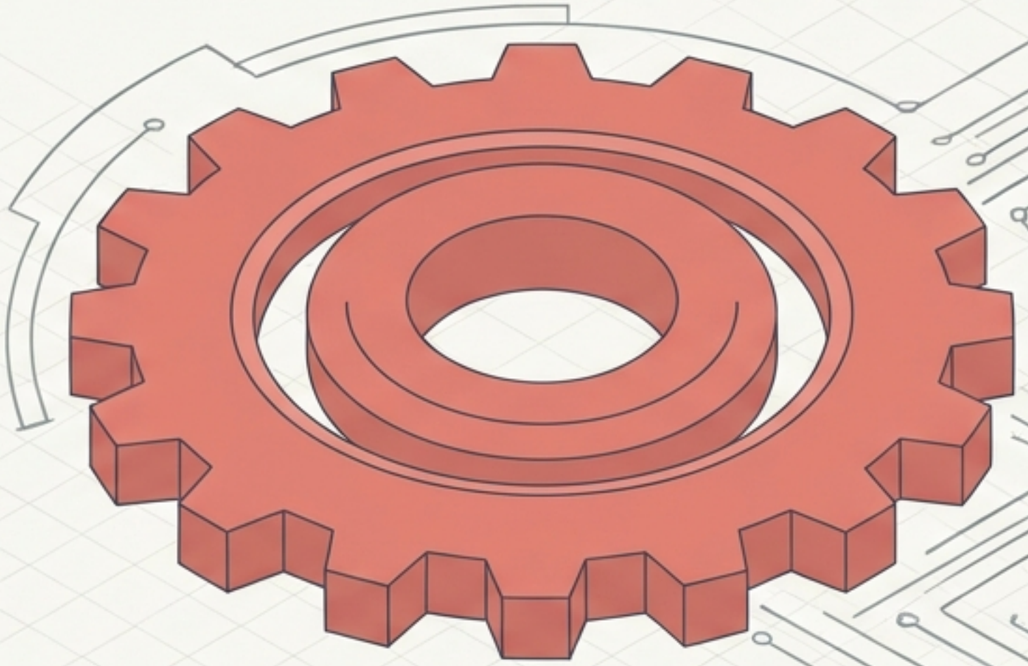


Orchestration: The Muscles

Dual-Layer Job System

REDIS + DRAMATIQ (Fast Queue)

