COELHO NEXUS

A KUBERNETES-NATIVE MULTI-SERVICE AI AGENT PLATFORM

DOCS DISTILLER
YOUTUBE CONTENT SEARCH
RESEARCH RADAR

FASTAPI
FASTHTML
FASTMCP

Rafael Coelho

COELHO NEXUS

BUILT SOLO. PRODUCTION-READY.

3 MICROSERVICES FastAPI · FastHTML · FastMCP	6 DATA STORES PostgreSQL · Neo4j · Qdrant Elasticsearch · MinIO · Redis	18 TERRAFORM MODULES K3d · ArgoCD · Helm Full GitOps IaC
3 AI FEATURES Docs Distiller YouTube Search Research Radar	100% FREE-TIER LLMS 20+ models · Bandit routing BYOK encryption	1 ENGINEER Code → Infra → GitOps End-to-end ownership

COELHO
NEXUS

ABOUT THE PROJECT

Production-grade, Kubernetes-native AI Agent Platform — built solo, shipped end-to-end.

- 3 microservices (FastAPI, FastHTML, FastMCP) fused into a unified intelligence layer
- 100% free-tier LLMs — 20+ models, bandit routing, BYOK (Bring Your Own Key) credential management
- Full OpenTelemetry + LangFuse observability — every LLM call traced, every token accounted for

COELHO NEXUS

PLATFORM ARCHITECTURE

01.

THREE MICROSERVICES & THREE FEATURES

FastAPI + Celery orchestrate six isolated task queues, one per AI feature pipeline.

FastHTML delivers an HTMX-in-SSR (Hypertext eXtensions + Server-Side Rendering) reactive UI with Cytoscape graph visualizations and SSE (Server-Sent Events) live event streaming.

FastMCP exposes four AI agent discovery tools with rate-limited token buckets and BYOK (Bring Your Own Key) credential injection.

02.

INFRASTRUCTURE AS CODE & SOLO OWNERSHIP

18 Terraform modules provision the full data layer — PostgreSQL, Neo4j, Qdrant, Elasticsearch, MinIO, Redis — declaratively via Terragrunt on K3d.

ArgoCD automates the full GitOps delivery pipeline; OpenTelemetry dual export feeds LangFuse trace visualization and Grafana dashboards.

THE PLATFORM IN ACTION

A visual walkthrough of the COELHO Nexus platform — three production-grade AI features (Docs Distiller · YouTube Content Search · Research Radar) built on FastAPI, FastHTML, and FastMCP, orchestrated on Kubernetes with full LangFuse and OpenTelemetry observability.

COELHO NEXUS

DOCS DISTILLER

Catalog
Ingestion
Pipeline
Study

COELHO
NEXUS

COELHO
NEXUS

DOCS DISTILLER

The full depth of 400+ pages of official documentation — distilled from weeks of reading into hours of study.

1. Pick any framework and it downloads the complete official documentation — every page, automatically cleaned into Markdown

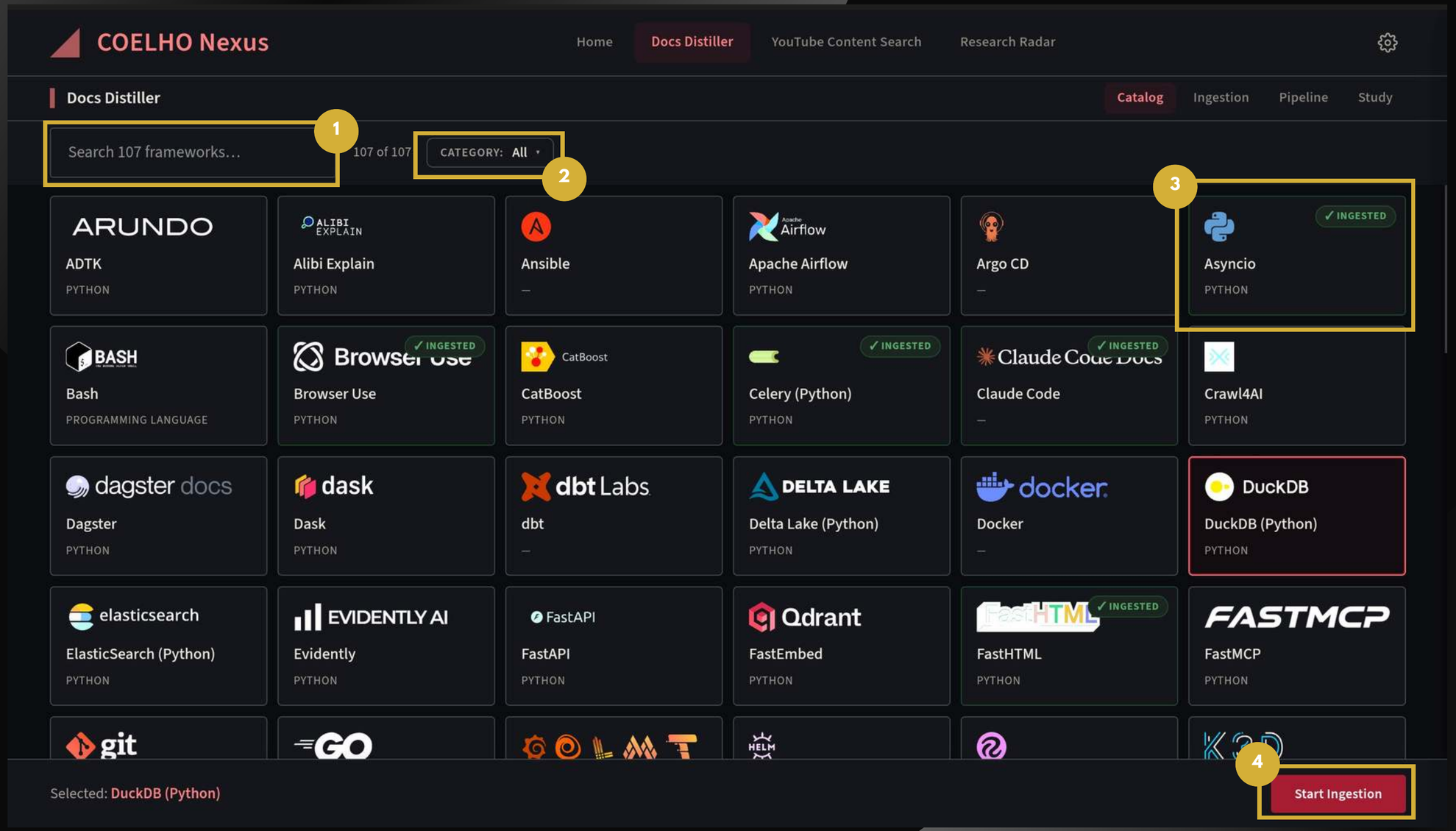
2. Planner reads the entire corpus and organizes it into a textbook outline — the structure a human expert would design

3. Synth writes each chapter from the real source material — code examples preserved, checked against the docs for accuracy

Why it matters:

- The same depth as reading every page yourself — at a fraction of the time, because the work is already done.
- Zero cost: Every LLM call — reading, planning, writing — automatically routes to the best free-tier model available. No API bills, ever.

DOCS DISTILLER - CATALOG



1 Search field

Real-time search across 100+ catalogued documentation sources. Partial-match filtering instantly narrows the catalog by framework name or technology domain.

2 Category filter

Filter the catalog by technology domain — AI/ML, databases, infrastructure, DevOps, monitoring. Groups semantically related frameworks for focused multi-source ingestion runs.

3 Ingested option

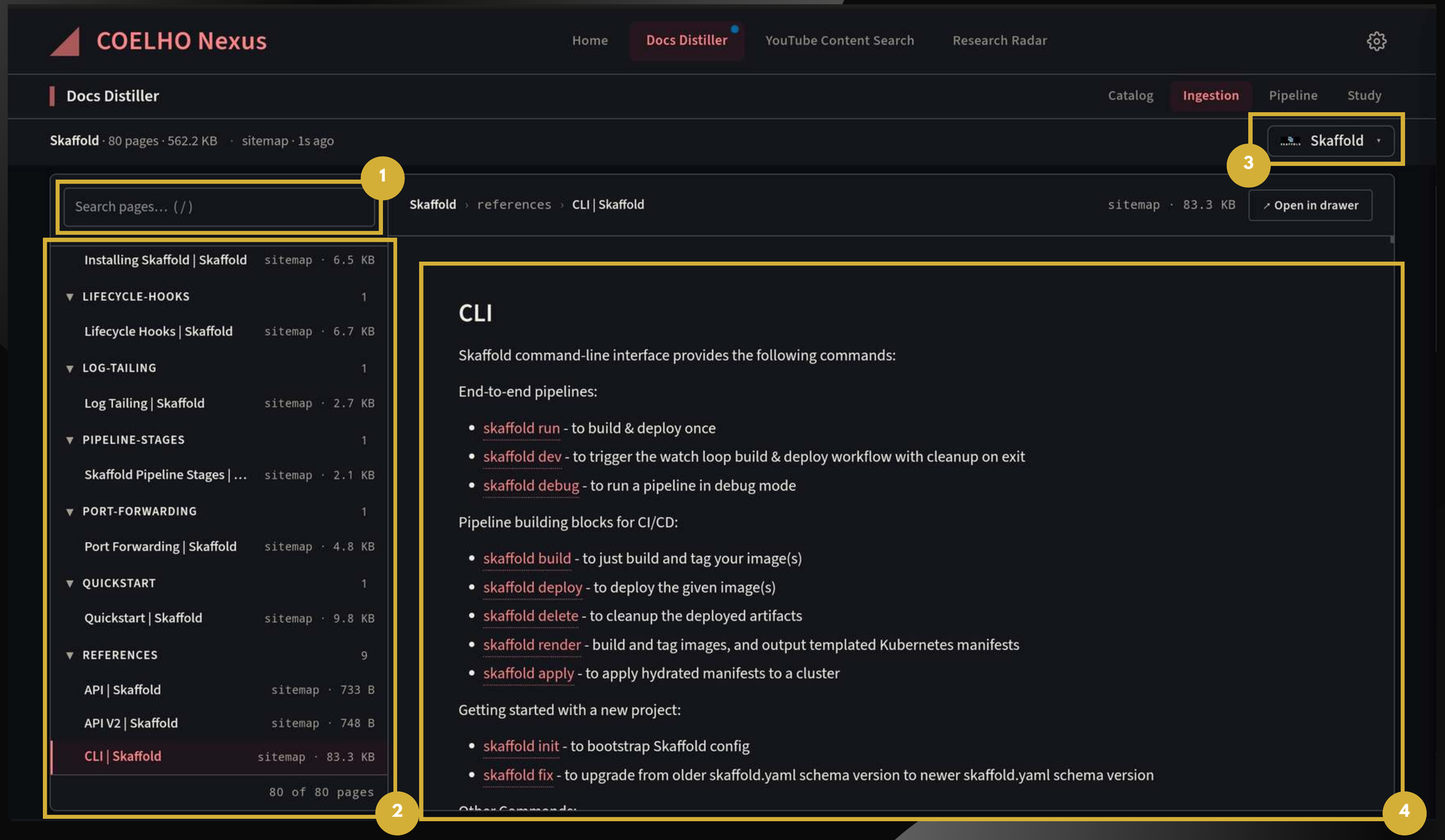
Marks frameworks already ingested into the knowledge base. The badge identifies gaps and lets you trigger re-ingestion when documentation is updated.

4 Ingestion button

Triggers the 4-tier ingestion pipeline for selected sources — llms_full.txt → llms_txt → sitemap → httpx-crawl-with-Playwright-fallback — with real-time progress streamed via SSE to the Pipeline view.

COELHO NEXUS

DOCS DISTILLER - INGESTION



1 Search pages

Search within the ingested documentation pages of the selected framework. Instantly locates specific sections, commands, or API references within the full fetched documentation tree.

2 Fetched docs pages

Complete list of documentation pages fetched from the source URL. Each entry shows the page title and ingestion tier used — llms_full.txt, llms_txt, sitemap, httpx-crawl-with-Playwright-fallback.

3 Ingested frameworks dropdown

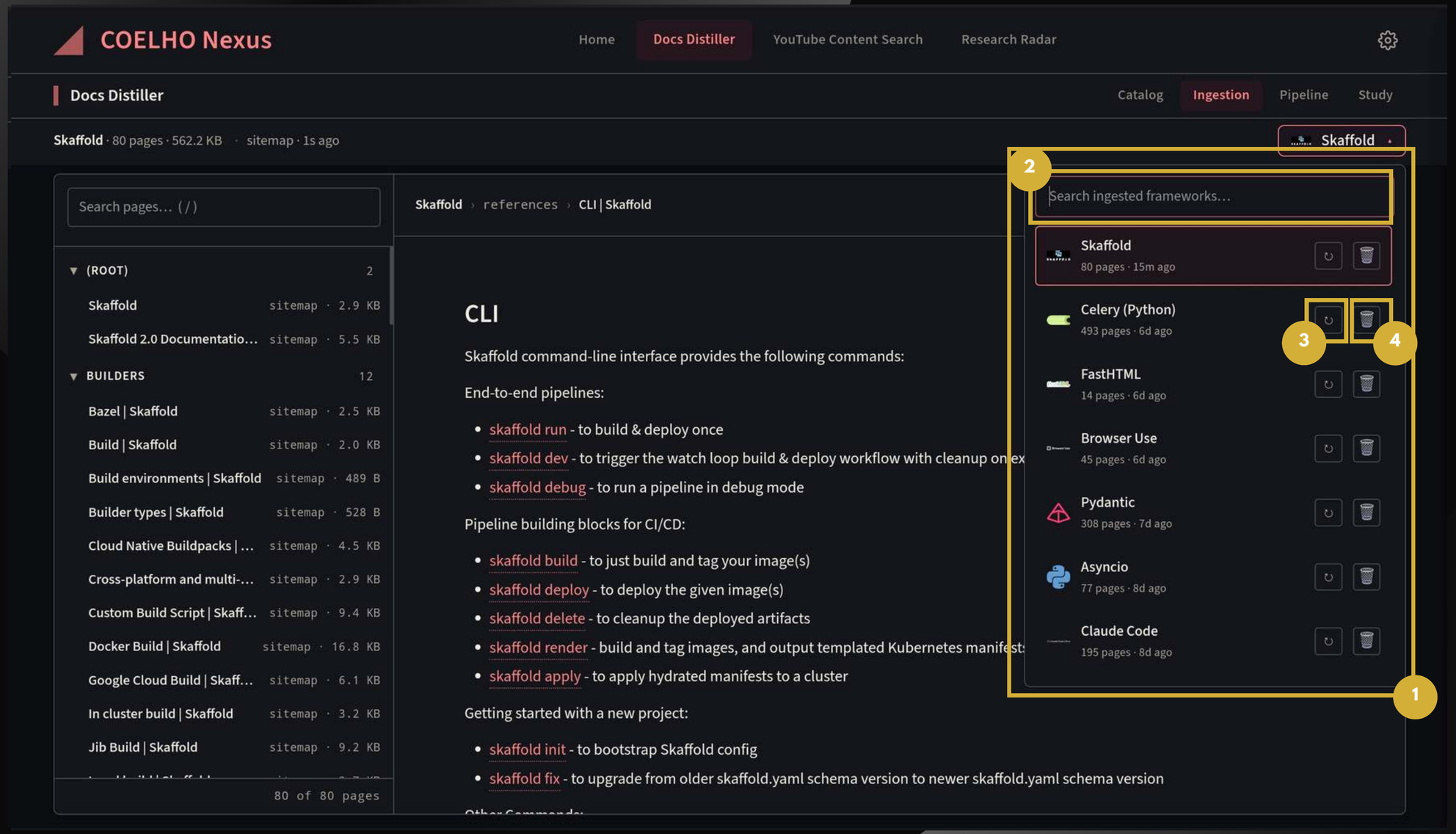
Switch between all frameworks currently in the knowledge base. Selecting a framework updates the page list and main view to show its complete documentation structure.

4 Main page visualization

Full rendered view of the selected documentation page as captured and processed. Shows the raw content before it enters the Planner → Synth pipeline for study guide generation.

COELHO NEXUS

DOCS DISTILLER - INGESTION



1 Ingested frameworks dropdown options

Lists all documentation sources currently in the knowledge base, each showing its framework name, and total page count for context planning.

2 Search ingested frameworks options

Real-time search within the framework picker. Filter across all ingested documentation sources by name to quickly locate and switch to any framework in the knowledge base.

3 Refresh framework docs ingestion

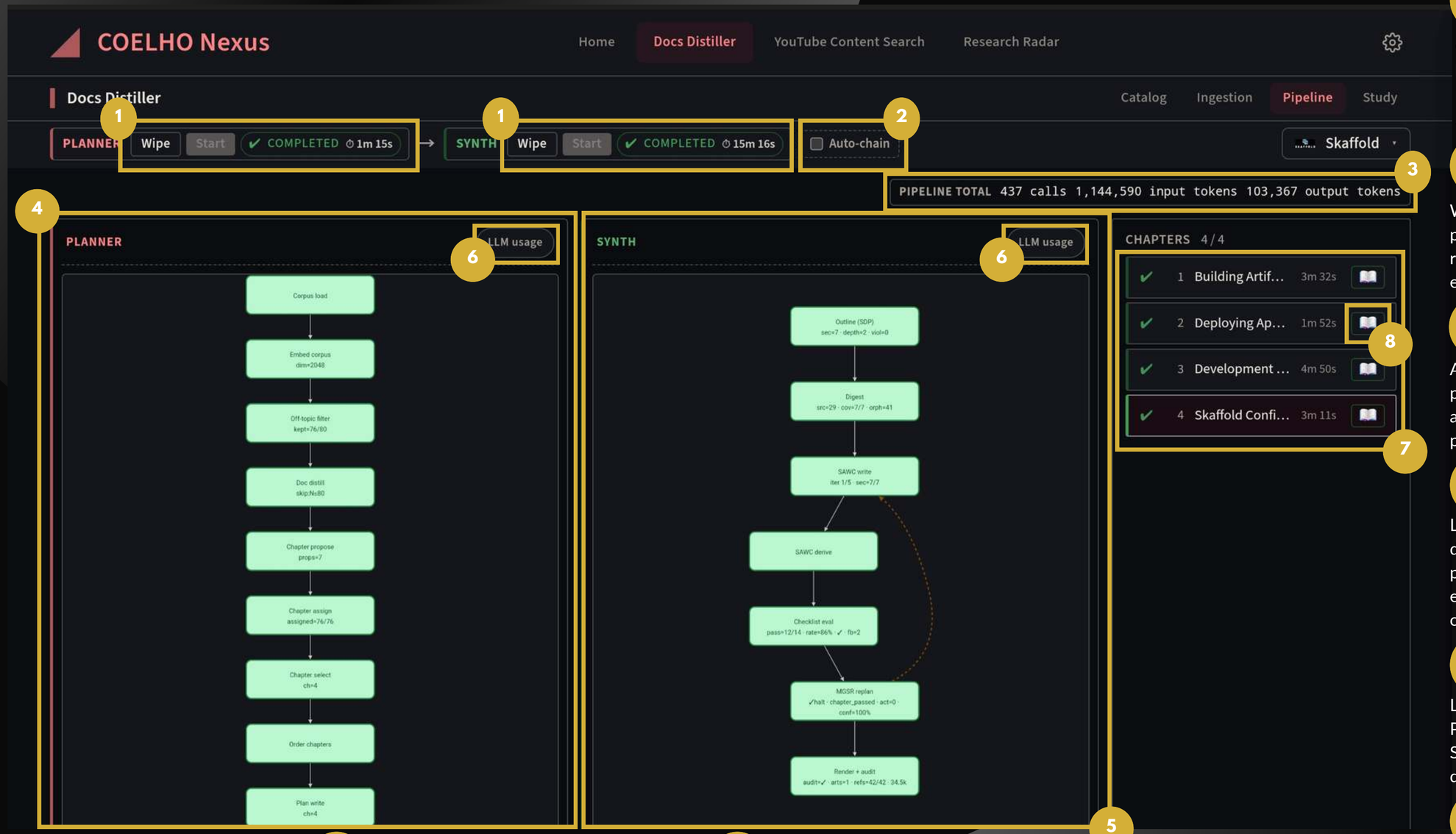
Re-runs the full 4-tier ingestion pipeline for the selected framework. Use when upstream documentation has been updated to sync the knowledge base with the latest content.

4 Delete framework docs ingestion

Removes all ingested pages for the selected framework from the knowledge base — clearing Qdrant vectors, chunk metadata, and tier records — enabling a clean re-ingestion from scratch.

COELHO NEXUS

DOCS DISTILLER - PIPELINE



1 Pipeline control bar

Controls for starting, stopping, and wiping the Planner and Synth pipelines. The status bar shows real-time pipeline state (RUNNING / COMPLETED / IDLE) and elapsed time per stage.

2 Auto-chain option

When enabled, automatically triggers the Synth pipeline immediately after Planner completes — running the full Planner → Synth sequence end-to-end without manual intervention.

3 Run completion stats

Aggregate run statistics: total documents processed, chapters proposed, chapters assigned, and synthesis iterations completed across the full pipeline execution.

4 LangGraph Planner

LangGraph planning pipeline that distills ingested documents into a structured chapter outline — LLM proposes chapters from parallel summaries, assigns each doc to its best fit, then selects for maximum corpus coverage.

5 LangGraph Synth

LangGraph synthesis pipeline that expands the Planner outline into full study guide chapters via SAWC (Section-Aware Writer-Critic) — iterative drafting with LLM critic until quality gates pass.

6 LLM telemetry per stage

Per-node LLM call telemetry: model used, tokens consumed, latency, and bandit arm selection. Routes through the FGTS-VA (Fine-Grained Thompson Sampling with Variance Awareness) bandit rotator across 20+ free-tier models.

