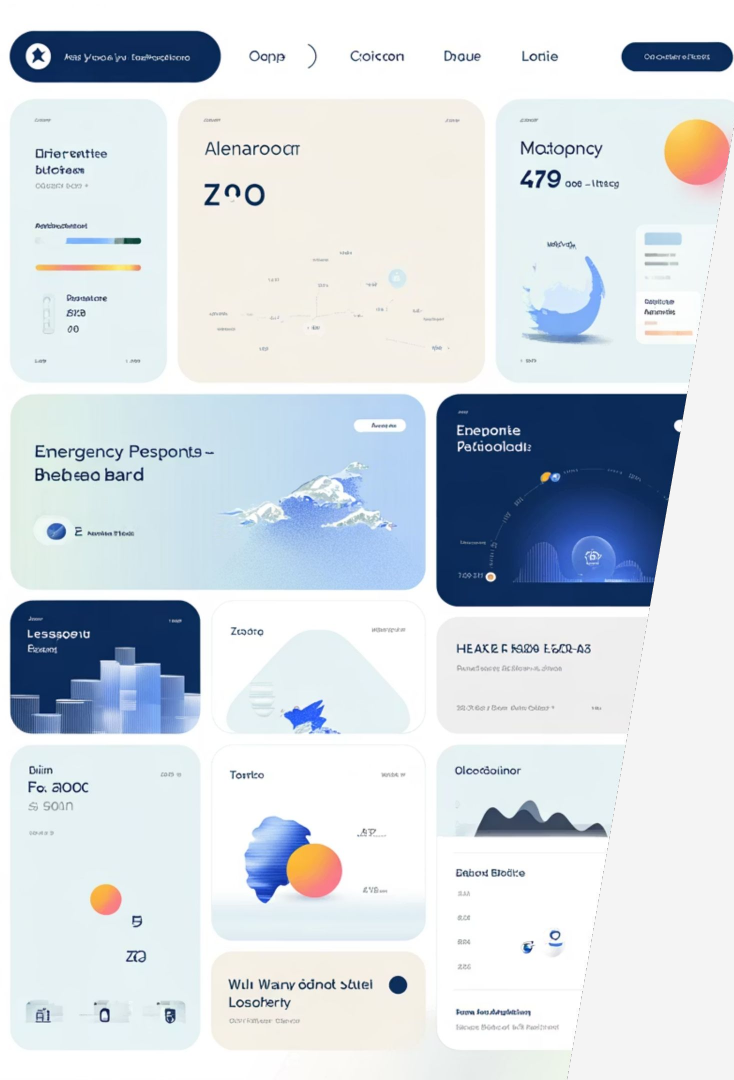# 🚨 Traceback: Data Pipeline Incident Triage System

AI-Powered Incident Response with Advanced Retrieval Methods

Executive Summary

30-60

Minutes Before

Traditional incident response time

<5

### Minutes After

Traceback response time

3

Data Sources

Unified search across documentation, code, and lineage

Traceback provides instant access to business impact analysis, blast radius assessment, and recommended actions through intelligent search across fragmented data sources.

🎯 The Problem

On-call data engineers can't quickly determine business impact during production incidents because requirements live in fragmented sources not built for fast triage.
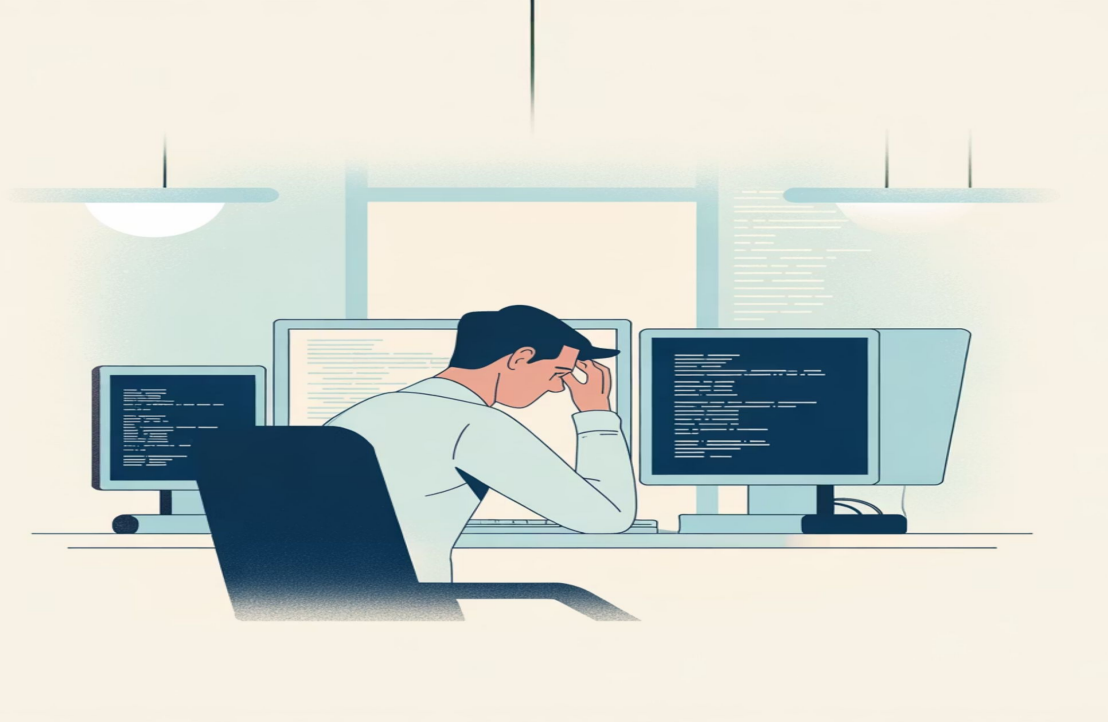
For Data Engineers & SREs