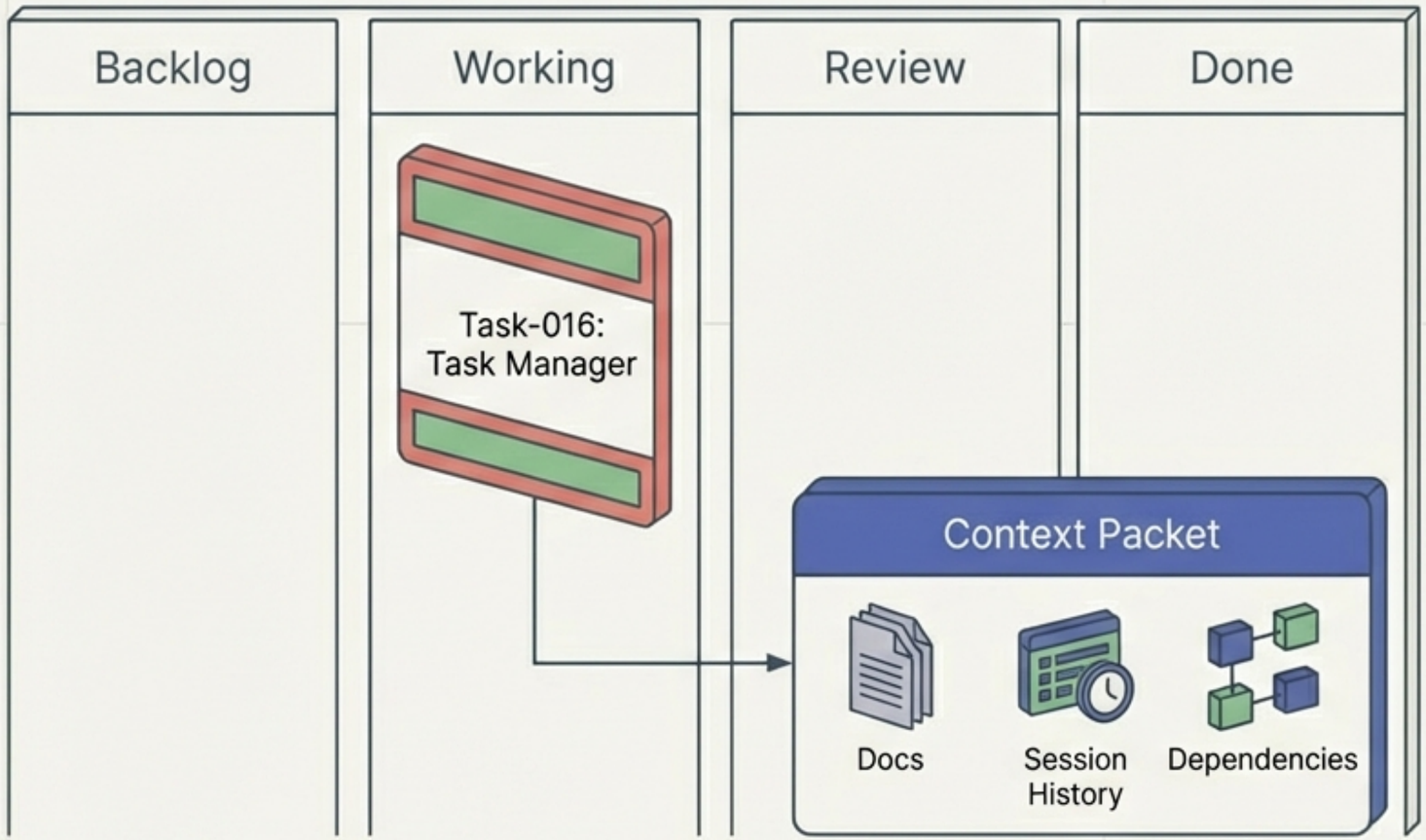
High Volume, <30s Latency
Jobs: Sync Events, File Embedding
Status: Completed (Task-005B)



TEMPORAL (Durable Queue)

Stateful Workflows, Minutes/Hours
Jobs: RepoScanWorkflow, MultiStepAnalysis
Feature: Saga Pattern (Auto-rollback on failure)
Status: Completed (Task-046)