7 Study chapters synthesized

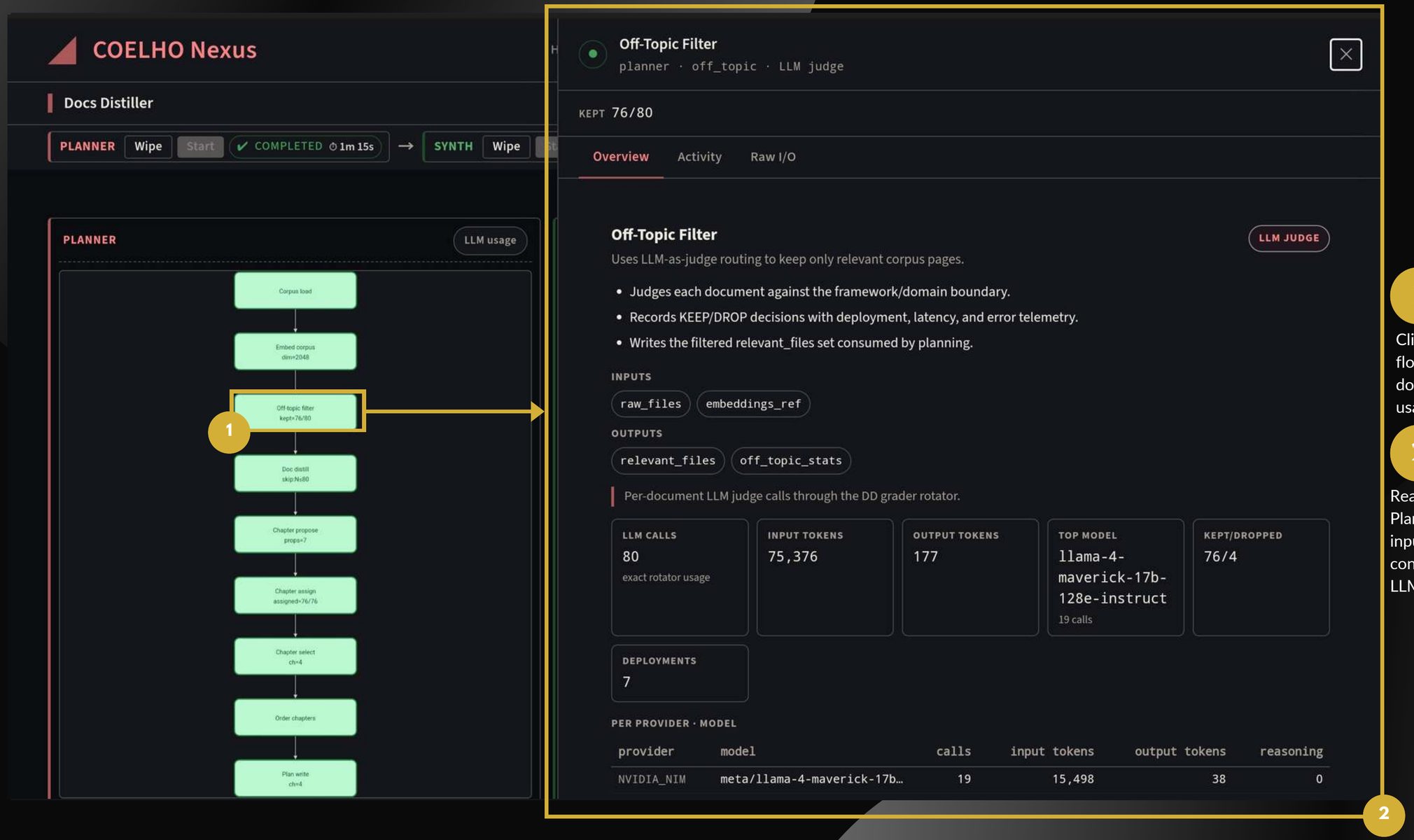
Running count of completed study guide chapters. Each chapter is independently synthesized and streamed to the Study view as it finishes — no need to wait for the full run.

8 Chapter reading button

Opens the completed study guide chapter directly in the Study view. Available as each chapter finishes synthesis, enabling progressive reading before the full run completes.

COELHO NEXUS

DOCS DISTILLER - PIPELINE



1 Clickable LangGraph node

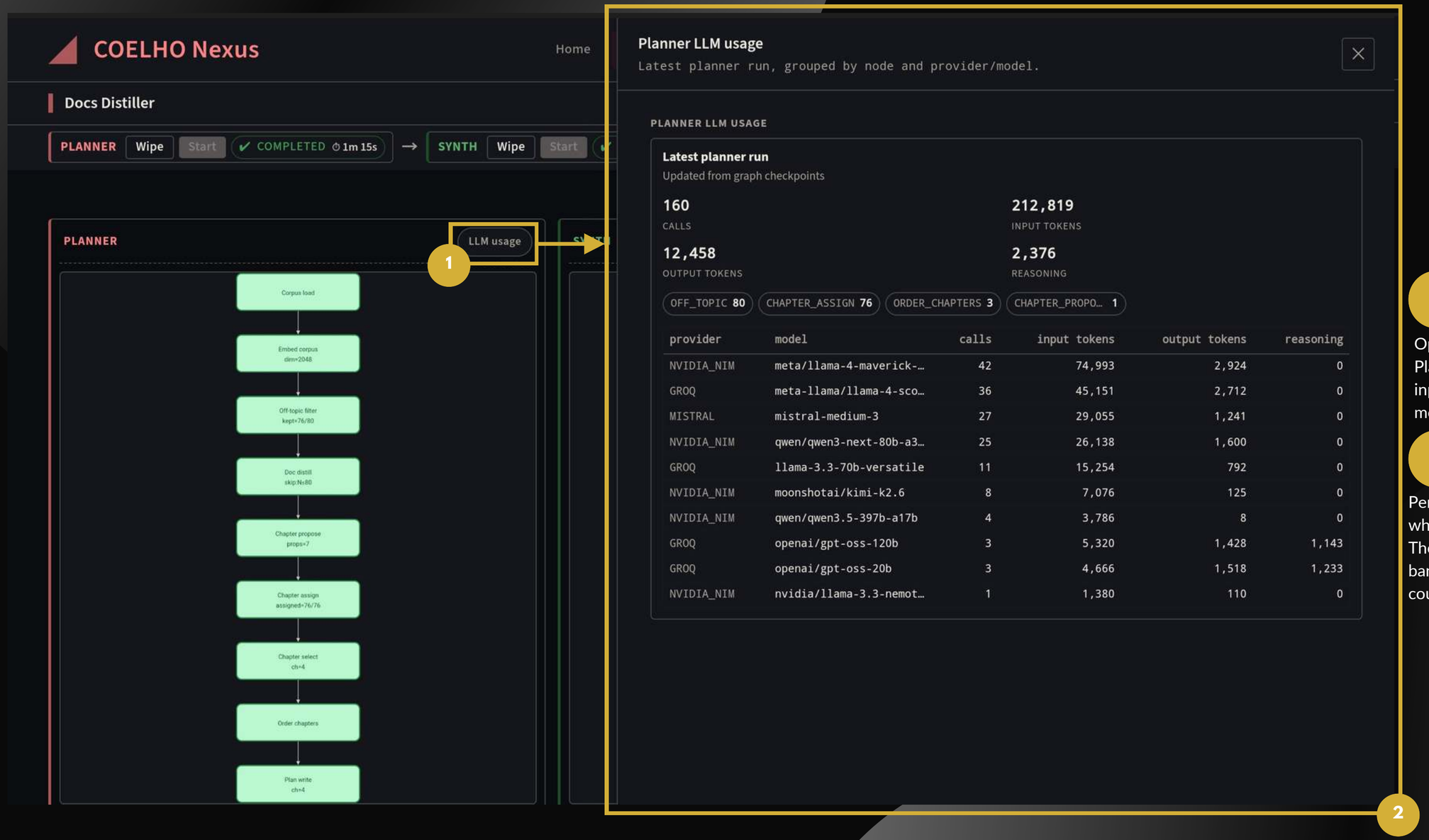
Click any node in the LangGraph Planner/Synth flow to inspect its execution — input/output documents, filtering decisions, LLM calls, and token usage per node.

2 Node execution telemetry

Real-time execution detail for the selected Planner/Synth node — judge configuration, input/output document counts, total tokens consumed, latency distribution, and per-document LLM call breakdown.

COELHO NEXUS

DOCS DISTILLER - PIPELINE



1 Planner/Synth LLM usage panel

Opens the LLM token summary for the last Planner/Synth run — total calls, input/output/reasoning tokens, and per-node model breakdown grouped by bandit algorithm arm.

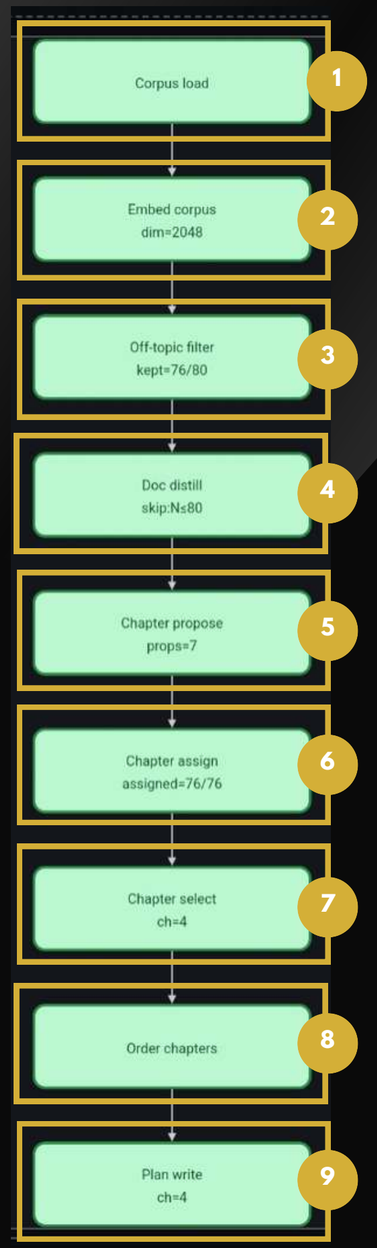
2 Bandit algorithm routing telemetry

Per-node, per-provider token accounting — shows which model the FGTS-VA (Fine-Grained Thompson Sampling with Variance Awareness) bandit routed each call to, with input/output token counts. Full traceability across 20+ free-tier models.

COELHO NEXUS

DOCS DISTILLER - PLANNER

COELHO NEXUS



1 Corpus Inventory

Reads the finalized MinIO ingestion manifest, resolves per-page object keys, and computes byte-size stats — the source of truth that seeds all downstream nodes.

2 Semantic Embedding

Token-aware BPE (Byte-Pair Encoding) chunking aligned to the NVIDIA NIM embedding model → L2-normalized dense vectors packed as .npz in MinIO. Hash-skipped on re-runs.

3 Off-Topic Filter

Bandit-routed LLM judge (KEEP / DROP) with cosine anchor similarity and concurrent per-doc evaluation. Defaults to KEEP on failure — no content silently dropped.

4 Document Distillation

Parallel LLM calls produce structured distillates (summary + key terms) per doc. Repair loop + fallback ensure full corpus coverage under rate-limits or parse failures.

5 Chapter Proposal

Long-context LLM samples N chapter structures from all distillates + H2 structural seeds. USC (Universal Self-Consistency) voting aggregates proposals; optimal stopping halts early when consensus converges.

6 Document Assignment

Per-doc LLM confidence scoring across all proposed chapters. Lexical fallback + rescue floor guarantee every content doc gets assigned — no silent drops.

7 Coverage Selection

Greedy algorithm selects the minimal chapter set covering all docs above the minimum threshold. Writes versioned + latest plan blobs to MinIO for resume-safety.

8 Pedagogical Ordering

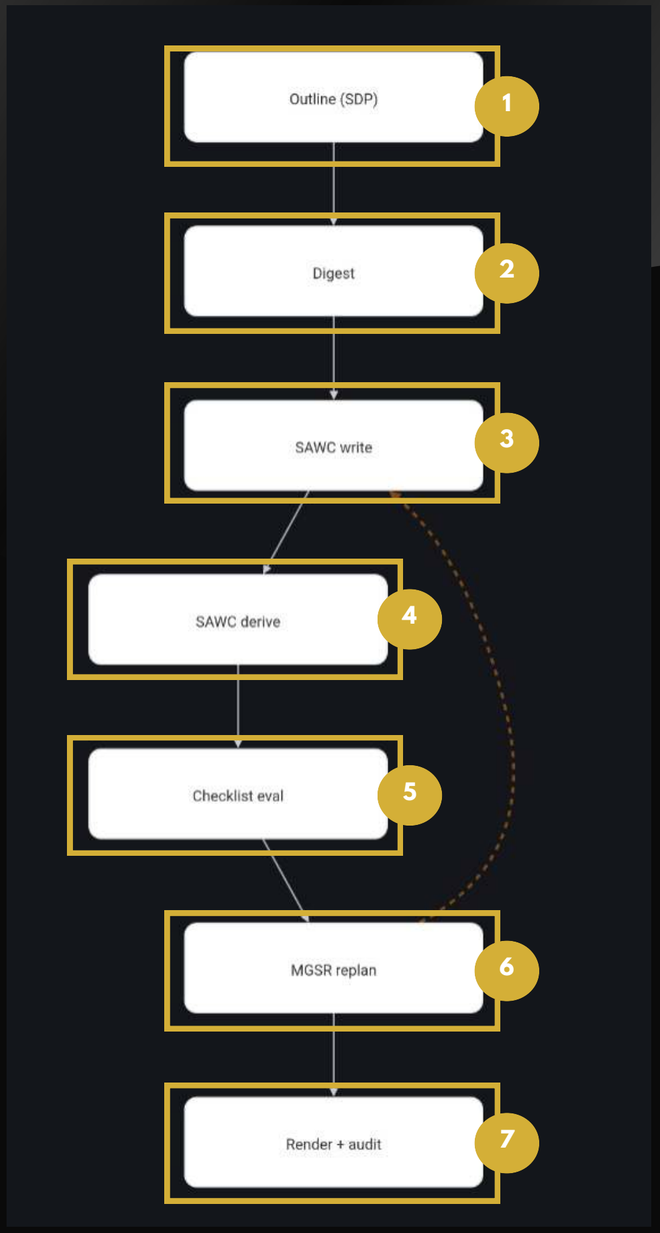
N bandit-routed LLM samples rank chapter sequences independently. Borda aggregation + foundational-prefix rule ensure foundational concepts always precede advanced material.

9 Plan Assembly

LLM-free final step: sanitizes chapter metadata, inlines provenance refs, and persists the versioned + latest plan blob. Output feeds directly into the Synth pipeline.

DOCS DISTILLER - SYNTH

COELHO NEXUS



1 Outline Generation

LLM generates the chapter outline as a dependency DAG (Directed Acyclic Graph) — H2 sections with prerequisite links, adaptive section count, and fuzzy-dedup headings. The structured skeleton all downstream nodes build from.

2 Source Digest

24-concurrent bandit-routed LLM call per source doc — extracts key facts, section contributions, and relevance scores. Content-addressed cache avoids redundant calls on re-runs.

3 SAWC Writer-Critic

Best-of-N LLM drafts per section + critic picker (MAMM-Refine, arXiv:2503.15272). Structured output: heading, intro, subtopics with code refs and citations. 2-attempt repair loop enforces SAWC (Section-Aware Writer-Critic) alignment.

4 Derived Code Enrichment

Analogical Prompting (arXiv:2310.01714) detects thin vault blocks and generates derived business code examples via MPSC (Multi-Path Self-Consistency) sampling — ensures every section has implementable code.

5 Quality Checklist

Multi-criteria pass/fail gating: density, code ref coverage, citation density, LLM-judge criteria, CoCoA (Code Comprehension then Auditing, arXiv:2410.03131) alignment, and faithfulness scoring. Generates per-criterion feedback for the replan loop.

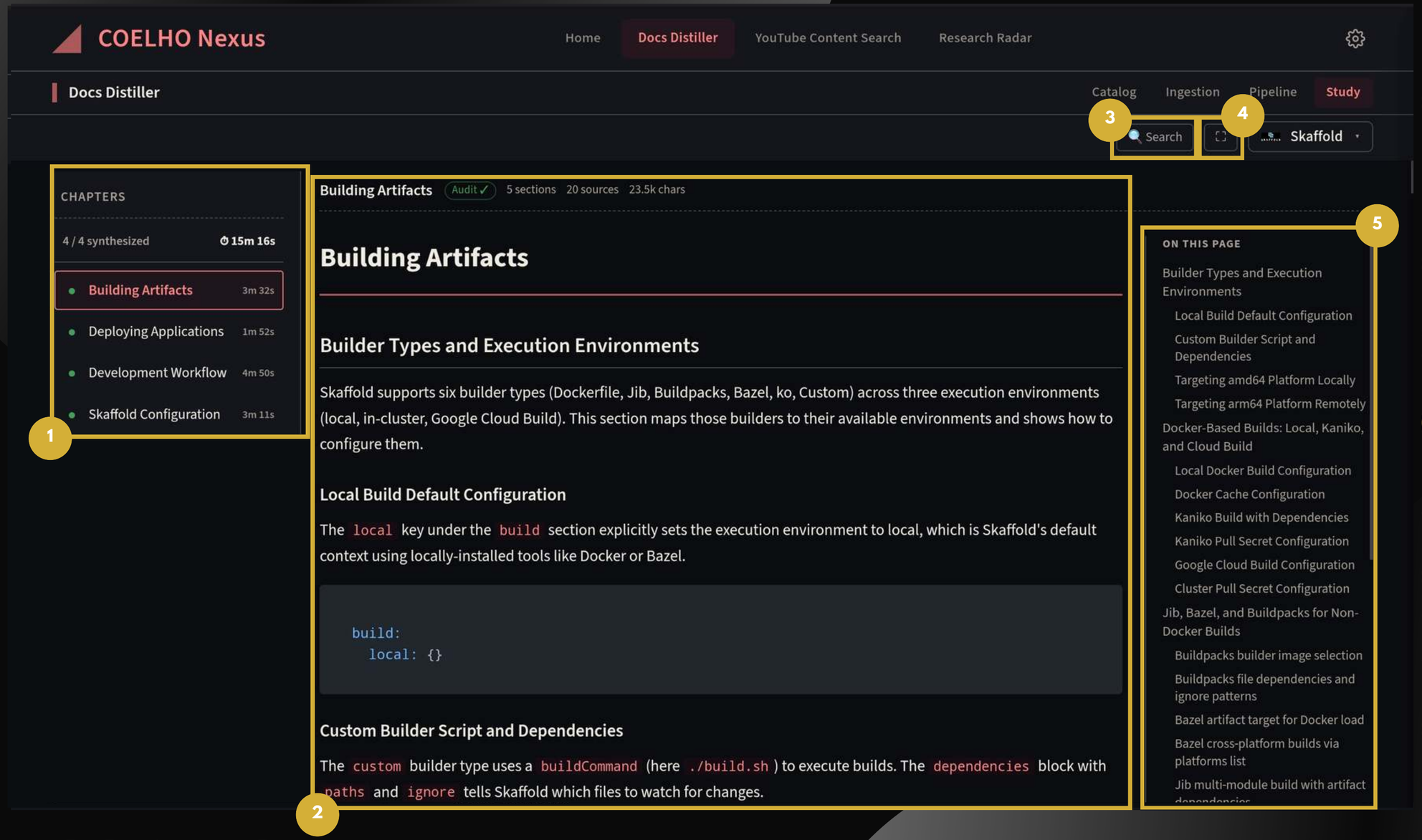
6 MGSR (Meta-Grounded Self-Refinement)

Reads checklist failures and issues structured replan actions. Trivial-pass fast-path at ≥80% skips LLM entirely. On failure: bandit-routed replan → conditional edge loops back to SAWC (Section-Aware Writer-Critic) write.

7 Render & Audit

Resolves vault sentinel hashes to byte-exact code blocks, runs cached LLM normalization, deduplicates sections, computes audit metrics, and persists the final chapter Markdown artifact.

DOCS DISTILLER - STUDY



1 Synthesized Chapter List

Navigation panel listing all chapters generated by the LangGraph Synth pipeline for the selected framework — each chapter assembled from 100s of ingested source pages.

2 Rendered Chapter View

Full AI-synthesized chapter rendered in-browser — structured sections, subtopics, code examples, and citations produced by the SAWC Writer-Critic pipeline from the Synth agentic graph.

3 Cross-Chapter Search

Full-text search across all synthesized chapters — instantly locate any term, concept, or code snippet across the entire distilled knowledge base.

4 Expand Chapter View

Collapses the left chapter navigation panel to maximize the reading area — full-width immersive view of the synthesized chapter content.

5 Chapter Table of Contents

Auto-generated table of contents from the chapter's H2 section structure — jump directly to any section within the synthesized chapter.

COELHO NEXUS

YOUTUBE CONTENT SEARCH

Source
Ingestion
Ask
Query

COELHO NEXUS

COELHO
NEXUS

YOUTUBE CONTENT SEARCH

Information locked inside YouTube videos — invisible to Google, unreachable by ChatGPT or Claude — finally searchable.

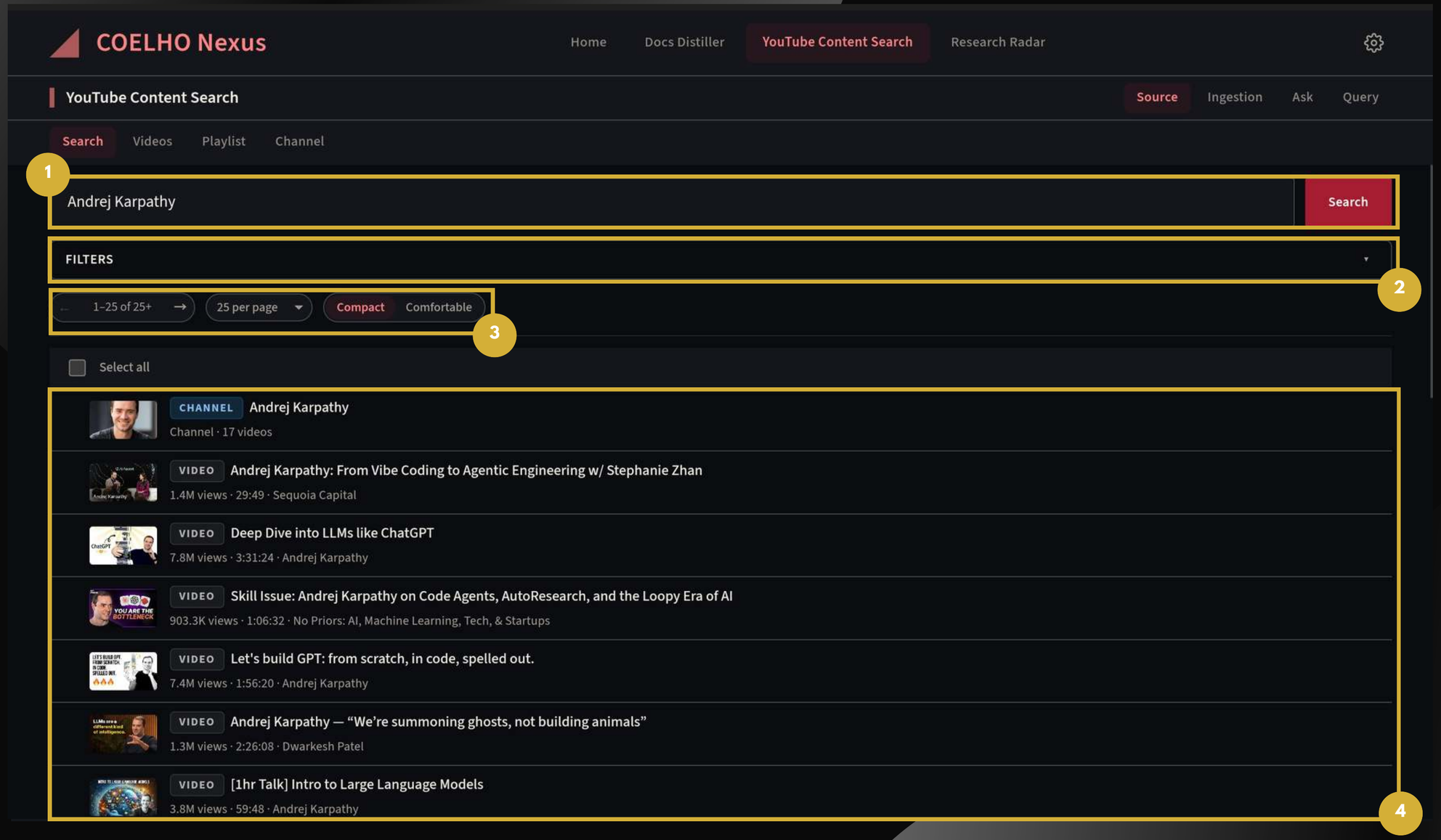
1. Point it at any channel and it extracts full transcripts straight from the video page — the same content Google can't index

2. Every transcript is indexed three ways — semantic, graph, and full-text (Qdrant + Neo4j + Elasticsearch) — connections surface automatically

3. Ask a question and get an answer synthesized across every matching video — with the exact source clips cited, not a generic summary

Why it matters: Cross-video insight that Google and general AI assistants can't give you — they can't see inside YouTube's transcripts.