Engineers spend 30-60 minutes chasing answers across disconnected systems, manually correlating Confluence docs, Git repos, and data catalogs. This leads to inflated MTTR, SLA breaches, and risky hotfixes.

For Analytics Leads & Product Managers

Cascading delays create unclear updates about data trust and availability, resulting in paused reports, broken trust in data quality, and delayed business decisions.

🛠️ The Solution

Traceback is an agentic RAG application that unifies three fragmented data sources to power fast, context-rich incident response.

Requirements Docs

PDF/Markdown specifications with SLAs and ownership

Pipeline Code

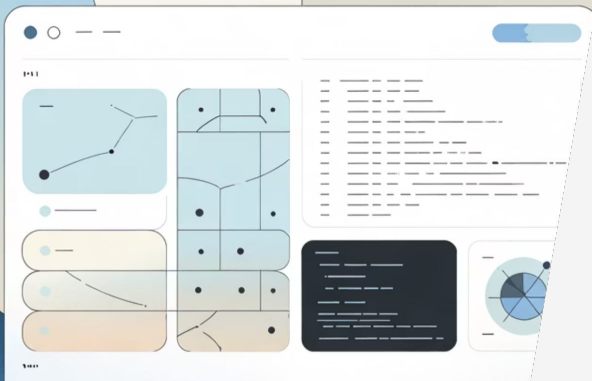SQL/Python transformations and configurations

Lineage Graph

Column-level dependency mapping

> *"Job curated.sales_orders failed — who's impacted?"*

Traceback returns structured incident briefs with impact summaries, blast radius analysis, and recommended actions.

Technology Stack

OpenAI GPT–4o–mini
Low latency with high reasoning capabilities for incident analysis

LangGraph
Agent loops and intelligent tool routing for complex workflows

Qdrant Vector Store
Portable, zero-infrastructure setup with fast similarity search

FastAPI
High-performance API with automatic OpenAPI documentation

LangSmith
Comprehensive logging, tracing, and prompt capture

RAGAS
Industry-standard metrics for RAG system assessment

Agentic Architecture

Supervisor Agent
Routes questions to appropriate tools

Impact Assessor
Performs RAG and queries lineage graphs

Writer Agent
Crafts structured incident briefs with citations

📊 Data Sources & Strategy

Requirements & Runbooks

PDFs and Markdown with definitions, SLAs, ownership, and incident protocols providing authoritative context

Column Lineage Graph

JSON-based directed graph enabling precise impact assessment and dependency mapping at table/column level

Pipeline Code & Configurations

SQL, Python, and YAML files showing actual implementation and recent changes that might cause incidents

External API Integration

Tavily Search API for fresh lookups on error messages and current reference links