Task Management & Context Injection

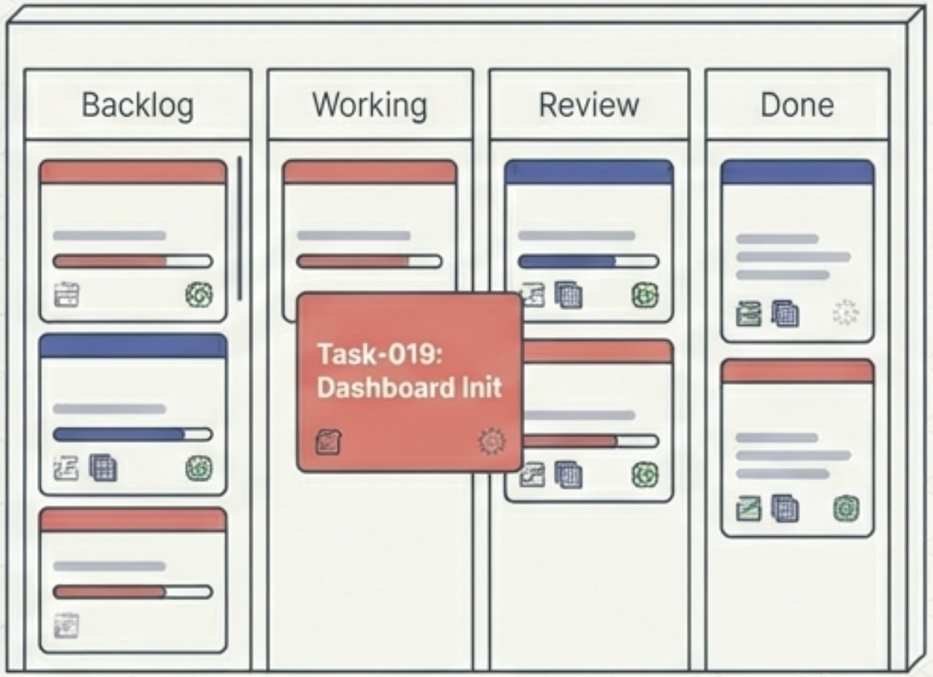


The Logic (Task-016)

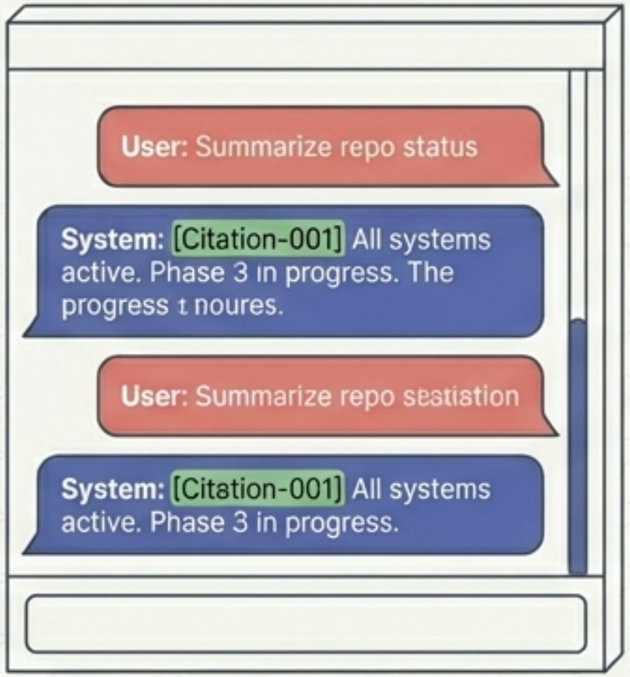
- **Context Injection**: Agents automatically receive relevant docs when claiming a task.
- **Task Types**: Feature, Bug Fix, Refactor, Security.
- **Review Logic**: Auto-approval rules vs. Human review gates.

ACTIVE / IN PROGRESS
(Assigned to Gemini Sub-Agent)