YOUTUBE CONTENT SEARCH - SOURCE



1 yt-dlp Live Search

Quota-free YouTube search via yt-dlp — returns videos, channels, and playlists with title, channel, view count, duration, likes, and upload date in a single call. Sub-2-second results piped directly into the multi-row ingestion selector.

2 Native HTML Filter Grid

Nine search parameters — duration range, content kind, upload date window, view and like thresholds, title keyword, and channel name — applied server-side via native HTML form controls on every yt-dlp call. No JavaScript event handlers, no client-side filter state.

3 Results Controls

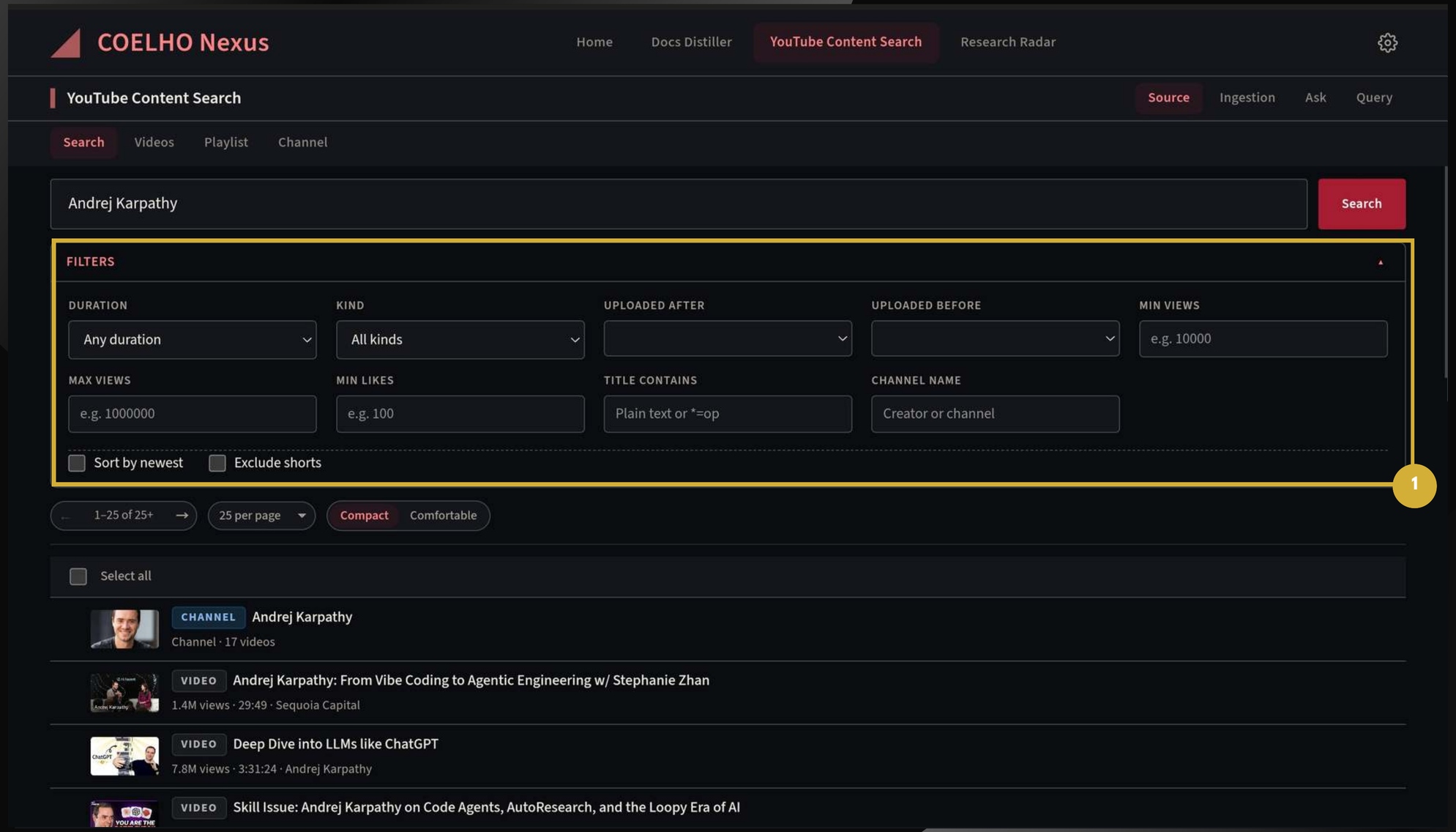
Pagination with configurable page size (10–200 per query), prev/next controls with a live count display, and Compact/Comfortable density toggle for scanning large result sets fast.

4 Video Result Cards

Each card surfaces thumbnail, channel name, view count, duration, like count, and upload date. Per-row checkbox plus master Select All feeds the Start Ingestion pipeline with exactly the videos you choose.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - SOURCE



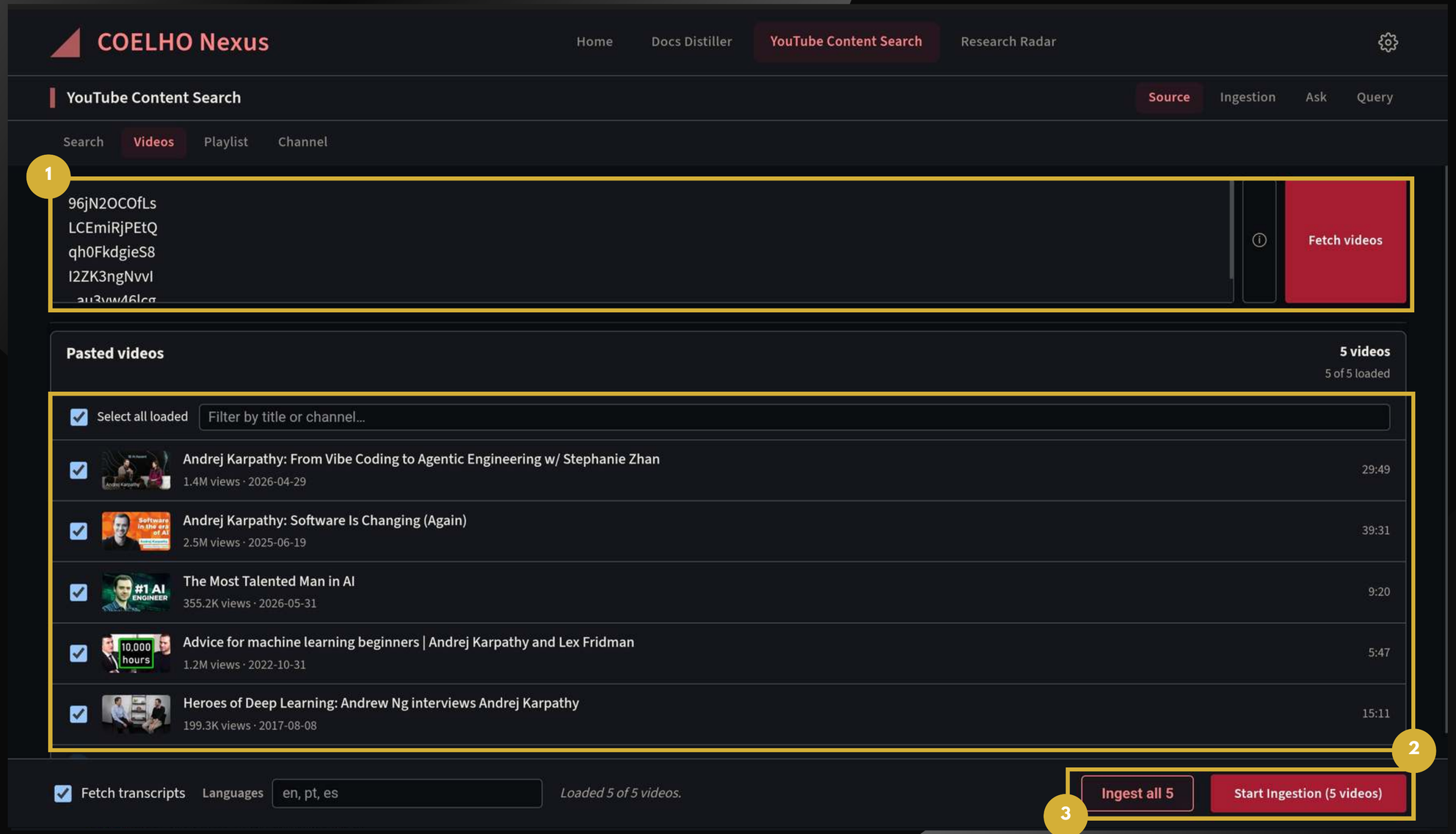
1

Parametric Search Controls

Nine real-time search parameters — duration range, content kind (video/channel/playlist), upload date bounds, view range and like minimum, title keyword, and channel filter — applied on every yt-dlp query via native HTML form controls.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - SOURCE



1 Batch URL Input

Paste YouTube video IDs, short URLs, or full watch URLs — one per line, 50+ entries per batch. Click Fetch Videos to pull title, channel, view count, duration, likes, and upload date via yt-dlp with zero API quota.

2 Fetched Video Cards

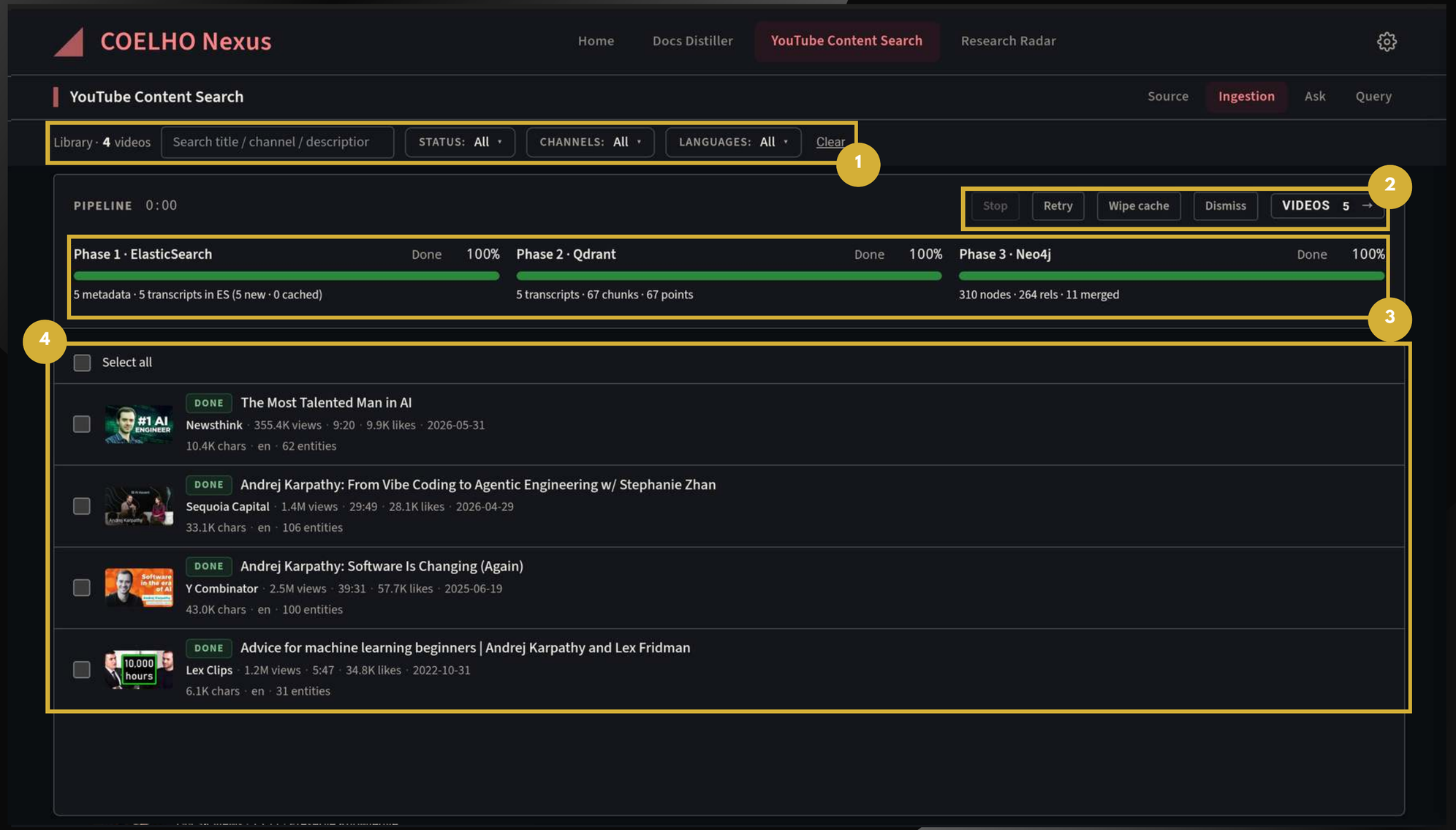
Each card shows thumbnail, title, channel, view count, duration, and upload date. Per-row checkbox and master Select All let you pick exactly which videos to ingest before starting the pipeline.

3 One-Click Pipeline Trigger

Ingest All selects every fetched video. Start Ingestion dispatches the Celery chain: yt-dlp extract → Qdrant vector upsert → Neo4j entity graph → cache invalidation. Count badge updates in real time as videos are selected.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - INGESTION



1 Search & Filter Bar

Filter the already-ingested video library by status, channel, and language. Use the search bar to locate any video by title or keyword.

2 Pipeline Run Controls

Manage the active ingestion run: Stop a running pipeline, Retry failed phases, Wipe cache to bypass cached transcripts and force a full re-ingest, or Dismiss the run. VIDEOS shows the batch size with a link to the queued set.

3 3-Phase Pipeline Progress

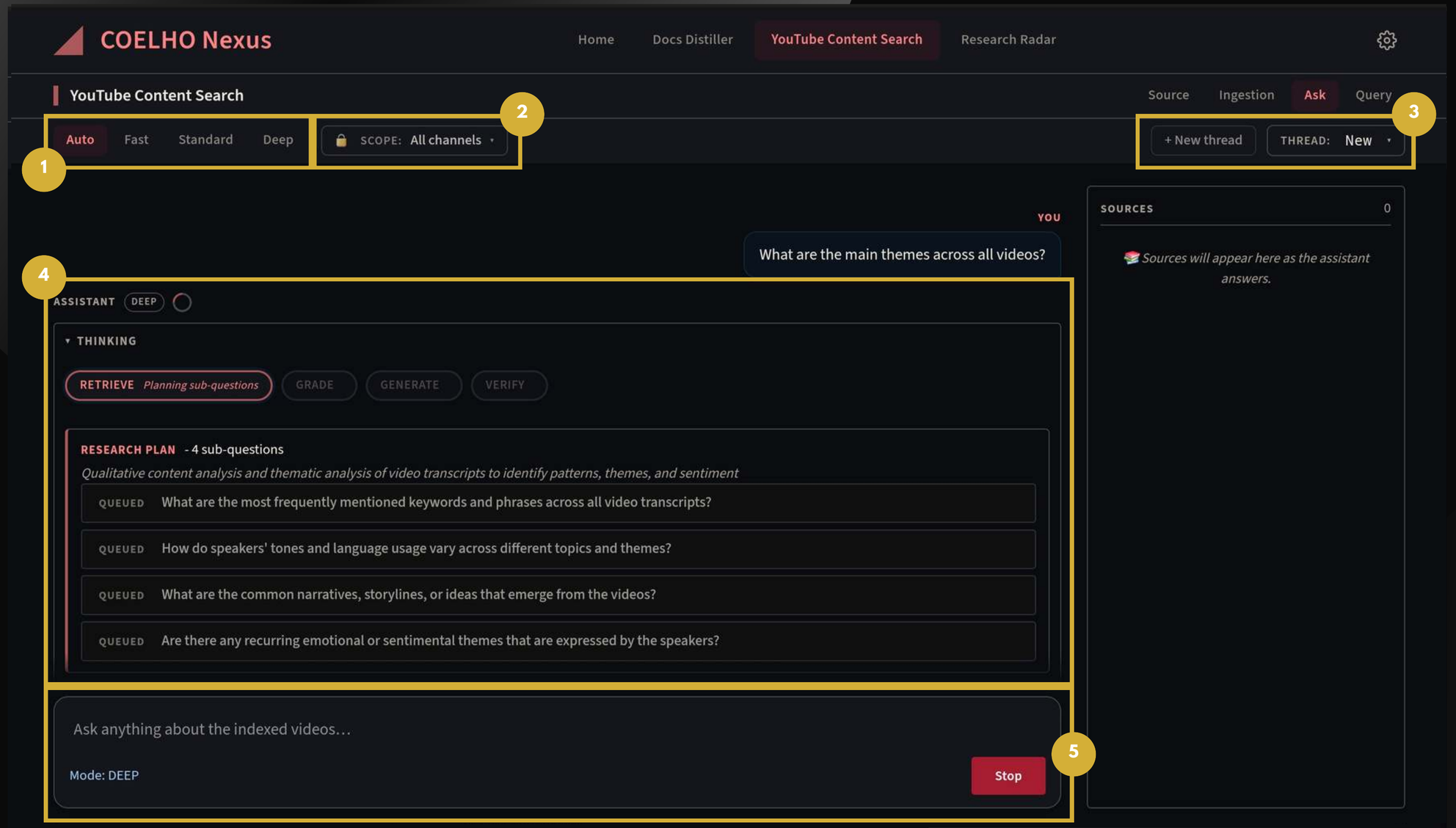
Live Celery chain status across three sequential phases: Phase 1 Elasticsearch (transcript indexing + BM25), Phase 2 Qdrant (dense vector upsert), Phase 3 Neo4j (entity extraction + graph write). Real-time percentage shown per phase.

4 Per-Video Status Cards

Each card shows thumbnail, title, channel, view count, duration, and live ingestion status. Failed videos expose the broken phase inline so you can retry a single video without re-running the full batch.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - ASK



1 Query Mode Selector

Four depth modes — Auto, Fast, Standard, and Deep — balance speed against answer quality. Auto picks the best mode per question; Deep runs full multi-step RAG with sub-question decomposition and source verification.

2 Channel Scope

Scope any query to all indexed channels or a specific one. Narrows vector retrieval and BM25 search to a focused corpus — ideal for drilling into a single creator's content or comparing themes across channels.

3 Conversation Threads

Each thread maintains its own conversation history and context window. Create a new thread to start fresh or switch between saved threads to resume a prior research session without losing context.

4 Deep RAG Pipeline

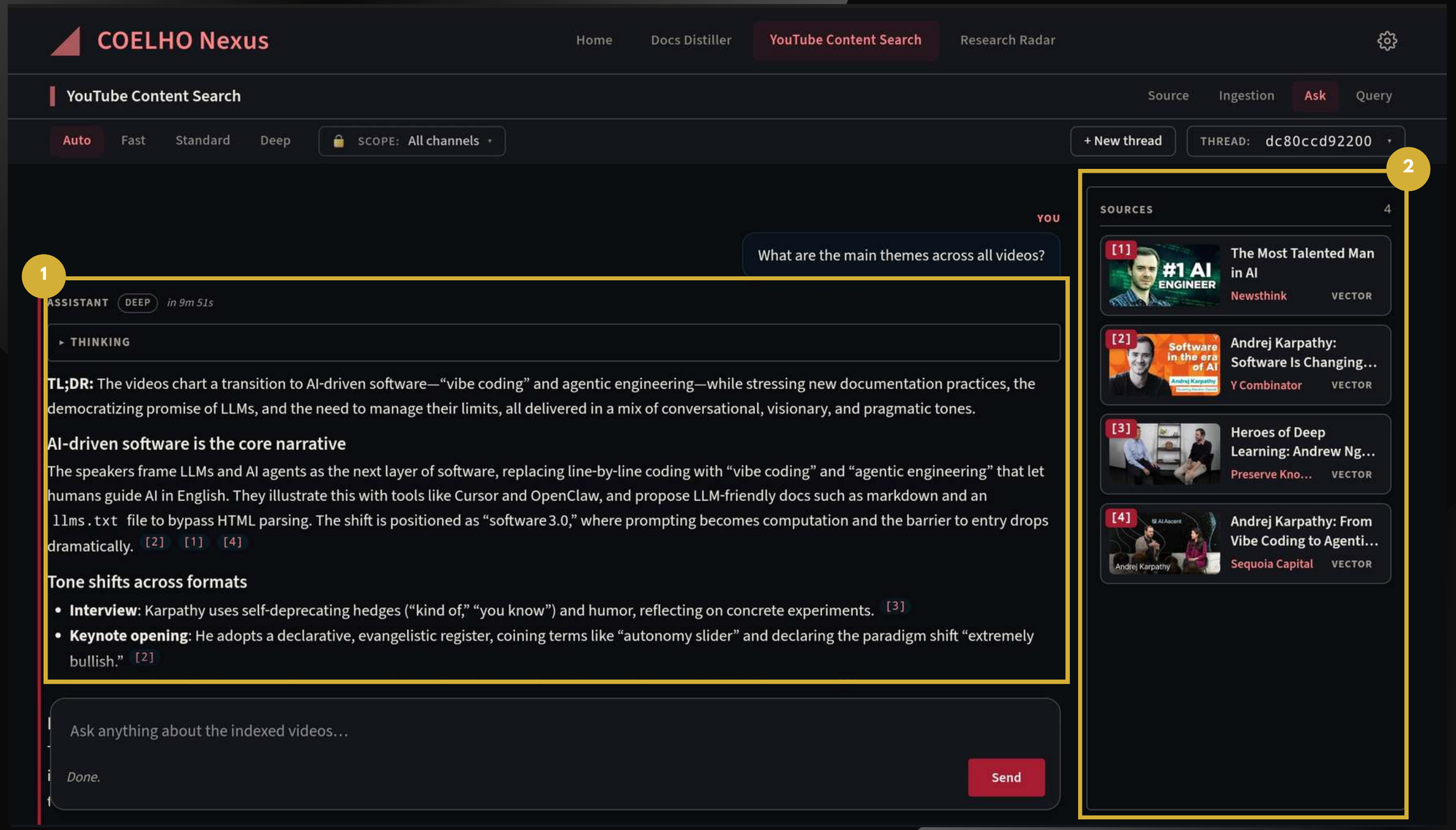
Deep mode decomposes the question into sub-questions, runs RETRIEVE → GRADE → GENERATE → VERIFY across indexed transcripts, and streams a grounded answer with inline sources. The research plan and step status update in real time.