Chunking Strategy

Documentation

350-600 tokens with 10-15% overlap

Preserves meaning across headings and paragraphs

Code

400-700 tokens per function/statement
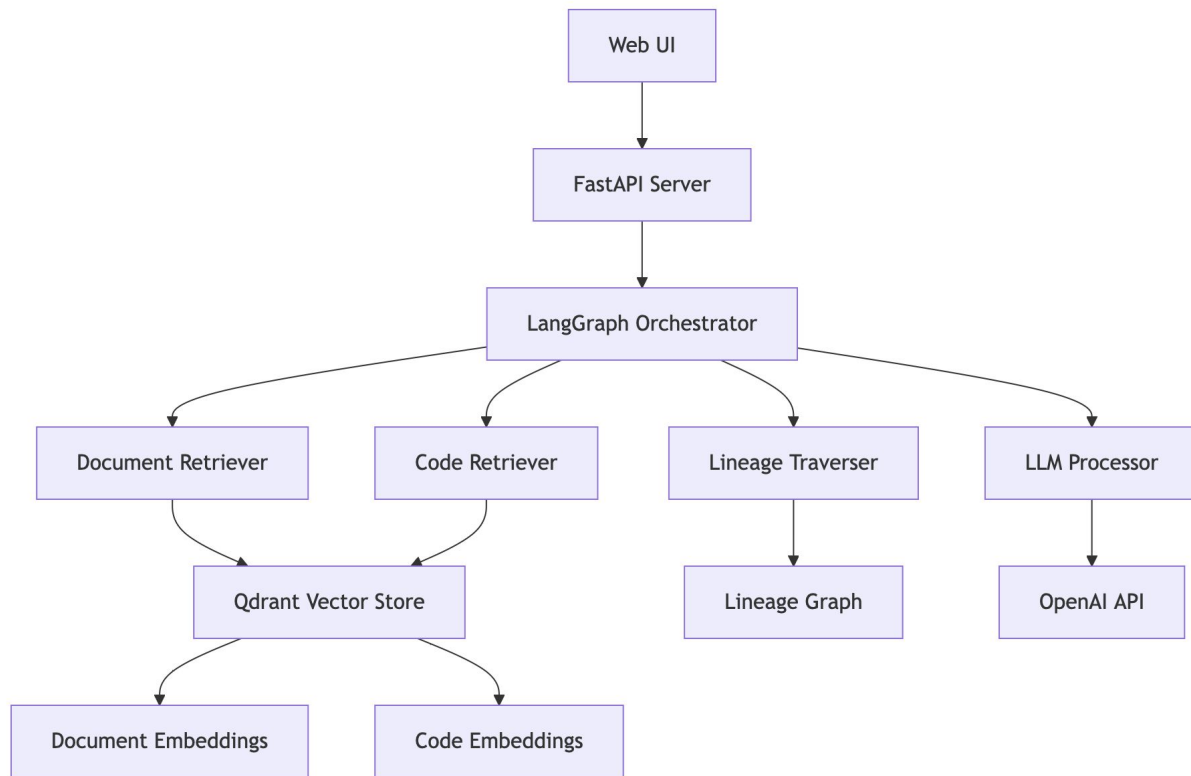
Maintains local scope for accurate answers

Lineage

Graph traversal in memory

Structural querying for relationships

🚀 System Architecture

```
                          ┌──────────┐
                          │  Web UI  │
                          └──────────┘
                                │
                                ▼
                       ┌─────────────────┐
                       │ FastAPI Server  │
                       └─────────────────┘
                                │
                                ▼
                    ┌────────────────────────┐
                    │ LangGraph Orchestrator  │
                    └────────────────────────┘
```

| Document Retriever | Code Retriever | Lineage Traverser | LLM Processor |

Qdrant Vector Store

Lineage Graph

OpenAI API

Document Embeddings

Code Embeddings

API Server	Web Interface	API Docs
http://localhost:8000	http://localhost:3000	http://localhost:8000/docs

🔍 Advanced Retrieval Methods

1 Hybrid Search

Combines semantic vector search with BM25 keyword scoring for better precision on keyword-heavy queries

0.742 overall score, +108.1% improvement

2 Lineage–Aware Retrieval

Enhances queries with data lineage context and table relationships for pipeline-specific optimization

0.783 overall score, +119.7% improvement (best overall)

3 Cohere Reranking

Advanced reranking using Cohere's rerank-english-v2.0 model for complex query precision

0.780 overall score, +118.9% improvement

4 Query Expansion

Semantic query enhancement with multiple search strategies for better recall through alternative phrasings