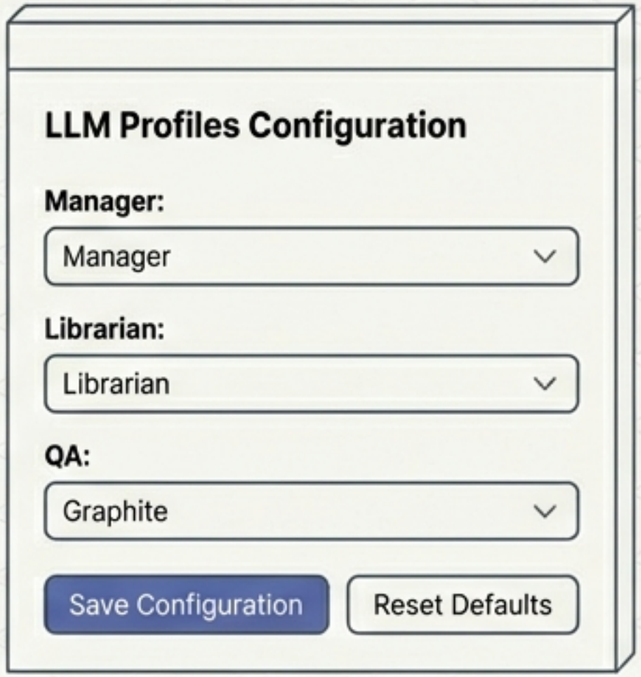
The Face: Visualization Dashboard



Kanban Board



Chat Interface



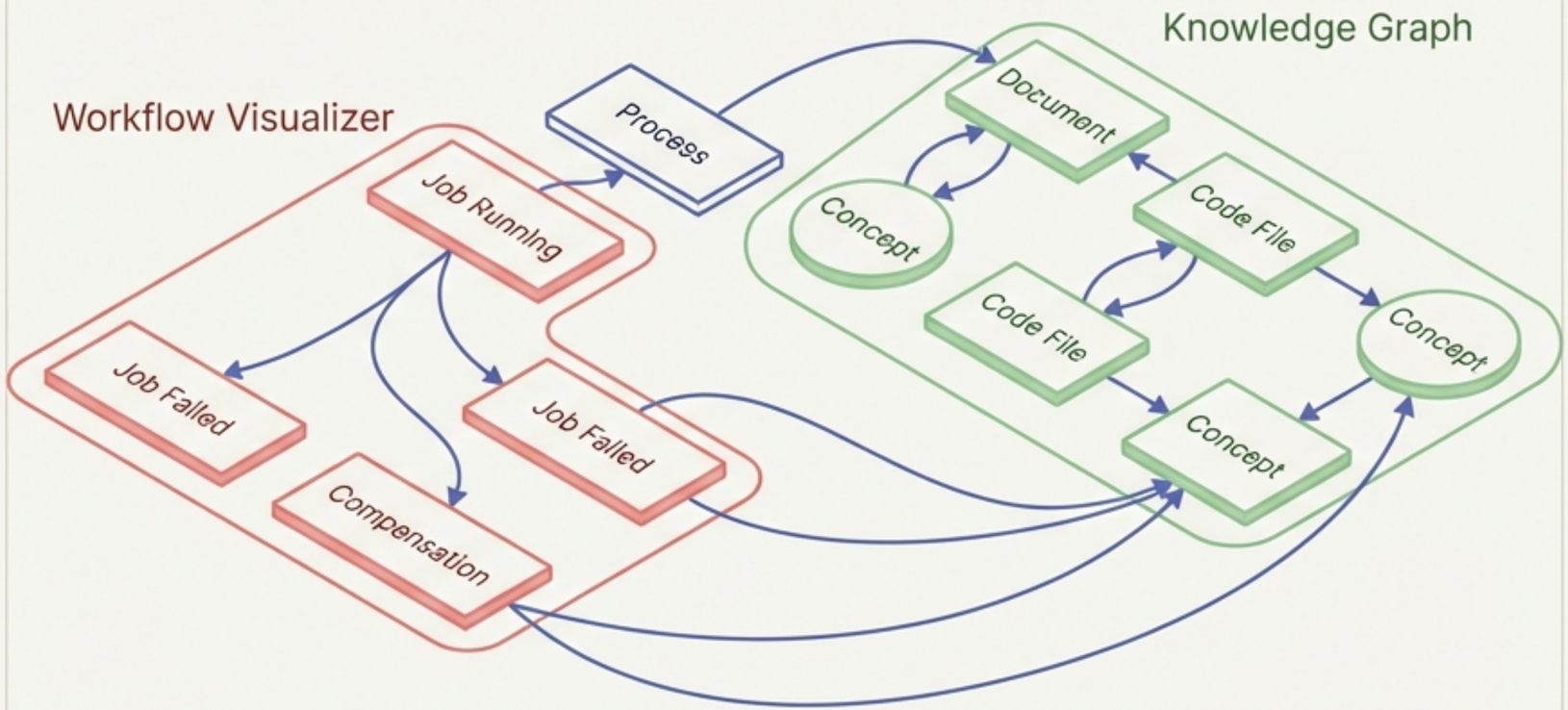
Settings / LLM Profiles

Tech Stack: React 18, Vite, TanStack Query.
Real-time: WebSocket streams (/ws) for lineage, sync events, and query traces.
Status: Phase 3 Completed (Task-019)

Graph Visualization Engine

React Flow Implementation

- Interactive exploration with Dagre auto-layout.
- Drill-down capabilities into specifc entities.
- Task-066: Core infrastructure delivered.