5 Natural Language Input

Ask anything about the indexed video library in plain language. The active query mode is shown inline; Stop halts a running generation mid-stream without losing the partial answer.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - ASK



1 Grounded Answer with Citations

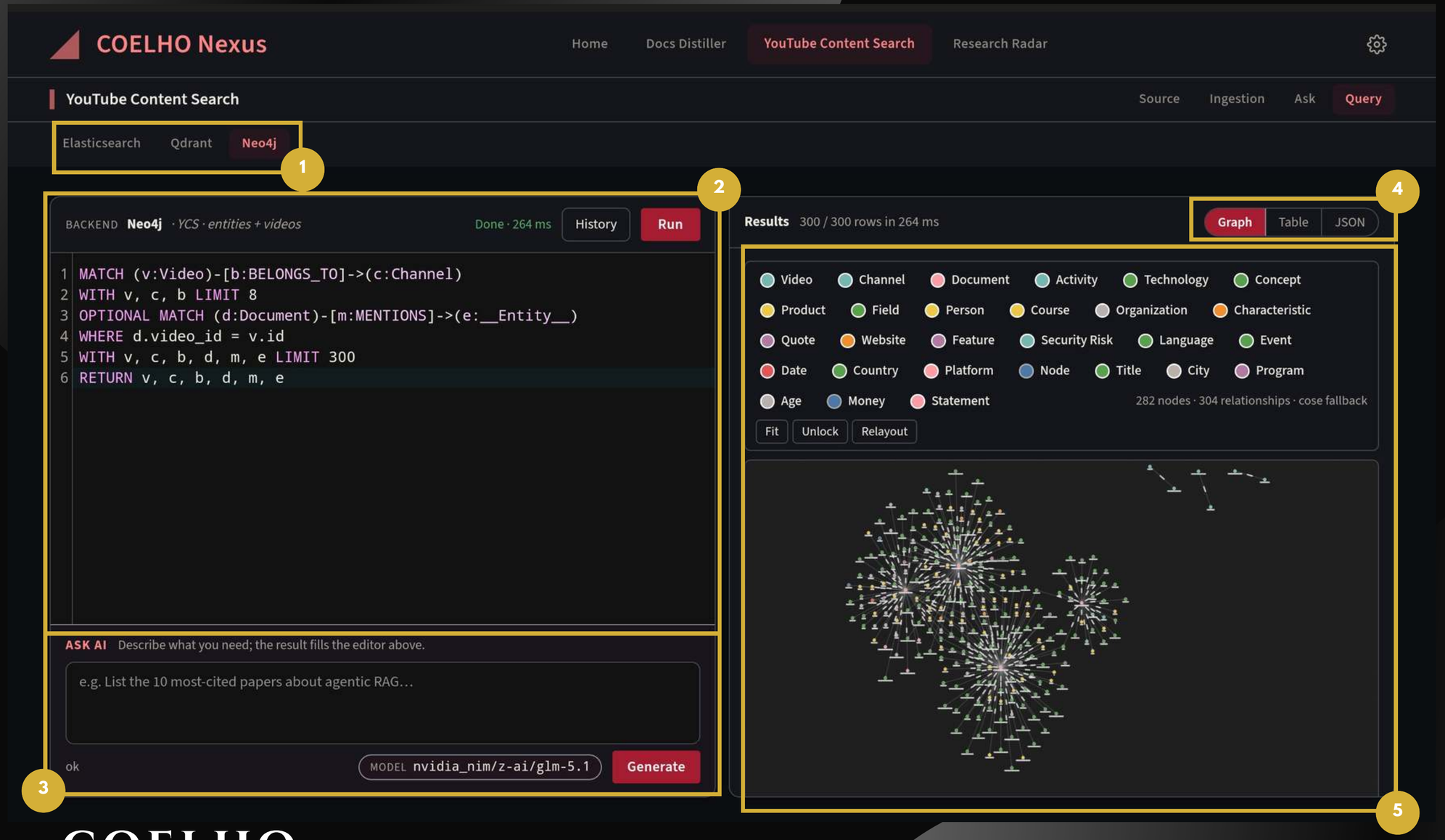
Deep mode delivers a structured answer with TL;DR, section headings, and inline numeric citations linking each claim to its source video. The assistant's thinking time and total generation duration are shown above the response.

2 Source Attribution Panel

Each cited source surfaces as a card with thumbnail, title, channel, and retrieval method (VECTOR = Qdrant dense search). Citations in the answer body map directly to the corresponding source card — full traceability from claim to transcript.

COELHO NEXUS

YOUTUBE CONTENT SEARCH - QUERY



COELHO NEXUS

1 Query Backend Selector

Switch between three query backends: Elasticsearch (full-text BM25), Qdrant (dense vector search), and Neo4j (Cypher graph queries). Each backend exposes its own editor tuned to that store's query language.

2 Backend Query Editor

Query editor that adapts to the selected backend — Cypher for Neo4j, DSL for Elasticsearch, or filter syntax for Qdrant. Run queries directly against the chosen store; execution time and result count shown inline.

3 AI Query Generator

Describe what you need in plain language and the LLM writes the correct query for the active backend — Cypher, Elasticsearch DSL, or Qdrant filters — filling the editor above. Zero boilerplate, instant executable output.

4 Result View Modes

Switch results between Graph (force-directed visualization), Table (row/column), and JSON (raw response). All three views update instantly from the same query execution — no re-run needed.

5 Query Result Visualization

Results rendered according to the active backend and selected view — a force-directed knowledge graph for Neo4j, ranked document rows for Elasticsearch, or vector similarity results for Qdrant. Node legend, relationship count, and layout controls available for graph views.

RESEARCH RADAR

Pipeline
Digest

COELHO
NEXUS

COELHO
NEXUS

RESEARCH RADAR

"Hundreds of papers published daily, most never read and fewer ever built — Research Radar finds the ones worth coding.

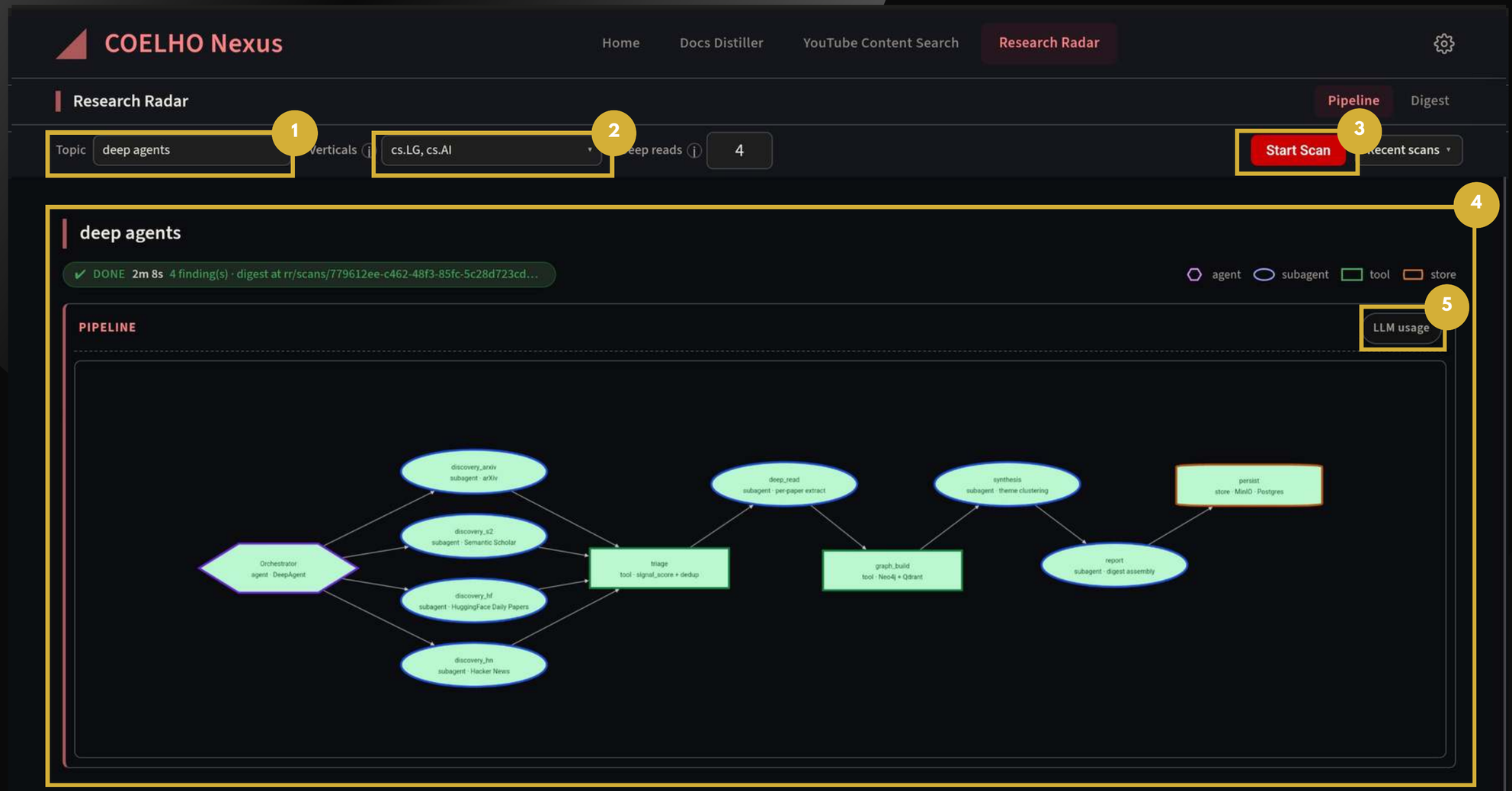
1. Continuously scans arXiv, Semantic Scholar, HuggingFace Daily Papers, and Hacker News for what's actually worth reading

2. A DeepAgents orchestrator reads each paper and extracts the core idea — what it would actually take to build this in Python

3. Every idea is ranked by real-world value — prioritizing what's implementable and profitable across LLMOps, agents, and quant finance

Why it matters: Turns a firehose of research nobody has time to read into a short list of ideas you could actually ship.

RESEARCH RADAR - PIPELINE



1 Research Topic Input

Define the research topic in plain language — the orchestrator uses it as the seed query across all discovery subagents. Supports any domain: AI, finance, math, security, or any intersection thereof.

2 Vertical Scope Selector

Pin the scan to specific academic verticals (e.g. cs.LG, cs.AI) to focus discovery on relevant arXiv categories. Narrows signal extraction to the exact subfields that matter for the topic.

3 Scan Controls

Start Scan launches the DeepAgents orchestrator; deep reads sets how many papers receive full per-paper extraction. Recent scans gives one-click access to prior runs by topic and timestamp.

4 DeepAgents Pipeline Graph

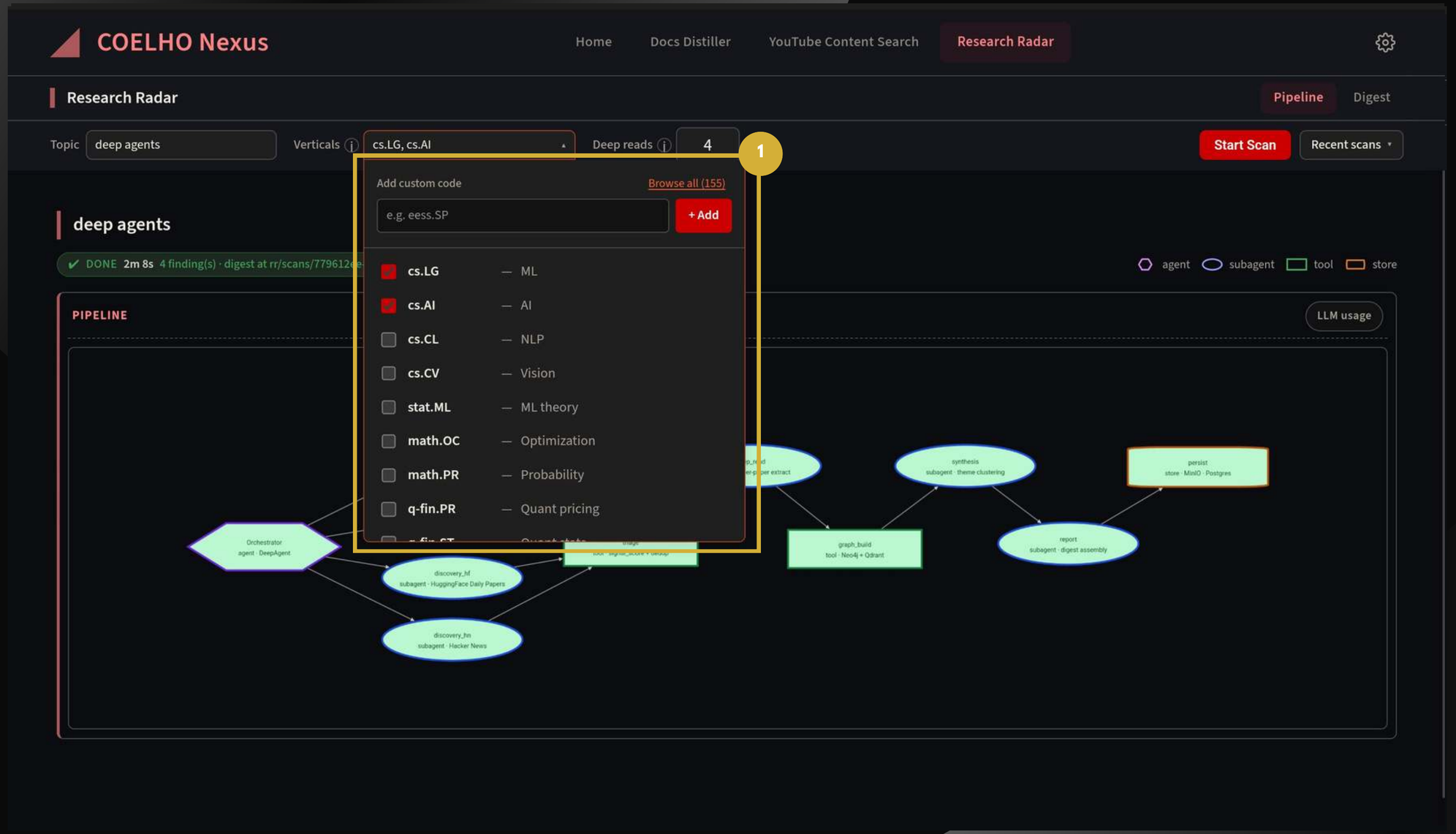
Live execution graph of the full Research Radar pipeline: Orchestrator → 4 parallel discovery subagents (arXiv, Semantic Scholar, HuggingFace Daily Papers, Hacker News) → triage → deep reads → synthesis → graph build → persist → digest assembly. Each node shows agent/subagent/tool type and real-time status.

5 LLM Usage Tracker

Per-run token consumption breakdown across all LLM calls in the pipeline — orchestrator planning, per-paper extraction, theme synthesis, and digest assembly. Full cost attribution across the DeepAgents execution.

COELHO NEXUS

RESEARCH RADAR - PIPELINE

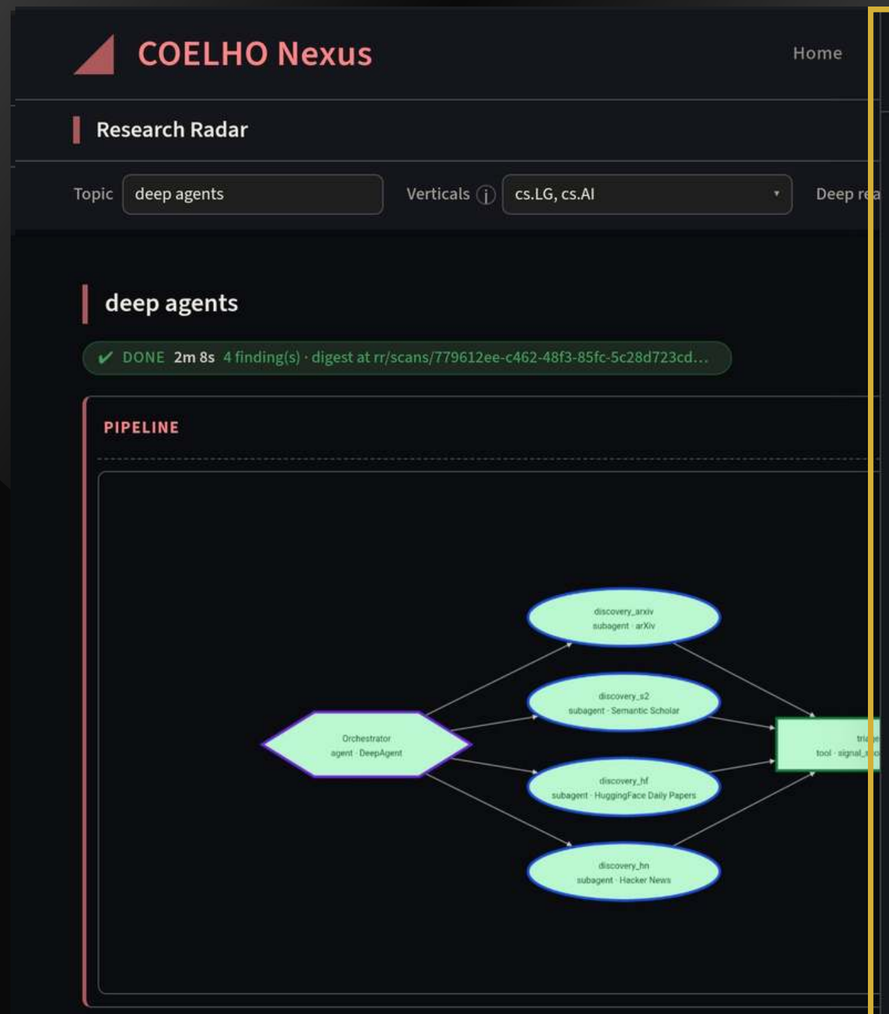


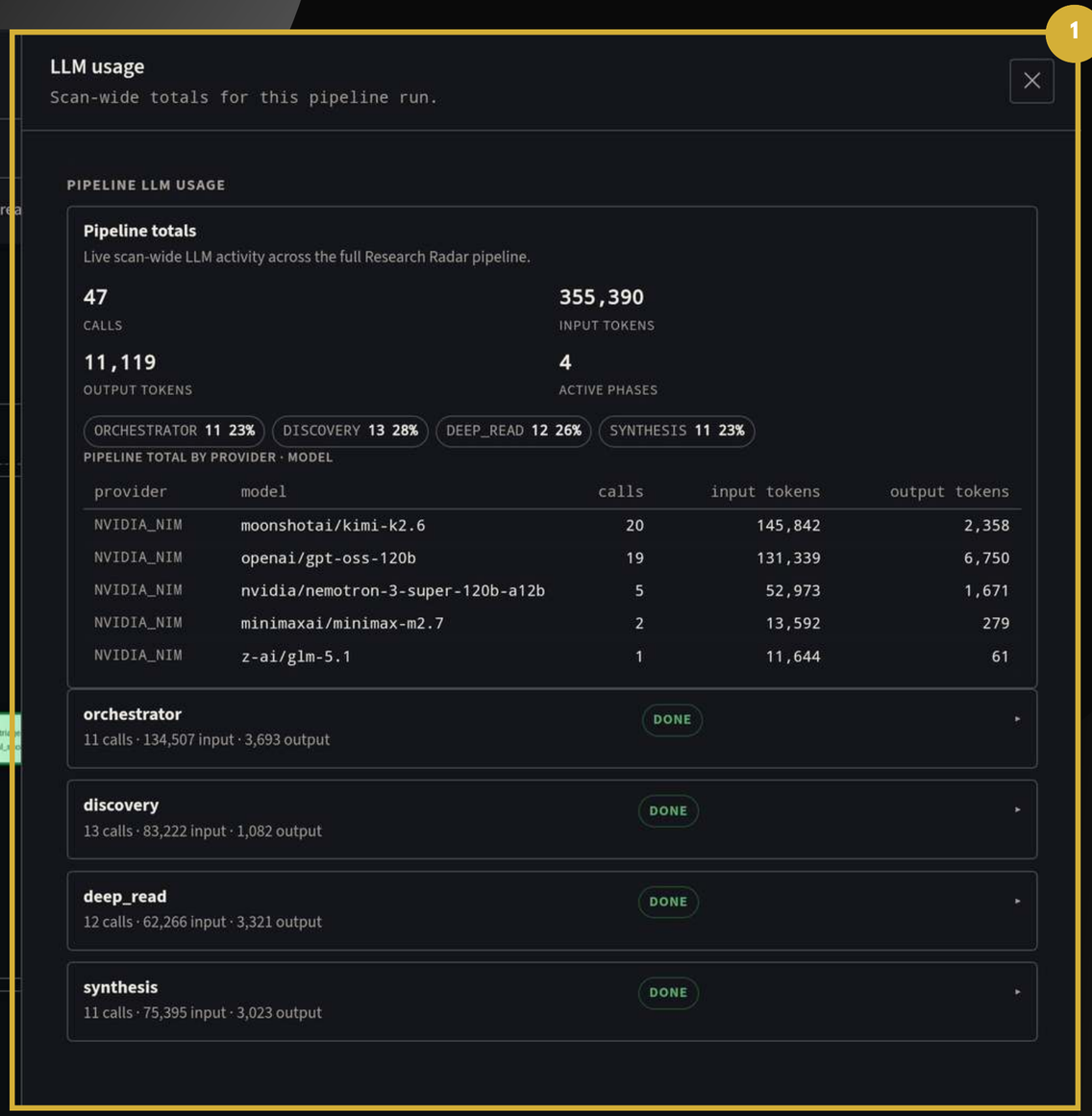
1 arXiv Vertical Picker

Multi-select dropdown covering all 155 arXiv categories — from cs.LG (ML) and cs.AI to math.OC (Optimization), q-fin.PR (Quant pricing), and beyond. Type any custom arXiv code to add it. Active verticals are checked in red and drive which papers the discovery subagents fetch.