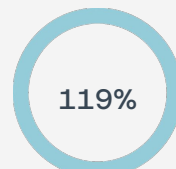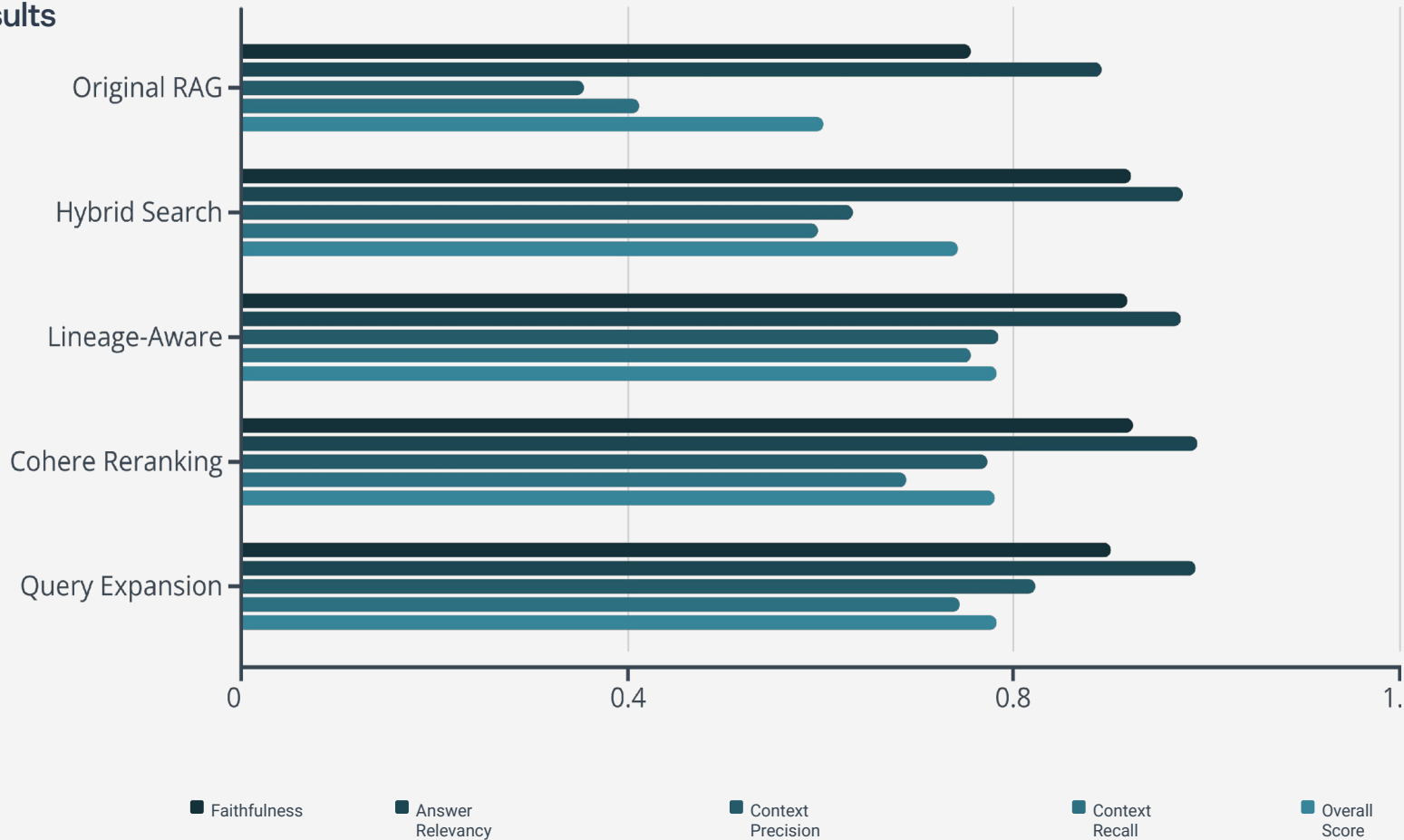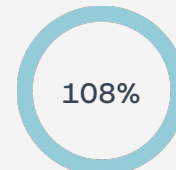0.782 overall score, +119.3% improvement

5 Original RAG

Standard vector similarity search serving as performance baseline

0.604 overall score (baseline)

🧪 RAGAS Evaluation Results

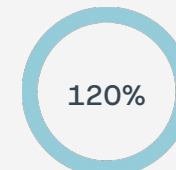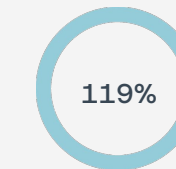Chart comparing RAGAS metrics across RAG approaches:

Systems (y-axis): Original RAG, Hybrid Search, Lineage-Aware, Cohere Reranking, Query Expansion

Metrics: Faithfulness, Answer Relevancy, Context Precision, Context Recall, Overall Score

X-axis: 0 to 1.0 (marks at 0, 0.4, 0.8, 1.)

Summary badges:
- **119%** — Cohere Reranking — Highest answer relevancy
- **108%** — Hybrid Search — Improvement over baseline
- **120%** — Lineage–Aware — Best overall performance
- **119%** — Query Expansion — Best context precision

The Future

Rapid & impactful improvements:

- Combined retrieval for improved scoring.
- Real-time data integration.

Building foundational capabilities for broader reach:

- Visual lineage diagrams.
- Robust enterprise integration.

Innovating for proactive, collaborative intelligence:

- Predictive incident models.
- Collaborative response features.