Development Scorecard

Phases 0-3 Complete | ~85% Backend Core

Phase 0-1 (Foundation) - COMPLETE

- Tooling, Postgres Schema, Async Scanner, Sync System.

Phase 2 (Intelligence) - COMPLETE

- Tree-sitter Parsers, CLI Bridge, LightRAG, Temporal Workflows.

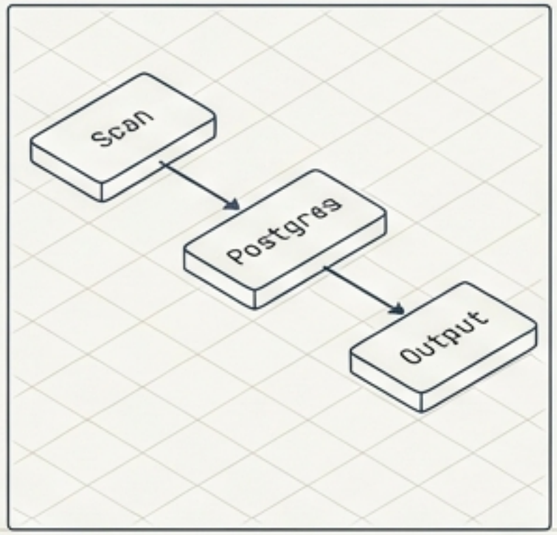
Phase 3 (MCP + Tasks) - COMPLETE

- MCP Server, Report Tools, Dashboard Scaffold, Chat.

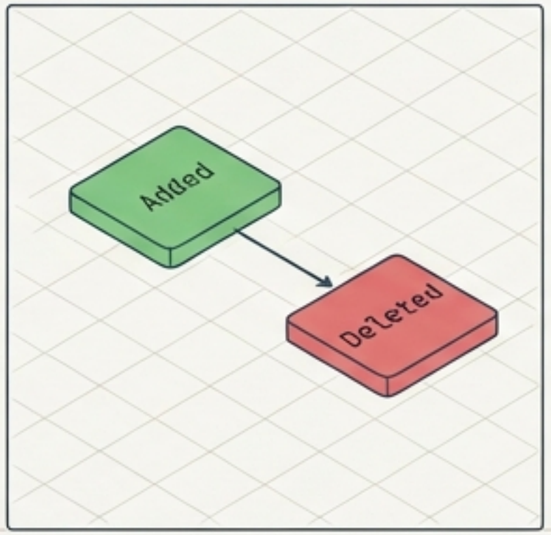
32 Tasks Completed.

Evidence: 'Dogfooding' complete. Docrunch documented itself (Task-008A).

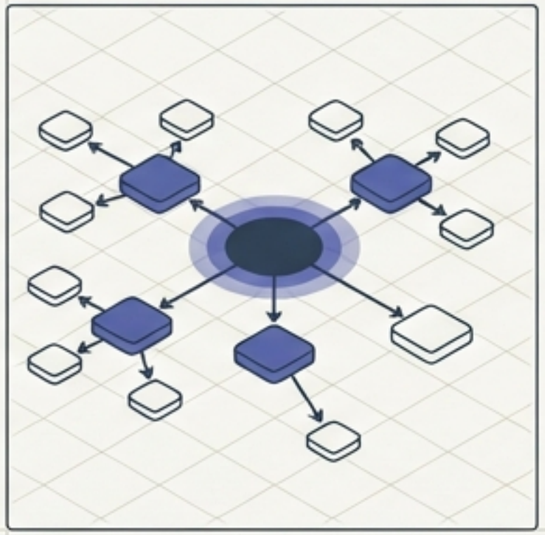
Current Focus: Phase 4 Advanced Visualization



Data Lineage (Task-088)



Diff Visualization



Impact Analysis

- **Active Queue**: 33 Tasks.
- **Collaborative Cursors**: Multi-user exploration.
- **Repository Graph**: Entity expansion (tables, components).