COELHO NEXUS

RESEARCH RADAR - PIPELINE





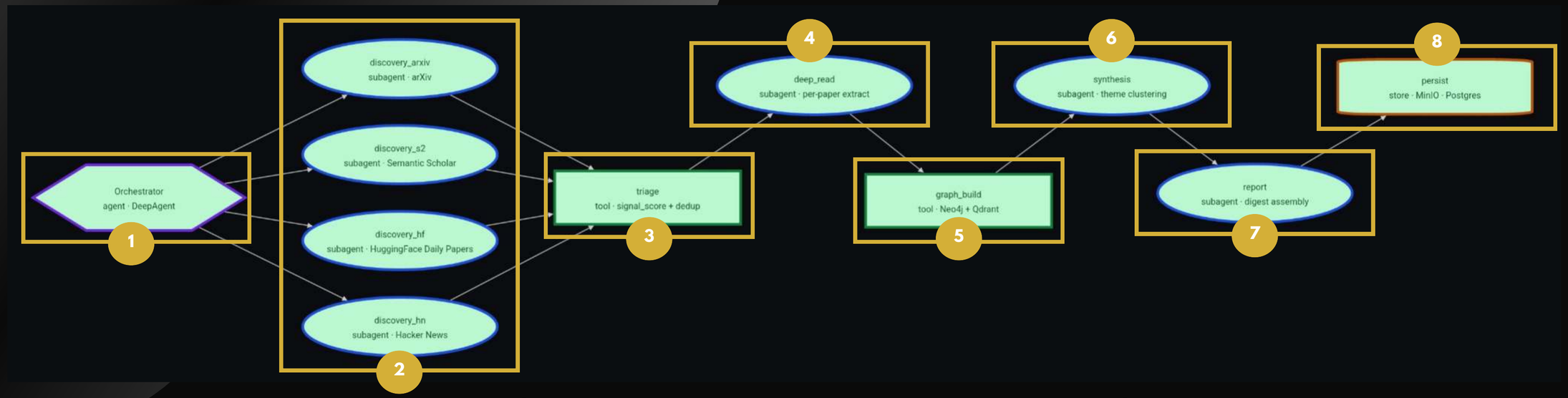
1 LLM Usage Breakdown

Scan-wide token accounting across every LLM call in the pipeline, broken down by provider/model via the LLM rotator and by pipeline phase — orchestrator, discovery, deep_read, and synthesis. Shows call counts, input and output tokens per phase, giving full cost attribution for any run.

COELHO NEXUS

RESEARCH RADAR - DEEPAGENTS

COELHO NEXUS



1 DeepAgent Orchestrator

FGTS-VA bandit-routed orchestrator built with create_deep_agent. Sequences the full pipeline: parallel discovery → triage → deep_read fan-out → graph_build → synthesis → digest. PhaseEnforcerMiddleware guarantees every fs artifact exists before the run closes; PhaseEventsMiddleware streams per-phase SSE to the UI.

2 4-Source Discovery Fan-Out

Four parallel DeepAgents subagents — arXiv, Semantic Scholar, HuggingFace Daily Papers, and Hacker News — each with isolated context and MCP-backed search tools. Skills (arxiv_query_shaping, rotator_etiquette) injected at build time. Results stashed to the virtual fs for triage.

3 Signal Triage & Dedup

Deterministic tool: normalize → cross-source dedup by arxiv_id → off-topic rerank gate (NIM nemotron-rerank-1b, drops bottom 50%) → source-diversity quota (min-1-per-source) → signal_score ranking → top-N written to fs. Idempotency guard prevents re-triage mid-scan.

4 Deep Read Fan-Out

Parallel subagent fan-out — one DeepAgents instance per top-N paper. Each extracts: problem, method, math, how-to-build, money angle, and confidence score. Isolated context per paper; augmented with the paper_extraction skill. Results written to fs per arxiv_id.

5 Graph & Vector Persist

Deterministic tool: embeds each abstract via llama-nemotron-embed-1b-v2 (NIM, 2048d), then upserts to Neo4j (paper metadata + relationships) and Qdrant (dense vectors). No LLM calls — pure I/O sink for both stores.

6 Theme Synthesis

Single subagent dispatched once after deep_read completes. Reads all extractions + triage ranking; produces 3–7 emerging themes spanning ≥2 papers, cross-paper convergence notes, and a 2–3 sentence executive summary. Augmented with the cross_paper_synthesis skill.

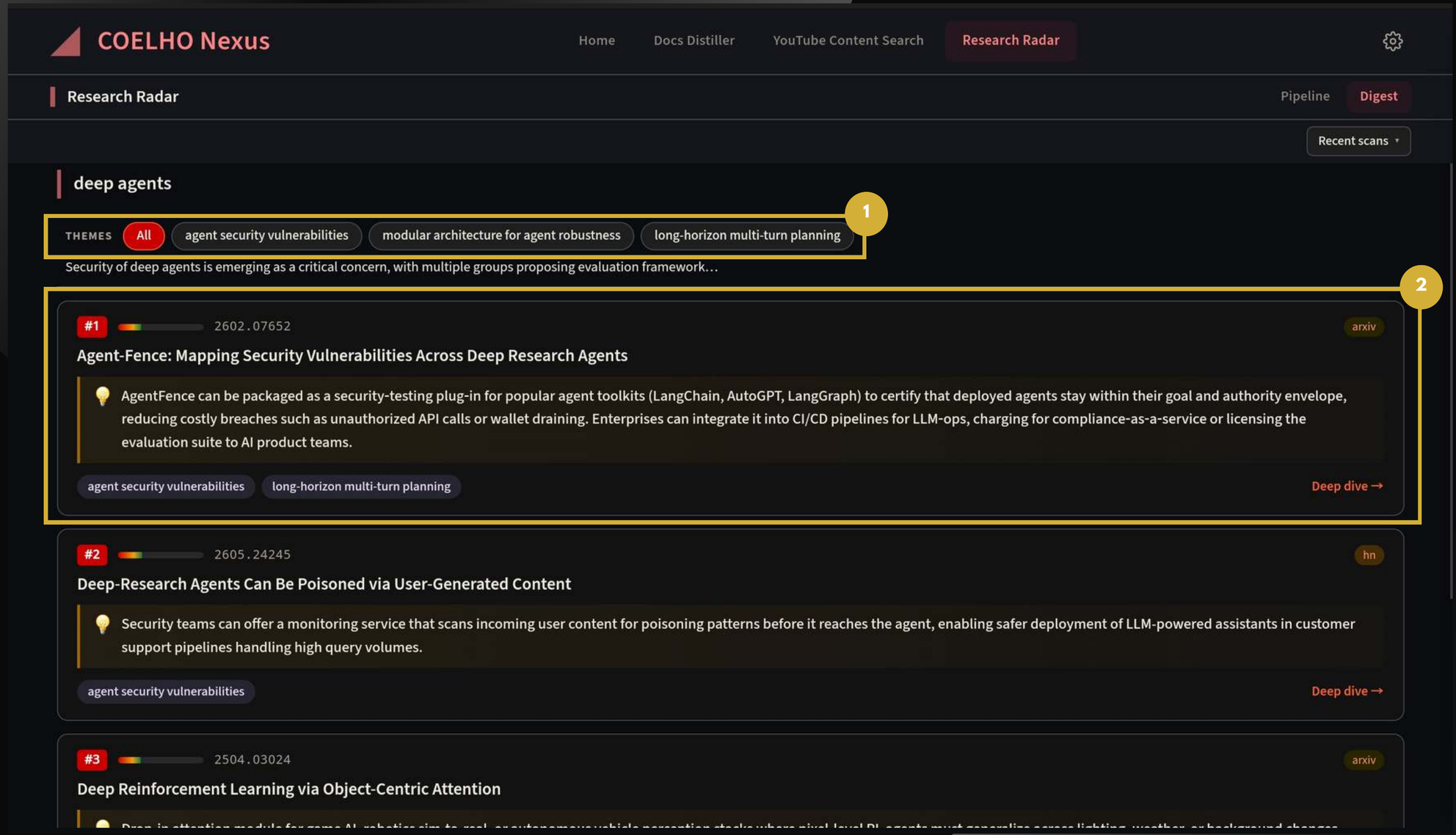
7 Digest Assembly

Reads synthesis themes, per-paper extractions, and triage ranking from the virtual fs. Assembles the final structured digest — findings, themes, and executive summary — as a JSON artifact ready for persistence.

8 Scan Persistence

Digest JSON and per-paper extraction artifacts persisted to MinIO under rr/scans/{scan_id}/. Scan metadata and finding records written to Postgres. Both stores bootstrapped idempotently at FastAPI lifespan startup.

RESEARCH RADAR - DIGEST



1 Theme Navigator

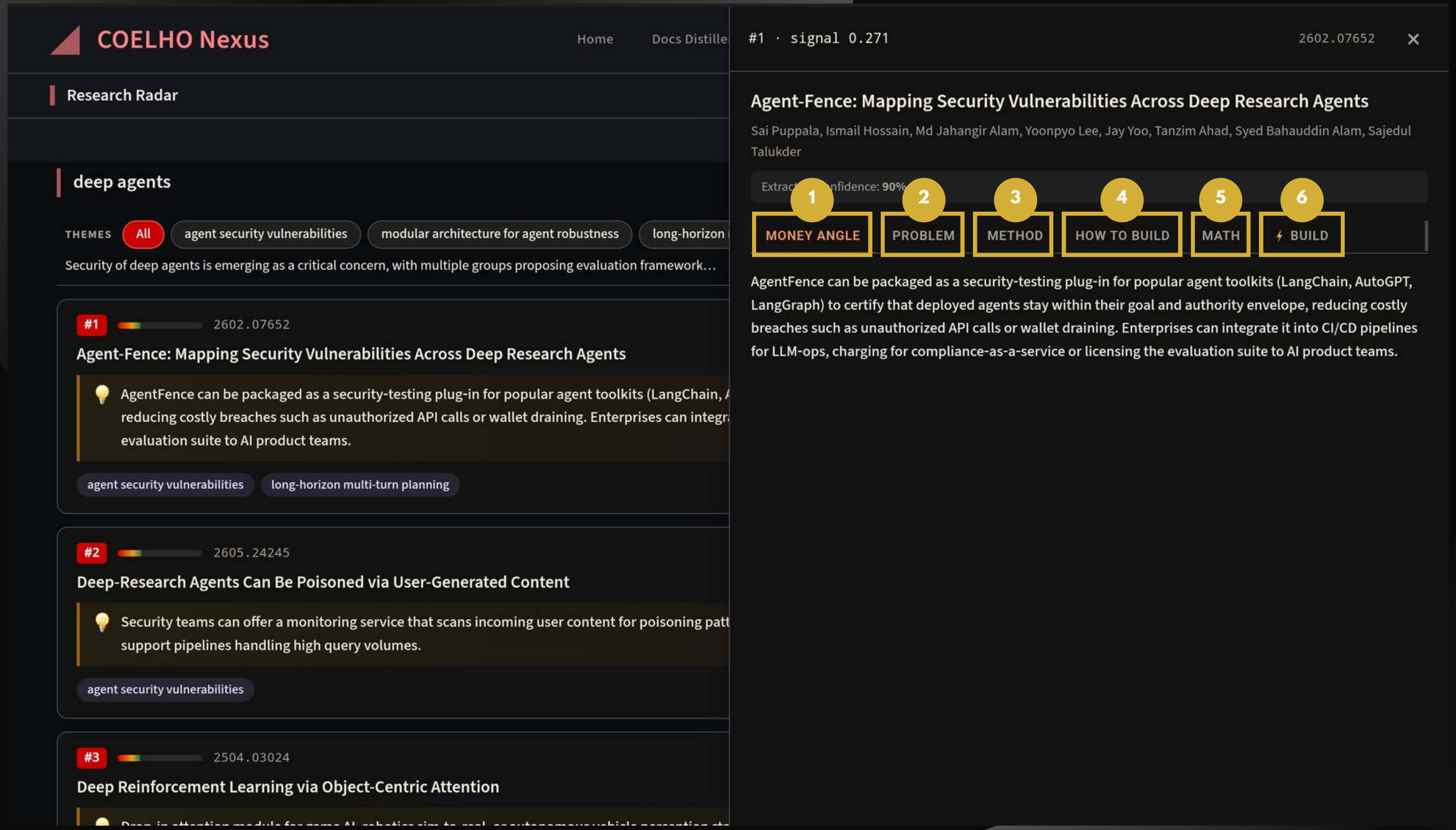
Synthesis-extracted themes from the scan — click any chip to filter the digest to papers tagged under that theme. The executive summary below gives the LLM's 2–3 sentence cross-paper overview of the full scan.

2 Ranked Paper Cards

Each card shows the paper's signal rank, arXiv ID, source badge (arXiv, Hacker News, Semantic Scholar, HuggingFace Daily Papers), title, LLM-extracted money angle, and assigned themes. Deep dive opens the full per-paper extraction: problem, method, math formulation, how-to-build, and confidence score.

COELHO NEXUS

RESEARCH RADAR - PIPELINE



1 Money Angle

LLM-extracted commercial applicability — how the paper translates into a product, service, or portfolio edge. Written by the deep_read subagent from the abstract and extracted fields.

2 Problem Statement

2–3 sentence extraction of the real-world gap the paper closes and why existing approaches fall short. The fastest signal for deciding if the paper is relevant to your domain.

3 Method Summary

4–6 sentence extraction of the paper's core algorithm, architecture, or technique — enough depth to evaluate implementation feasibility without reading the full PDF.

4 How to Build

Implementation notes from the deep_read subagent — which components to wire together, which libraries apply, and what the critical implementation choices are. Bridges the gap between understanding the method and actually building it.

5 Math Formulation

Key formulas in LaTeX, each annotated with its role in the algorithm. Extracted by the deep_read subagent to surface the mathematical core without hunting through the paper.

6 On-Demand Code Synthesis

Generates complete, runnable Python from the paper's 5 extraction fields on click. FGTS-VA bandit routes to the strongest coder in the pool; one round of Self-Refine enforces no TODOs or stubs. Output cached in MinIO per paper and prompt version.

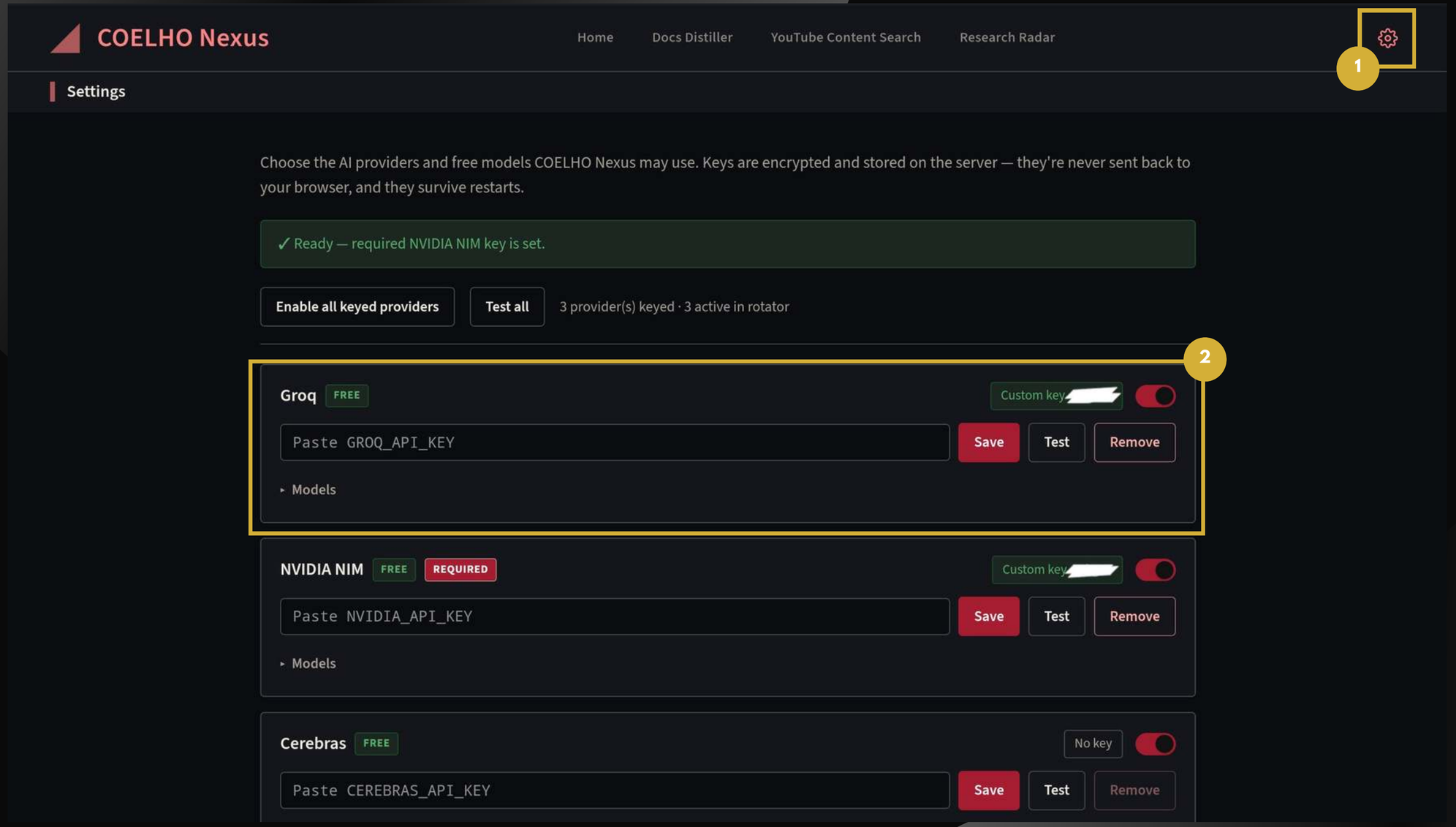
COELHO NEXUS

SETTINGS

COELHO
NEXUS

SETTINGS

COELHO NEXUS



1 Settings Gear

One click from anywhere in the app. Opens BYOK provider management — replacing what used to require editing Helm values and redeploying the cluster just to rotate a single API key.

2 Provider Key Rows

One row per provider — Groq, NVIDIA NIM, Cerebras, and more. Paste a free-tier key, save it, and test it live. The REQUIRED badge marks NVIDIA NIM as the canonical provider; the toggle takes any provider in or out of the rotator instantly. Keys are Fernet-encrypted at rest and never leave the server.

LLM ROTATOR — ZERO-COST ADAPTIVE INFERENCE

COELHO NEXUS

24-DIM CONTEXT VECTOR

Every LLM call is encoded as a 24-dimension feature vector — task type, chapter position, vault size, thinking-budget flag, time-of-day, and live per-provider error rates — so the bandit can tell a Synth call on a huge vault from a quick off-topic check, and route each differently.

FGTS-VA SELECTION (NEURIPS 2025)

Variance-aware Thompson sampling, published NeurIPS 2025: providers with noisy or thin reward history get sampled more optimistically, so one lucky response doesn't get over-trusted and one bad one doesn't get permanently written off.

COMPOSITE REWARD SIGNAL

A single success/fail flag can't tell a fast, well-formed response from a slow one that barely passed — so the reward blends four signals: success, schema validity, latency, and recall, each weighted by production impact. Failures are penalized by severity.

$0 PROVIDER POOL

Five free-tier providers — NVIDIA NIM, Groq, Cerebras, Mistral, and Gemini — spanning frontier open-weight models (Llama, Qwen, Nemotron, Kimi) to sub-second inference. Engineered for zero per-token cost and zero vendor lock-in — a deliberate architecture choice, not a budget constraint.

LLM ROTATOR — ZERO-COST ADAPTIVE INFERENCE

1. Initialization

$$A_a \leftarrow \frac{\lambda}{\max(0.1,\, p)}\, I_{24}$$

$$b_a \leftarrow A_a \cdot \frac{p}{24}\, \mathbf{1}_{24}$$

(p = benchmark_prior, λ = 1.0)

2. Context vector

$$\psi \in \mathbb{R}^{24}$$

3. Posterior mean (per arm)

$$\hat{\theta}_a = A_a^{-1} b_a$$

4. FGTS-VA variance estimate

$$\hat{\sigma}_a^2 = \max\left(\sigma_{\min}^2,\ \hat{\sigma}_{\mathrm{EWMA}}^2\right)$$

5. Variance-aware Thompson sample

$$\theta_s \sim \mathcal{N}\left(\hat{\theta}_a,\ \hat{\sigma}_a^2\, A_a^{-1}\right)$$

6. Score with feel-good bonus

$$s_a = \psi^\top \theta_s \ + \ \beta \sqrt{\psi^\top A_a^{-1} \psi}$$

7. Arm selection

$$a^* = \arg\max_{a \in \mathcal{A}} s_a(\psi)$$

8. Reward signal

$$r = \begin{cases} \mathrm{pen}(\mathrm{class}), & \text{call fails} \\ 0.30 + 0.25\, \mathbf{1}_{\mathrm{schema}} + 0.10 \cdot \mathrm{clip}(1 - \ell,\, -2,\, 2) + 0.25 \cdot \mathrm{clip}(\rho_{\mathrm{hash}},\, 0,\, 1), & \text{call succeeds} \end{cases}$$

(ℓ = latency_s / expected_latency_s)

$$\mathrm{pen}(\mathrm{class}) \in \{-0.10,\ -0.20,\ -0.40,\ -0.40,\ -0.50,\ -0.60,\ -0.80\}$$

rate_limit −0.10 · content_filter −0.20 · schema_invalid −0.40 · unknown −0.40 · server_error −0.50 · timeout −0.60 · auth_error −0.80

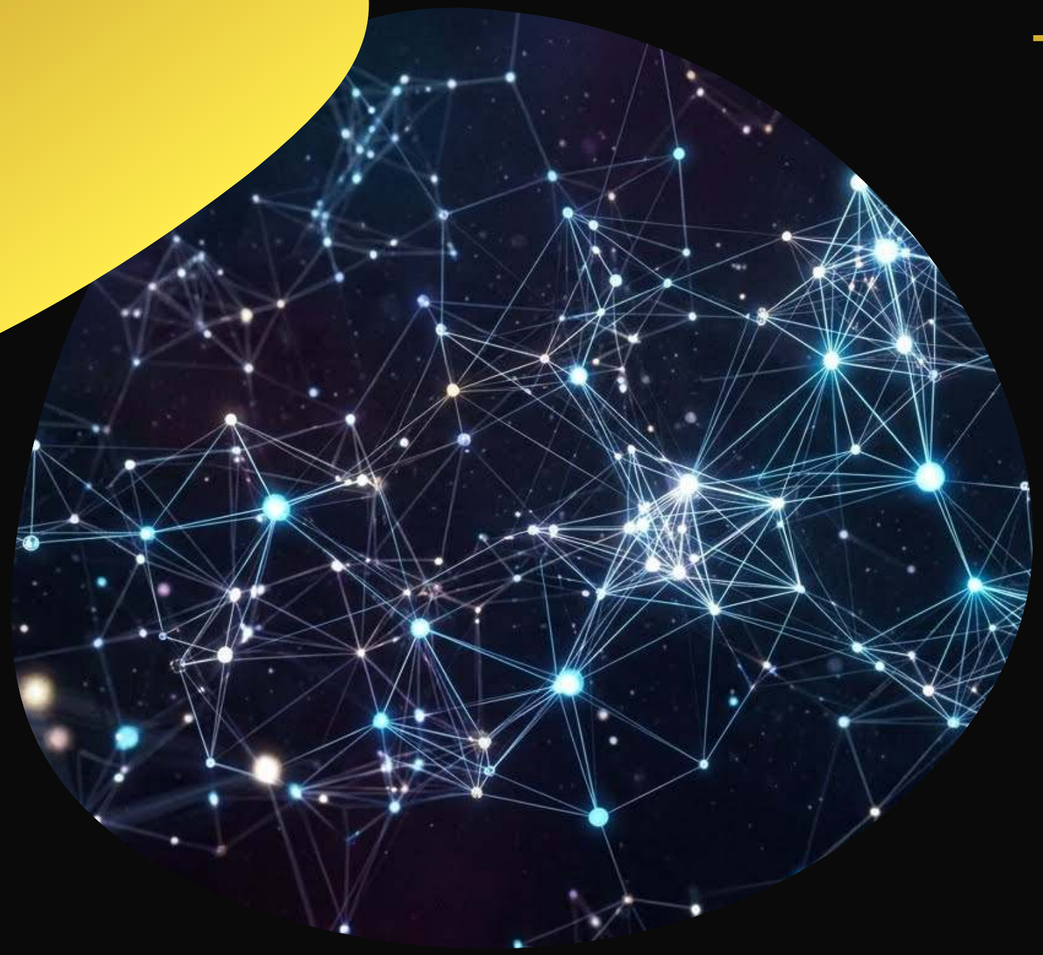
9. Forgetting update

$$\hat{\sigma}^2_{\mathrm{EWMA}} \leftarrow (1 - \alpha)\, \hat{\sigma}^2_{\mathrm{EWMA}} + \alpha \left(r - \psi^\top \hat{\theta}_a\right)^2$$

$$A_a \leftarrow (1 - \gamma)\, A_a + \psi \psi^\top$$

$$b_a \leftarrow (1 - \gamma)\, b_a + r\, \psi$$

COELHO NEXUS



01.

THREE MICROSERVICE ARCHITECTURE

Three purpose-built services compose into one coherent AI platform, sharing a single LLM rotator, six data stores, and a full OpenTelemetry observability pipeline.

FastAPI + Celery owns the compute layer — six isolated task queues route Docs Distiller ingestion, YouTube Content Search indexing, and Research Radar orchestration concurrently without contention.

FastHTML delivers a zero-JavaScript server-side frontend via HTMX-in-SSR, with Cytoscape.js pipeline graph visualization and SSE streaming for real-time lifecycle events.

FastMCP completes the surface layer, exposing four research discovery tools as MCP-native endpoints with per-tool token budgets, rate limiting, and Fernet-encrypted BYOK credential injection.

COELHO NEXUS

COELHO NEXUS

MAIN APPLICATIONS

Docs Distiller

Multi-tier document ingestion pipeline supporting Playwright CDP, sitemap, HTTPX, fallback, and manual modes. Transforms raw web content into structured knowledge through planner, resolver, and synth stages with full LangFuse observability.

YouTube Content Search

Full semantic search over YouTube transcripts with 21 subpackages covering caching, chunking, content extraction, embeddings, Elasticsearch indexing, Neo4j knowledge graphs, Qdrant vector search, RAG, and conversation management.

Research Radar

Autonomous research agent powered by DeepAgents framework. Scans arXiv, Semantic Scholar, Hacker News, and HuggingFace daily papers through an 8-phase orchestrator: Init, Discovery, Triage, Deep Read, Graph Build, Synthesis, Digest, and Persist.

MAIN APPLICATIONS

COELHO NEXUS

DOCS DISTILLER

4-Tier Ingestion Pipeline: llms_full.txt, llms_txt, sitemap, and httpx-crawl-with-Playwright-fallback modes. Planner-Resolver-Synth Workflow: index-graph chapter-outline planning, domain resolving, and multi-stage synthesis. LangFuse Session Tracking: every study_id tracked as a LangFuse session with full trace observability. MinIO Artifact Storage: page bodies, planner outputs, and synthesized content persisted in S3-compatible storage.

YOUTUBE CONTENT SEARCH

Full Semantic Search: complete pipeline from YouTube transcripts to dense vector embeddings via NVIDIA NIM. Neo4j Knowledge Graphs: extracts entities and relationships from video content into queryable graph structures. Qdrant Vector Search: 2048-dimensional embeddings for semantic similarity over processed transcripts. Elasticsearch Full-Text: hybrid search combining dense vectors with traditional full-text indexing.

RESEARCH RADAR

Autonomous DeepAgents Orchestrator: 8-phase pipeline — Init, Discovery, Triage, Deep Read, Graph Build, Synthesis, Digest, and Persist. 4 Discovery Sources: arXiv (Atom XML), Semantic Scholar (tldr/citations), HuggingFace daily papers, and Hacker News (Algolia). PhaseEnforcer Middleware: ensures all phases complete before termination; PhaseEvents Middleware streams per-node SSE events separately. Signal Scoring: an 8-term weighted composite — relevance, recency, citation velocity, influential-citation ratio, vertical fit, cross-tier buzz, code presence, and arXiv-ID presence.

CORE PLATFORM COMPONENTS

LLM Router & BYOK

FGTS-VA bandit (NeurIPS 2025) routes across ~20 free-tier model deployments — NVIDIA NIM, Groq, Mistral, Gemini, and Cerebras.
Per-arm variance-aware Thompson sampling with forgetting factor adapts routing in real time.
Cascading fallbacks on rate limits and errors.
Embeddings and reranking pinned to NVIDIA NIM. BYOK keys Fernet-encrypted in MinIO — never sent to the browser, survive restarts.

6 Data Stores

PostgreSQL: relational backbone with shared connection pool.
Neo4j: knowledge graphs for entity and relationship extraction across all three features.
Qdrant: 2048-dimensional vector search via NVIDIA NIM embeddings.
Elasticsearch: full-text and hybrid search over transcripts and document pages.
MinIO: S3-compatible artifact store for page bodies, planner outputs, and encrypted credentials.
Redis: caching, pub/sub, Celery broker, and bandit state persistence.

OpenTelemetry Observability

Dual-export pipeline: Alloy gRPC for the LGTM stack and LangFuse HTTP for LLM trace correlation.
BatchSpanProcessor tuned to q=6144, b=256, d=10s.
LangFuseSpanGate triple-gate filter drops non-domain spans.
Nine auto-instrumentations including httpx, redis, psycopg, and aio-pika.
Per-feature distinct trace patterns for Docs Distiller, YouTube Content Search, and Research Radar.

COELHO NEXUS

COELHO
NEXUS

02.

INFRASTRUCTURE AS CODE & MICROSERVICES

1 - CONTAINER ORCHESTRATION LAYER

2 - GITOPS & CI/CD LAYER

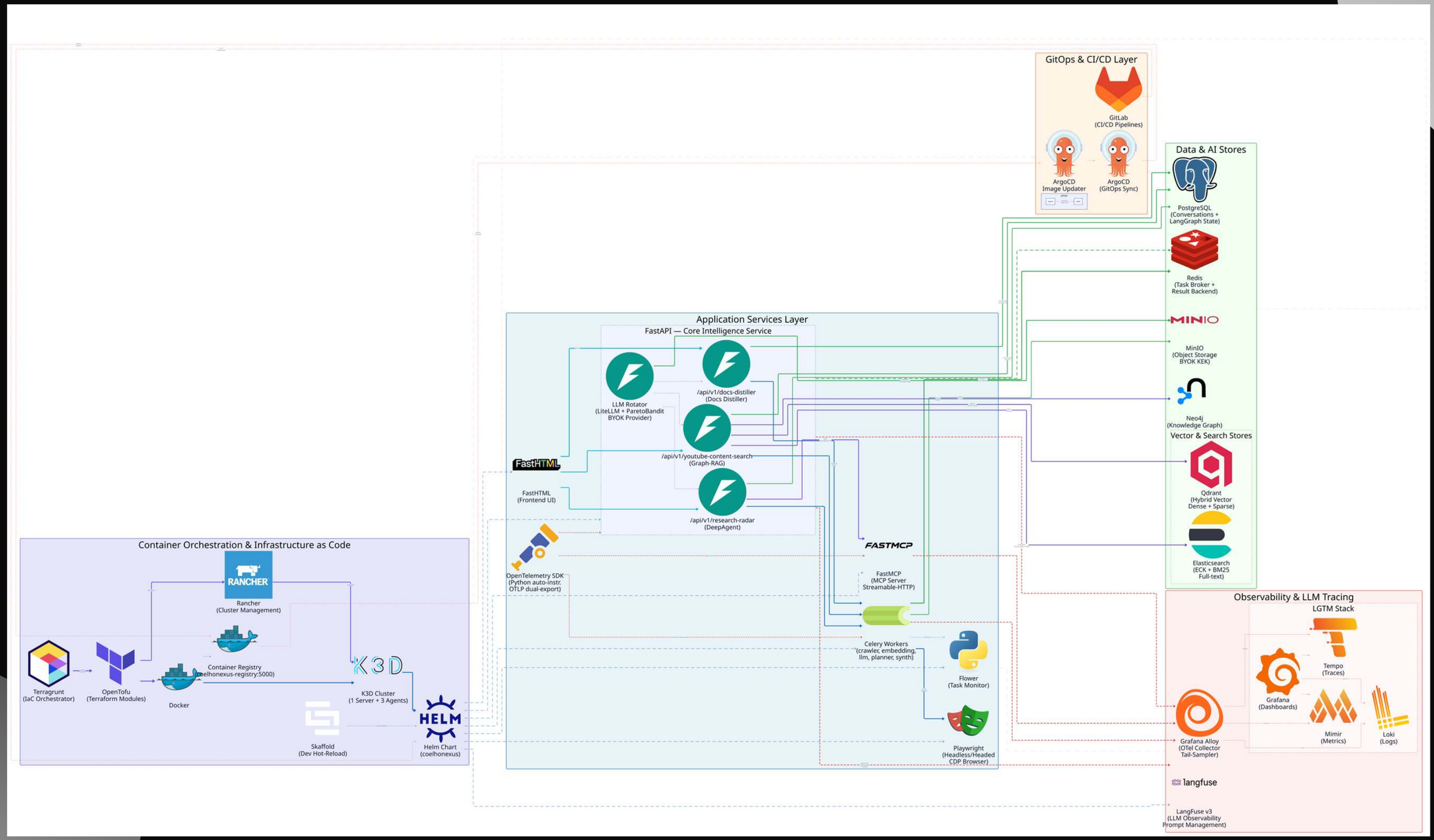
3 - DATA & STORAGE LAYER

4 - APPLICATION SERVICES LAYER

5 - OBSERVABILITY & MONITORING LAYER

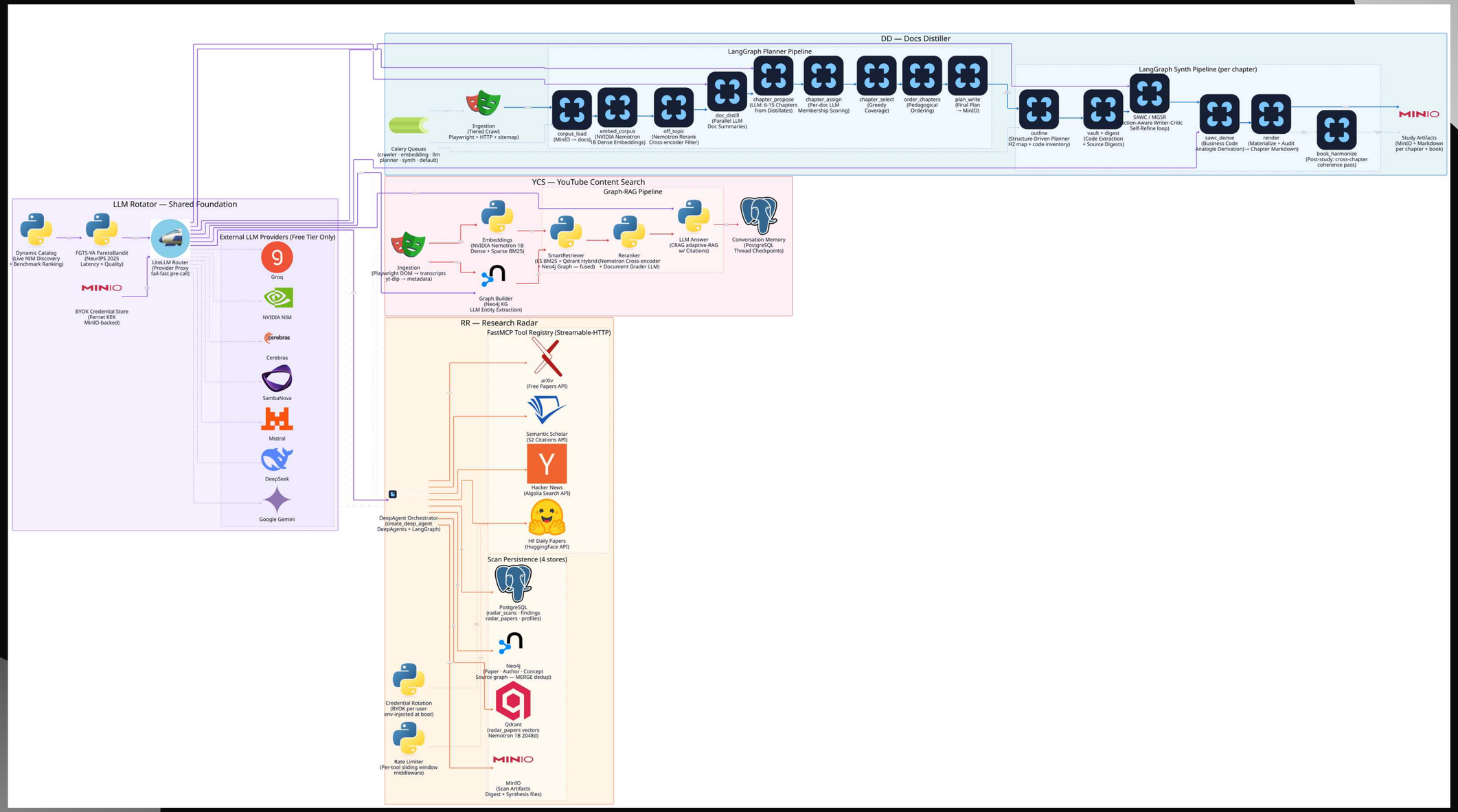
COELHO
NEXUS

PLATFORM ARCHITECTURE



COELHO
NEXUS

AI PIPELINE ARCHITECTURE



1 - CONTAINER ORCHESTRATION LAYER

TERRAGRUNT

- Orchestrates 18 OpenTofu modules across 5 dependency layers with DRY terragrunt.hcl configs
- Enforces provisioning order: k3d/Kubernetes cluster bootstrap → Rancher → ArgoCD → data stores → observability → apps
- Isolates remote state per module with automatic backend configuration, enabling safe parallel applies
- **Why Terragrunt:** The composable IaC orchestration layer shared across COELHO Nexus and all future COELHO projects

OPENTOFU / TERRAFORM

- Provisions the k3d cluster with 1 server + 2 agent nodes via local-exec wrapping the k3d CLI
- Manages Kubernetes, Helm, Rancher, and ArgoCD providers across 18 isolated state files
- Configures host volume mounts for persistent storage and a local container registry on port 5000
- **Why OpenTofu:** Open-source Terraform fork that eliminates vendor lock-in while maintaining full HCL compatibility — the execution engine underneath Terragrunt

KUBERNETES

- Orchestrates 20+ containerized services via declarative manifests — scheduling, networking, and lifecycle management
- Handles service discovery via ClusterIP and external access via NodePort, connecting frontend to backend seamlessly
- Enforces liveness/readiness probes, resource quotas, and automatic pod restart for self-healing operations
- **Why Kubernetes:** The industry-standard orchestration platform that provides the reliability, scalability, and operational patterns required for a production-grade microservice architecture

DOCKER

- Containerizes all custom services (FastAPI, FastHTML, FastMCP) with optimized single-stage Dockerfiles
- Provides BuildKit-accelerated layer caching for rapid image rebuilds during Skaffold hot-reload development
- Serves as the runtime foundation for k3d's Kubernetes nodes, running the entire cluster in containers
- **Why Docker:** Standardizes every service into portable, reproducible containers — a prerequisite for Kubernetes orchestration and consistent behavior across development and production

2 - GITOPS & CI/CD LAYER

K3D

- Runs lightweight K3s distributions inside Docker containers, replicating production Kubernetes locally
- Provides built-in container registry (coelhonexus-registry:5000) for local image distribution during CI/CD
- Supports 1 server + 2 agent nodes with host volume mounts and auto-restart via Docker unless-stopped policy
- **Why K3D:** Enables a production-identical Kubernetes environment on a single development machine with minimal resource overhead

HELM

- Packages FastAPI, FastHTML, FastMCP, Celery, and Flower into a single versioned chart with no external subchart dependencies
- values.yaml centralizes all service configuration: OTel endpoints, data store URIs, embedding model, and feature flags
- Layered values files (values.yaml + values-coelhonexus.yaml) enable cluster-specific overrides without duplicating config
- **Why Helm:** The standard Kubernetes package manager that turns the entire platform into a single, versioned, reproducible deployment unit

ARGOCD

- Syncs the coelhonexus Helm chart from GitHub master to the coelhonexus namespace with automated pruning and self-healing
- ArgoCD Image Updater polls coelhonexus-registry:5000 every 2 minutes via digest strategy for zero-downtime rollouts
- Two deployment targets: coelhonexus-dev (Skaffold, ports 23020–23039) and coelhonexus prod (ports 23000–23019)
- **Why ArgoCD:** GitOps-native deployment pipeline that makes Git the single source of truth for every running service

2 - GITOPS & CI/CD LAYER

SKAFFOLD

- Orchestrates local development with hot-reload Python file sync directly into running pods — no container rebuild
- Manages port forwarding for FastAPI, FastHTML, FastMCP, and Flower simultaneously on localhost
- sha256 content-based tag policy guarantees every build reflects current source — no stale image cache
- **Why Skaffold:** Eliminates the rebuild-redeploy cycle, enabling sub-second code changes in production-identical container environments

RANCHER

- Rancher Server v2.14.1 — single dashboard to view and manage every pod, service, and log across the cluster
- The daily operational front door — deployment health and failure debugging across 18 infrastructure modules, no kubectl required
- Complements Grafana/LGTM, not duplicates it: Rancher shows raw cluster state, LGTM shows application telemetry
- **Why Rancher:** Mission control for the cluster — what makes solo-operating a multi-service Kubernetes platform practical

3 - DATA & STORAGE LAYER

NEO4J

- Neo4j Community Edition + APOC Core powers dual knowledge graphs: YCS entities and RR paper networks
- LangChain LLMGraphTransformer populates both graphs via Bolt protocol (neo4j.neo4j.svc.cluster.local:7687)
- G1GC-tuned JVM with lazy heap commit, string deduplication, and ExitOnOutOfMemoryError for homelab stability
- **Why Neo4j:** The only store that captures relationships between entities — enabling graph-aware queries across video content and research papers that flat indexes cannot answer

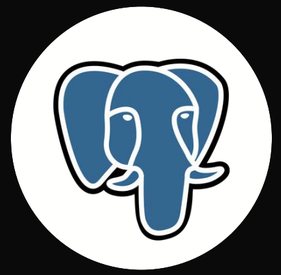
QDRANT

- Vector store for 2048-dimensional embeddings generated by NVIDIA NIM's llama-nemotron-embed-1b-v2 model
- Powers semantic similarity search for YCS transcript chunks and Research Radar paper deduplication across sources
- Chunk-level storage with payload metadata (video_id, channel, timestamps) enables filtered semantic search across YCS transcript collections
- **Why Qdrant:** Native vector search with API key authentication and ServiceMonitor telemetry — fastest path from embedding to nearest-neighbor retrieval

ELASTICSEARCH

- ECK Operator deploys Elasticsearch 8.18 with two YCS indexes: coelhonexus-youtube-metadata and coelhonexus-youtube-transcriptions
- File-realm authentication with ECK auto-generated TLS; ML and transform roles disabled for resource efficiency
- Kibana deployed via ECK Stack — index health, query debugging, and transcript search in one dashboard
- **Why Elasticsearch:** Enables hybrid search combining dense vector similarity with traditional full-text indexing — the query engine behind YCS Ask

3 - DATA & STORAGE LAYER



POSTGRESQL

- PostgreSQL 18 standalone serving LangFuse, YCS conversation history, Research Radar stores, and LangGraph AsyncPostgresSaver
- AsyncPostgresSaver persists Planner and Synth LangGraph state — enables mid-pipeline recovery without re-running completed nodes
- ServiceMonitor enabled — PostgreSQL query latency and connection pool metrics scraped by Alloy and exported to Grafana Mimir
- **Why PostgreSQL:** Relational backbone shared across LLM pipeline checkpointing, LangFuse observability, and application conversation history — one durable store for all stateful workloads