Roadmap: Phase 5 Advanced Features

Brain/Intelligence

Session Memory

Cross-session persistent context.

Nervous System/Transport

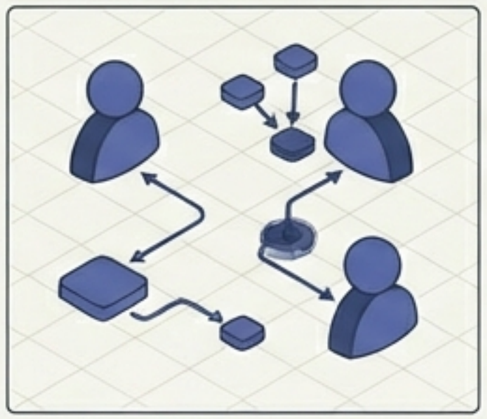
Git Integration

Auto-commits & PR summaries.

Muscles/Action

Cost Tracking

Token budget management.

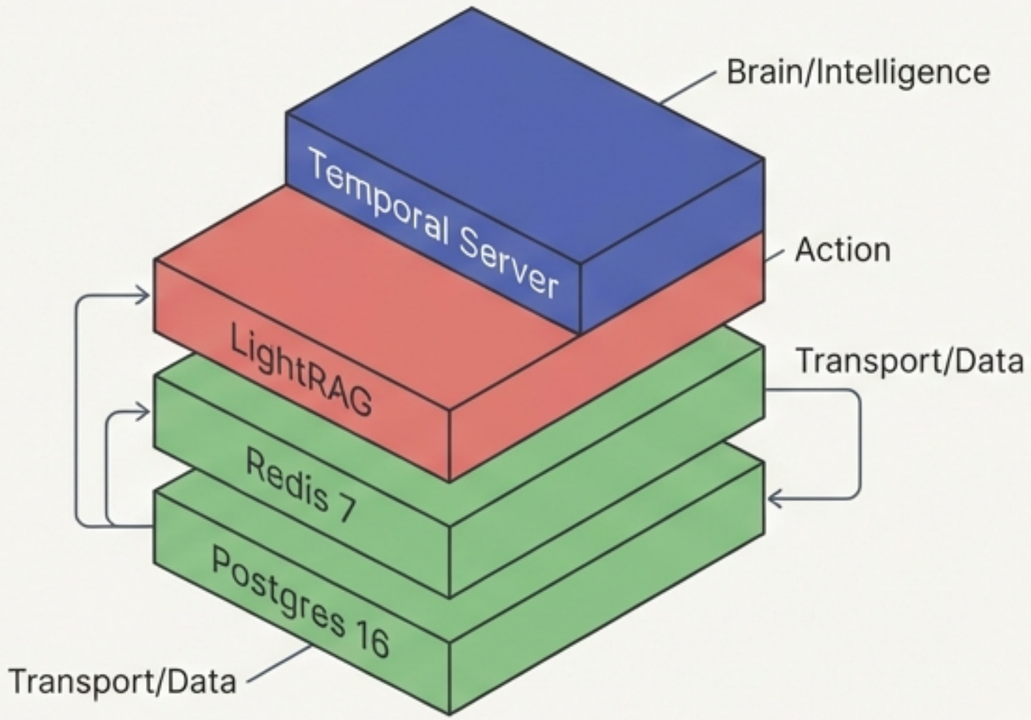


Multi-Model “Party Mode”

Parallel agent collaboration (e.g. Claude on Frontend, Gemini on Backend).

Deployment & Resilience

Docker Compose Stack




```
$ docrunch scan --offline
> Bypass Postgres/LLM... OK
> Reading local JSON cache... OK
> Generating Markdown docs... DONE
```

Summary: Enterprise-grade durability. Developer-friendly flexibility.