MINIO

- S3-compatible object store (standalone) with API on port 9000 and web console on port 9001
- Stores page bodies, planner outputs, and synthesized chapter artifacts in the coelhonexus bucket
- Hosts Fernet-encrypted BYOK LLM credentials — central object store across all pipeline stages
- **Why MinIO:** Production-grade S3 locally — eliminates external cloud storage dependencies while maintaining full AWS S3 API compatibility

REDIS

- Celery broker powering async task chains for DD ingestion, Planner, Synth, and YCS pipeline stages
- Persists FGTS-VA bandit CellState per provider/process and enforces single-flight pipeline locks (dd:planner:lock:{slug})
- Tracks ingestion progress counters and LLM rotator settings generation for cross-worker cache invalidation
- **Why Redis:** In-memory speed for sub-millisecond state reads across Celery workers, bandit updates, and progress polling

4 - APPLICATION SERVICES LAYER

FASTAPI

- Domain-driven async backend (DD, LLM, YCS, RR) with OpenTelemetry trace_id/span_id injected on every log line
- Provisions all 6 stores at lifespan: AsyncPostgresSaver, Redis, Neo4j, Elasticsearch, Qdrant, and MinIO bucket
- Celery task dispatcher for DD ingestion, Planner, Synth, and YCS pipeline chains
- **Why FastAPI:** Async-first, domain-isolated backend that co-provisions every data store and the full OTel pipeline at boot

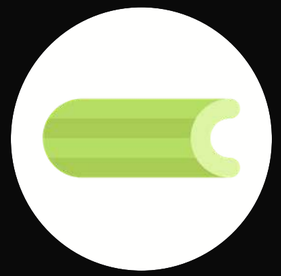
FASTHTML

- Server-side BFF rendering full-feature UIs for Docs Distiller, YCS, Research Radar, and Settings BYOK
- Reverse-proxies all /api/ calls to FastAPI — zero CORS configuration needed on the frontend
- Static assets served with Cache-Control: no-cache + ETag revalidation for instant Skaffold redeploy reflection
- **Why FastHTML:** Pure Python server-side rendering eliminates a JS framework while keeping the client snappy via HTMX + ES modules

FASTMCP

- Streamable-HTTP ASGI MCP server with 4 registered tools: arXiv, Semantic Scholar, HuggingFace Daily Papers, Hacker News
- RateLimitMiddleware + TelemetryMiddleware gate every tool call with OTel spans exported to Alloy
- Resources and prompts registered alongside tools — full MCP capability surface for Research Radar orchestration
- **Why FastMCP:** Exposes academic discovery tools as first-class MCP primitives consumable by any MCP-compatible AI client

4 - APPLICATION SERVICES LAYER



CELERY

- Distributed task queue executing DD ingestion, Planner, Synth, and YCS pipeline chains as async Celery chains
- OTel-instrumented via worker_process_init — every task emits spans with trace_id/span_id to Alloy
- Flower deployed alongside on port 23022 for real-time task monitoring, worker inspection, and queue visibility
- **Why Celery:** Decouples long-running LLM pipeline execution from FastAPI — workers scale independently of the API layer

PLAYWRIGHT

- Two separate browser pools run in the cluster — a full visible Chromium instance, and a lightweight invisible one
- The visible pool scrapes YouTube video transcripts straight from the page — YouTube blocks the same request when headless
- The invisible pool is Docs Distiller's last-resort tier, rendering JavaScript-only documentation sites simple fetching can't parse
- **Why Playwright:** Two of the three platform features depend on it for content with no public API — YouTube transcripts and JS-heavy docs

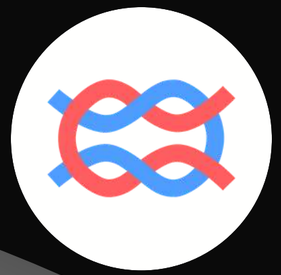
5 - OBSERVABILITY & MONITORING LAYER

GRAFANA ALLOY

- OTel-native collector receiving OTLP gRPC (:4317) and HTTP (:4318) from FastAPI and Celery workers across all 3 domains
- Routes all 3 telemetry signals from a single River config: traces → Tempo, logs → Loki, metrics → Mimir remote_write
- ServiceMonitor + PodMonitor discovery and namespace-scoped pod log tailing eliminate a separate Prometheus agent and log shipper
- **Why Grafana Alloy:** Single OTel-native pipeline unifying metrics, logs, and traces — replaces three separate collectors with one River config

GRAFANA

- External PostgreSQL backend for persistent dashboard and user storage — no bundled SQLite, survives pod restarts
- Datasource sidecar auto-imports Loki/Tempo/Mimir via ConfigMaps labeled grafana_datasource: 1 at startup
- Dashboard-as-code via Helm provisioning — all dashboards version-controlled alongside infrastructure
- **Why Grafana:** Single pane of glass unifying infrastructure metrics, distributed traces (including LangFuse LLM spans), and pod logs from the full LGTM stack



LANGFUSE

- Self-hosted LLM observability (sign-up disabled, single-org) tracking every LangGraph session across Docs Distiller, YCS, and Research Radar
- OTel integration correlates LLM spans (model, latency, token counts) with infrastructure traces in Grafana Tempo
- Redis-backed session queue with tuned socket timeouts for long-running Celery pipeline traces — no dropped spans on multi-hour runs
- **Why LangFuse:** The only store where LLM cost, quality, and latency are visible per-session — closes the observability gap that Mimir metrics alone cannot cover

5 - OBSERVABILITY & MONITORING LAYER

GRAFANA LOKI

- Monolithic SingleBinary (v3.7.1) with schema v13 + TSDB store — replaces deprecated BoltDB-shipper for index efficiency
- S3 backend on MinIO (loki-chunks + loki-ruler buckets) — pod logs persisted beyond pod lifetime with compactor-driven retention
- Datasource ConfigMap labeled grafana_datasource: 1 auto-wired to Grafana's sidecar on startup
- **Why Grafana Loki:** LogQL-queryable log store backed by the same MinIO already serving the platform — zero additional storage infrastructure

GRAFANA TEMPO

- Monolithic single-binary StatefulSet (v2.10.1) running ingester, querier, query-frontend, and compactor in one pod
- Receives OTLP traces from Alloy and correlates them with LangFuse LLM spans via shared trace_id across all 3 domains
- Trace blocks stored in MinIO with configurable retention and compactor-driven cleanup
- **Why Grafana Tempo:** End-to-end trace backend connecting FastAPI request spans to Celery task execution and LangFuse LLM calls

GRAFANA MIMIR

- Distributed mode (v3.0.4) with ingestion limits tuned to 100K series/s rate and 200K burst — handles cluster restart metric cascades
- Central MinIO backend with bundled MinIO + Memcached sub-charts disabled — single storage instance, leaner footprint
- Single-tenant, PromQL-compatible — all Grafana dashboards and alerting rules work without modification
- **Why Grafana Mimir:** Long-term metrics store with PromQL compatibility — replaces ephemeral Prometheus TSDB with durable MinIO-backed object storage

COELHO NEXUS

PYTHON ECOSYSTEM

LLM & AGENT ORCHESTRATION


LANGCHAIN


LANGGRAPH


DEEPAGENTS


LITELLM


LANGFUSE


RIVER

DATA & RETRIEVAL CLIENTS


QDRANT


NEO4J


ELASTICSEARCH


PSYCOPG (POSTGRESQL)


BOTO3 (MINIO/S3)

INGESTION & PARSING


CRAWL4AI


PLAYWRIGHT

YT-DLP

BEAUTIFULSOUP4


LXML

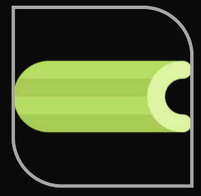

MARKDOWN-IT-PY

WEB FRAMEWORK & SERVICES


FASTAPI

FASTHTML

FASTMCP


CELERY

REDIS

CORE & OBSERVABILITY


PYDANTIC


NUMPY


CRYPTOGRAPHY


HTTPX


TENACITY

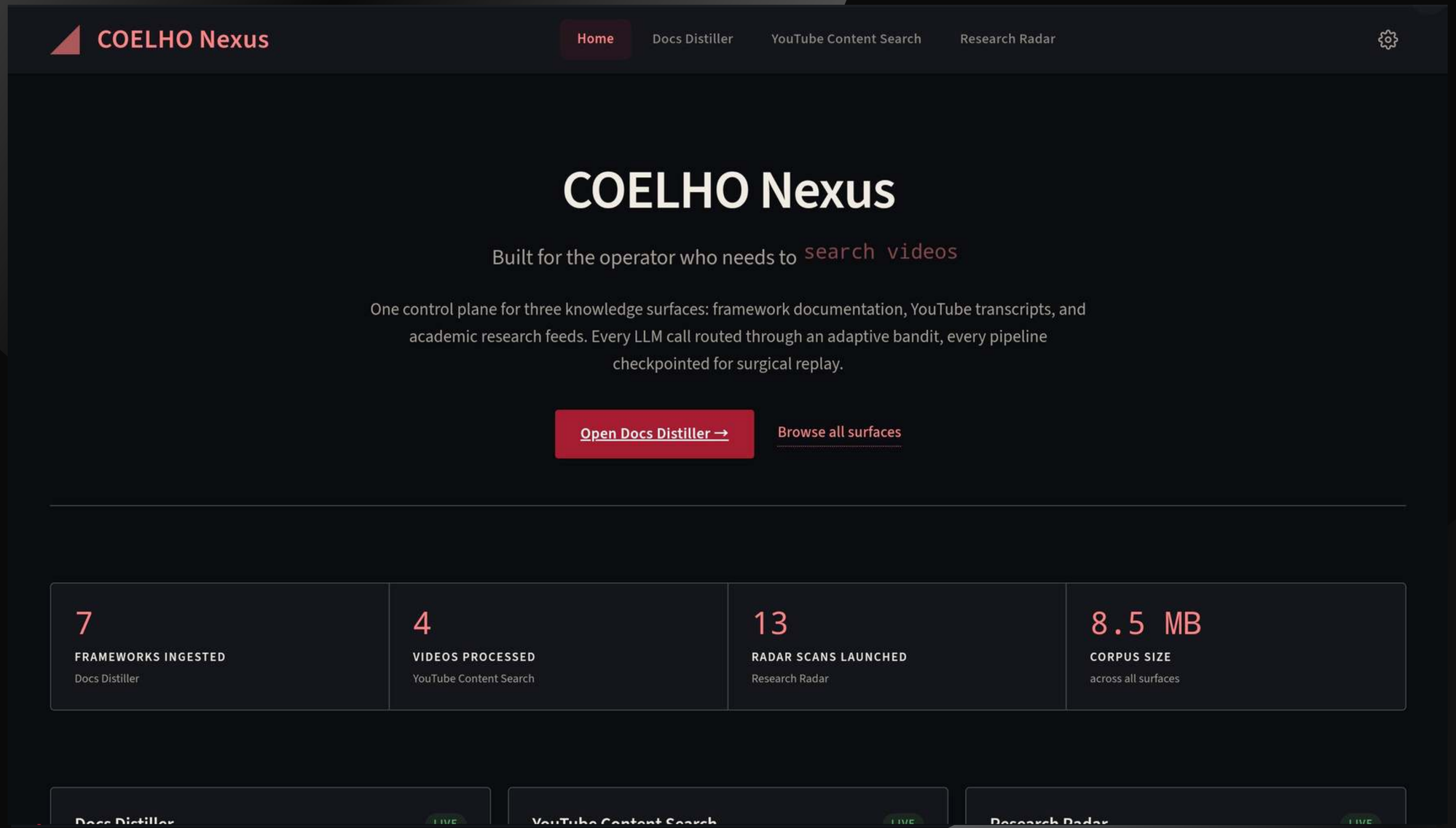

OPENTELEMETRY

THE MICROSERVICES ENGINE

A closer look at the services powering COELHO Nexus — the 4-service application layer (FastAPI, FastHTML, FastMCP, Celery), 6 data stores (PostgreSQL, Neo4j, Qdrant, Elasticsearch, MinIO, Redis), and the observability pipeline (OpenTelemetry → Grafana Alloy → Loki/Tempo/Mimir → Grafana, plus LangFuse for LLM traces).

COELHO NEXUS

MICROSERVICES - FASTHTML



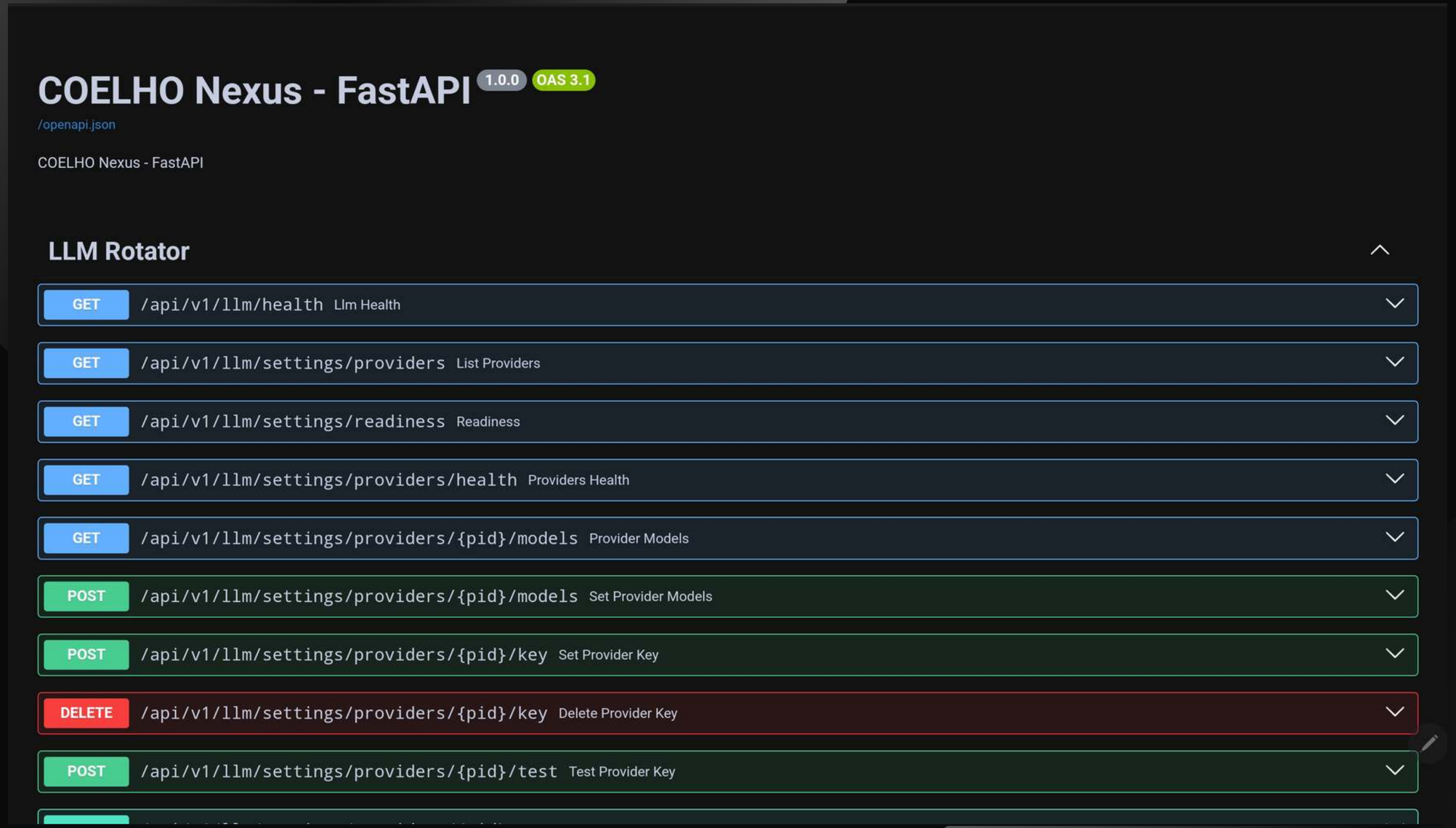
The Actual Homepage

Live numbers pulled from the running system, not placeholder stats. One control plane for all three features, with the LLM Rotator's settings one click away.

COELHO NEXUS

FastHTML
http://localhost:23026

MICROSERVICES - FASTAPI



Auto-Generated API Contract

Every domain — **Docs Distiller, YCS, Research Radar, LLM Rotator** — ships a live OpenAPI 3.1 schema at /openapi.json. 124 REST endpoints across 4 domains, always in sync with the code because it's generated from it, not hand-written.

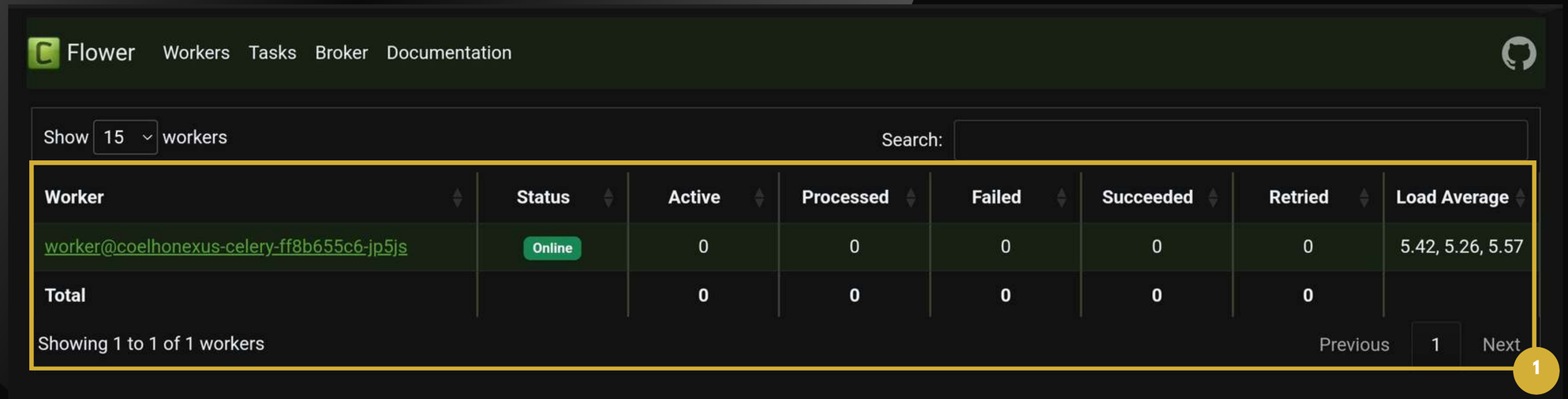
LLM Rotator Settings API

The exact endpoints behind the BYOK Settings page — list providers, check readiness, set or delete a key, test it live. The entire free-tier routing pool is reconfigurable without touching a deployment.

COELHO NEXUS

FastAPI
http://localhost:23024/docs

MICROSERVICES - CELERY (FLOWER)



1

6 Domain-Isolated Task Queues

Every domain — ingestion, planner, synth, YCS, Research Radar — runs on its own Celery queue, environment-suffixed so dev and prod share one Redis broker without stealing each other's jobs. Research Radar's long DeepAgents runs get a dedicated queue so they never block the rest of the pipeline.

1

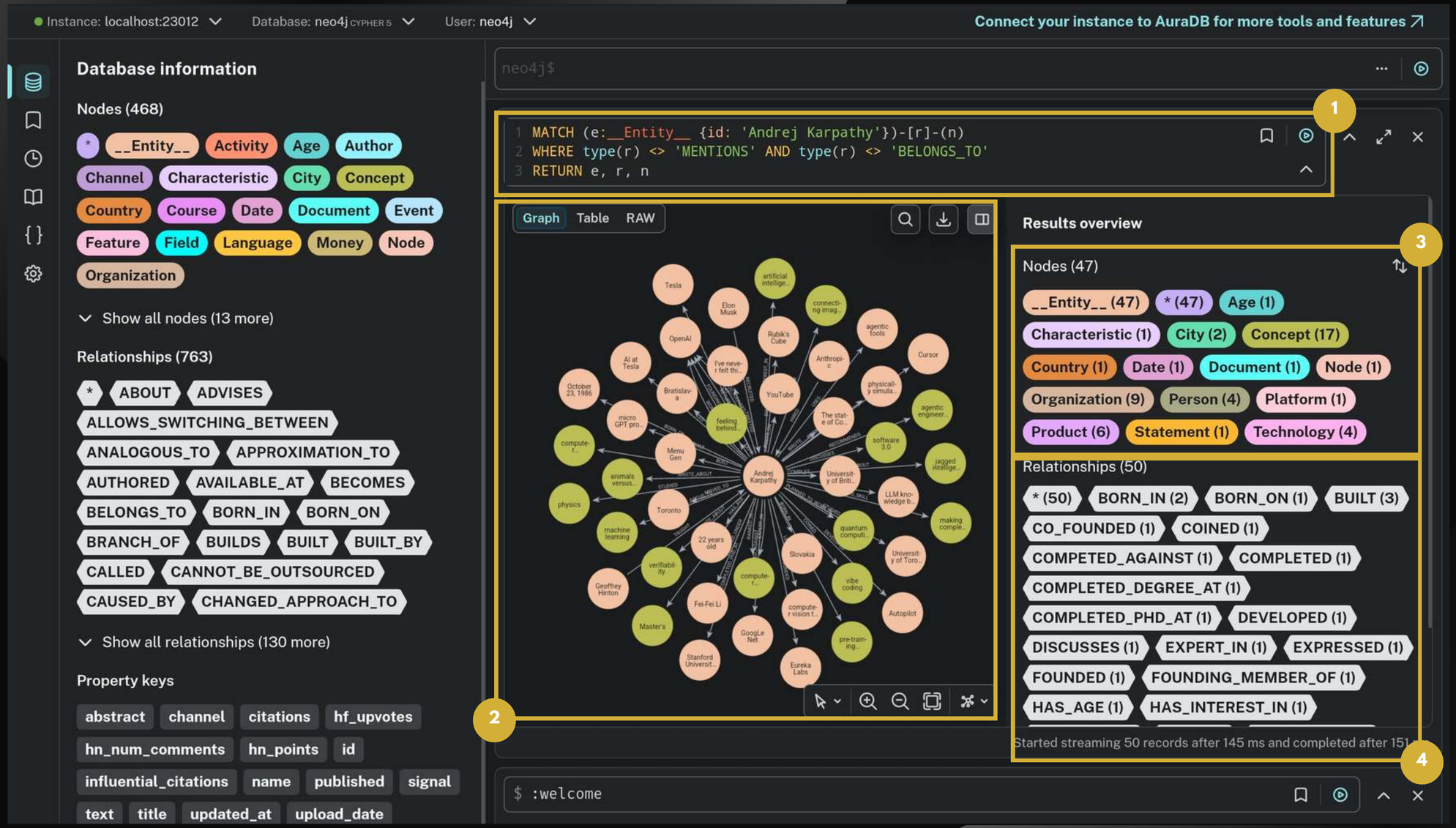
Live Worker Health, Zero Extra Code

The worker name is the actual running Kubernetes pod (coelhonexus-celery-ff8b655c6-jp5js), auto-discovered by Flower over the Redis broker. Active, Processed, Failed, Succeeded, Retried, and load average update in real time — full task-level observability Celery ships with out of the box.

COELHO NEXUS

Flower
http://localhost:23025

MICROSERVICES - NEO4J



1 Real Cypher, Real Answers

Every entity extracted from a transcript is queryable — ask what a video actually said about someone, instead of keyword-matching it.

2 One Entity, Merged Across Every Video

46 relationships — birthplace, education, career, people, concepts — extracted automatically by LangChain's LLMGraphTransformer. The same entity merges across videos instead of duplicating, so cross-video connections emerge as more gets ingested.

3 The LLM Infers Each Entity's Real Category

Person, Organization, Concept, Product, Country, Date — the LLM infers each entity's actual category, not just "thing mentioned".

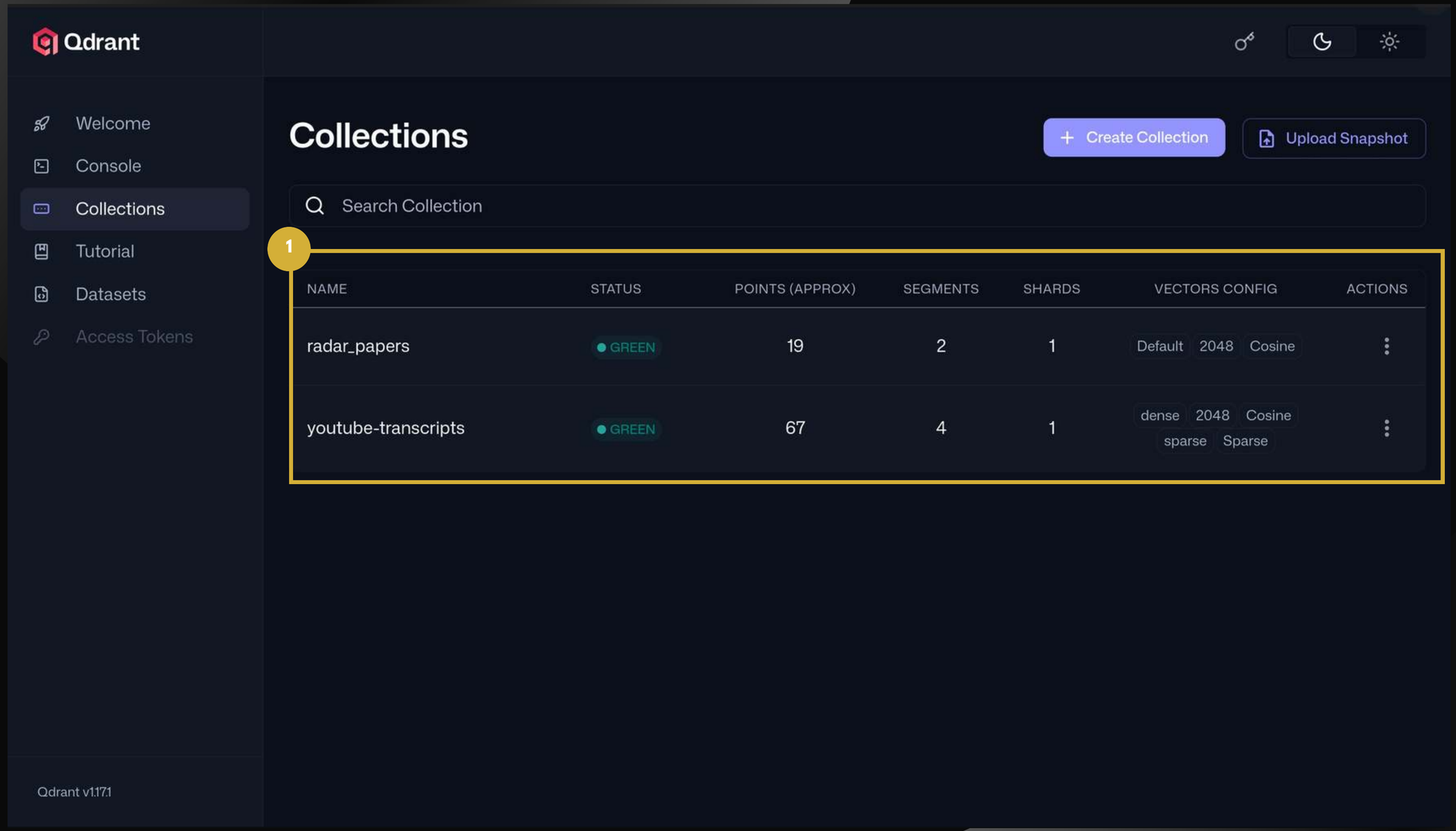
4 Typed Predicates Power Cross-Video Q&A

CO_FOUNDED, WORKED_AT, COMPLETED_PHD_AT — specific typed relationships, not generic "related to." This structure is what powers YCS's cross-video Q&A and Research Radar's author-paper network.

COELHO NEXUS

Neo4j
http://localhost:23001
Username: **neo4j**
Password: **neo4j-demo-password**

MICROSERVICES - QDRANT



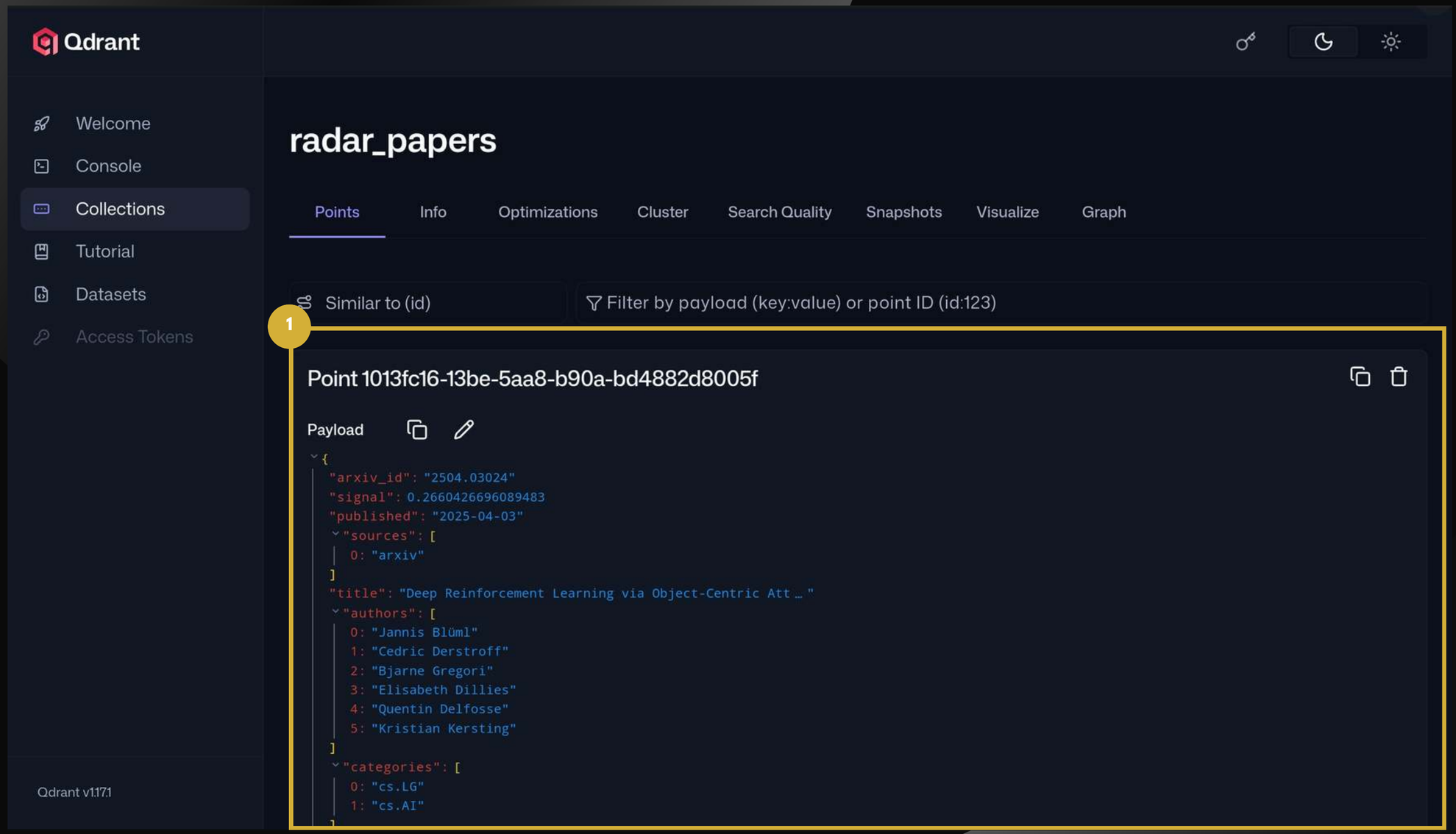
1 Two Collections, Two Features, Live

radar_papers backs Research Radar's paper dedup; youtube-transcripts backs YCS's semantic search — and it's hybrid, pairing dense 2048-dim embeddings with sparse vectors in one collection for better recall than similarity search alone.

COELHO NEXUS

Qdrant
http://localhost:23011/dashboard
API key: **qdrant-demo-api-key**

MICROSERVICES - QDRANT



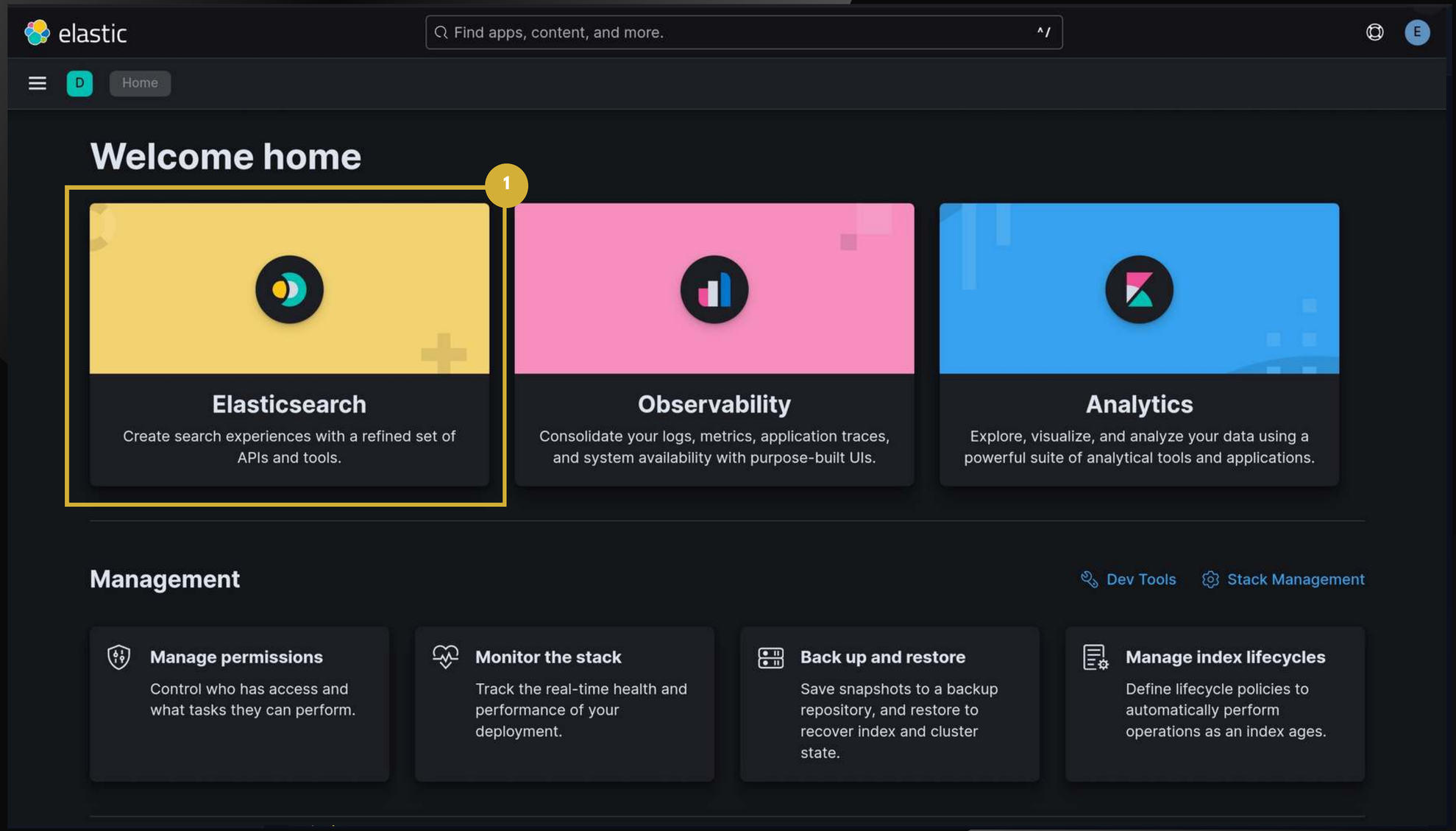
1 Every Payload Carries Its Own Relevance Score

eal arXiv metadata — title, authors, categories, publish date — plus signal, Research Radar's own computed relevance score that ranks papers before they ever reach the digest. The embedding is just one field in a much richer record.

COELHO NEXUS

Qdrant
http://localhost:23011/dashboard
API key: **qdrant-demo-api-key**

MICROSERVICES - ELASTICSEARCH (KIBANA)



1 Accessing Elasticsearch Data

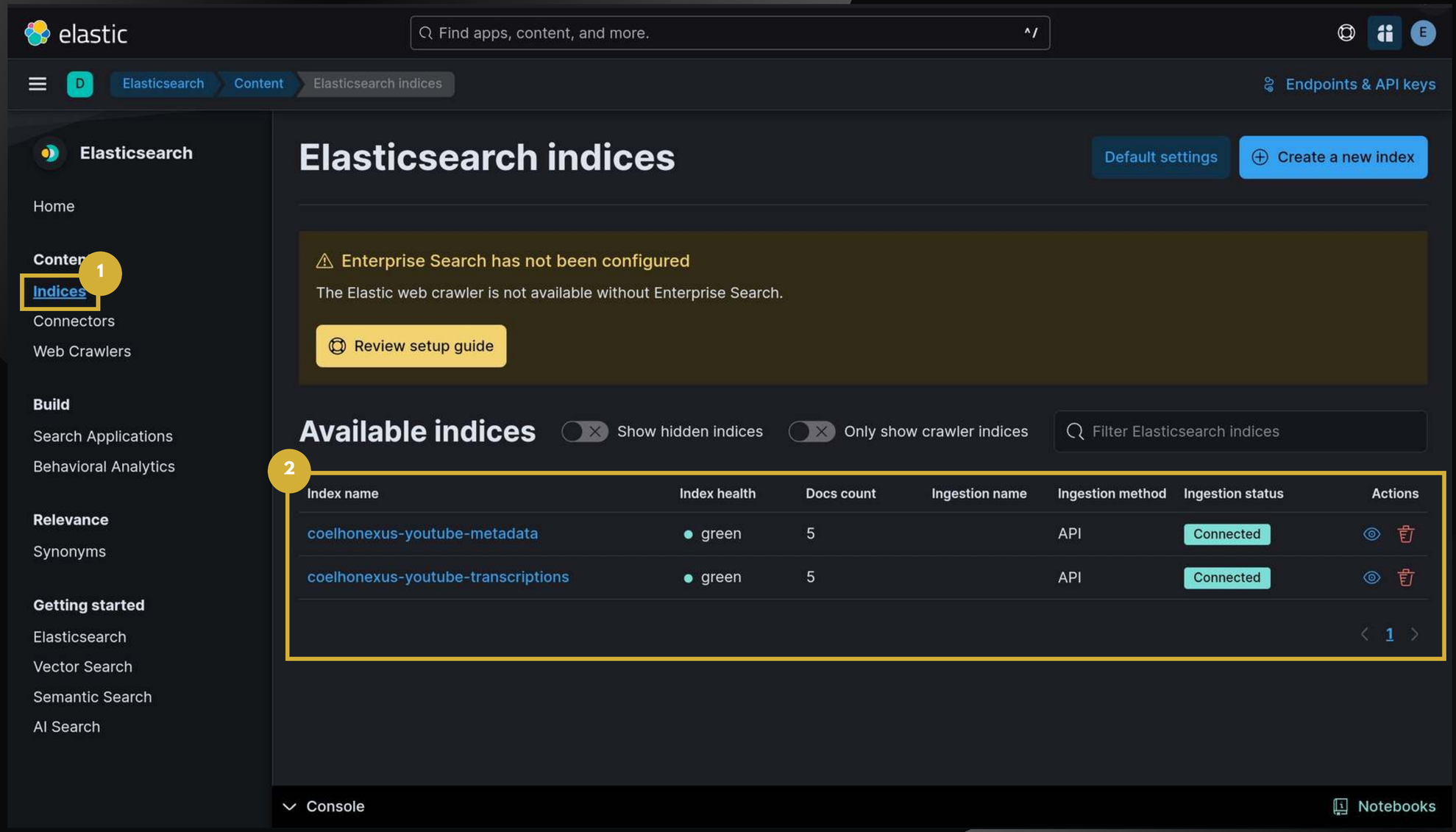
This tile is the entry point into Elasticsearch from Kibana — click it to browse, search, and manage the youtube-metadata and youtube-transcriptions indexes directly, no separate tool needed.

COELHO NEXUS

Kibana
https://localhost:23014
Username: **elastic**
Password: **coelhonexus-demo-password**

ElasticSearch (API)
https://localhost:23013
Username: **elastic**
Password: **coelhonexus-demo-password**

MICROSERVICES - ELASTICSEARCH (KIBANA)



1 Navigating to Elasticsearch Indices

Click Indices under Content in the left sidebar to see every index Elasticsearch is currently managing.

2 The Two Indices Powering YCS

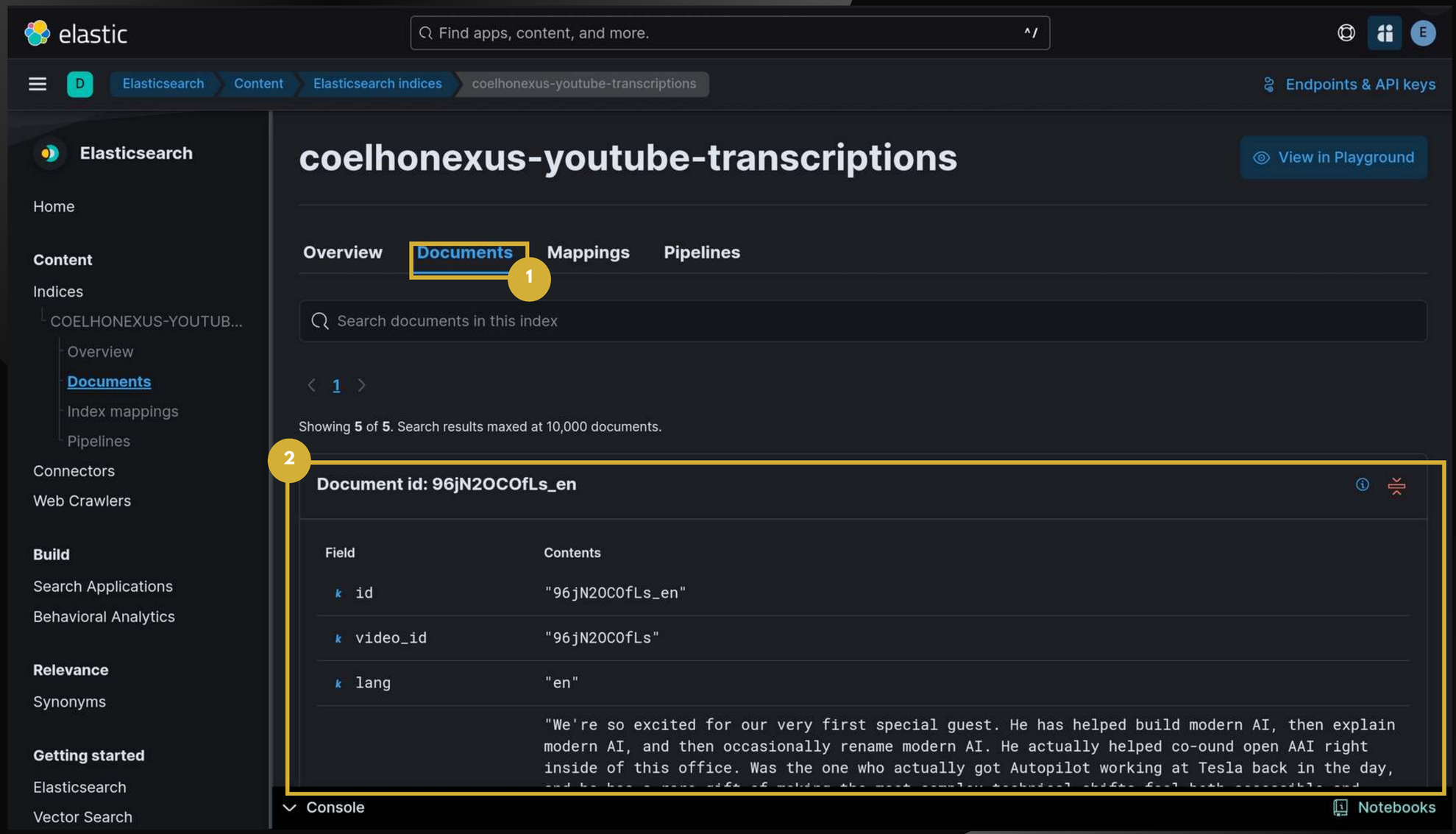
coelhonexus-youtube-metadata and coelhonexus-youtube-transcriptions — both green, both API-connected, both actively ingesting. This is the real data YCS Ask searches against.

COELHO NEXUS

Kibana
https://localhost:23014
Username: **elastic**
Password: **coelhonexus-demo-password**

ElasticSearch (API)
https://localhost:23013
Username: **elastic**
Password: **coelhonexus-demo-password**

MICROSERVICES - ELASTICSEARCH (KIBANA)



1 Click Documents to See What's Actually Stored

The Documents tab lists every record in this index — real ingested transcript chunks, not just index metadata.

2 This Is What YCS Ask Actually Searches Against

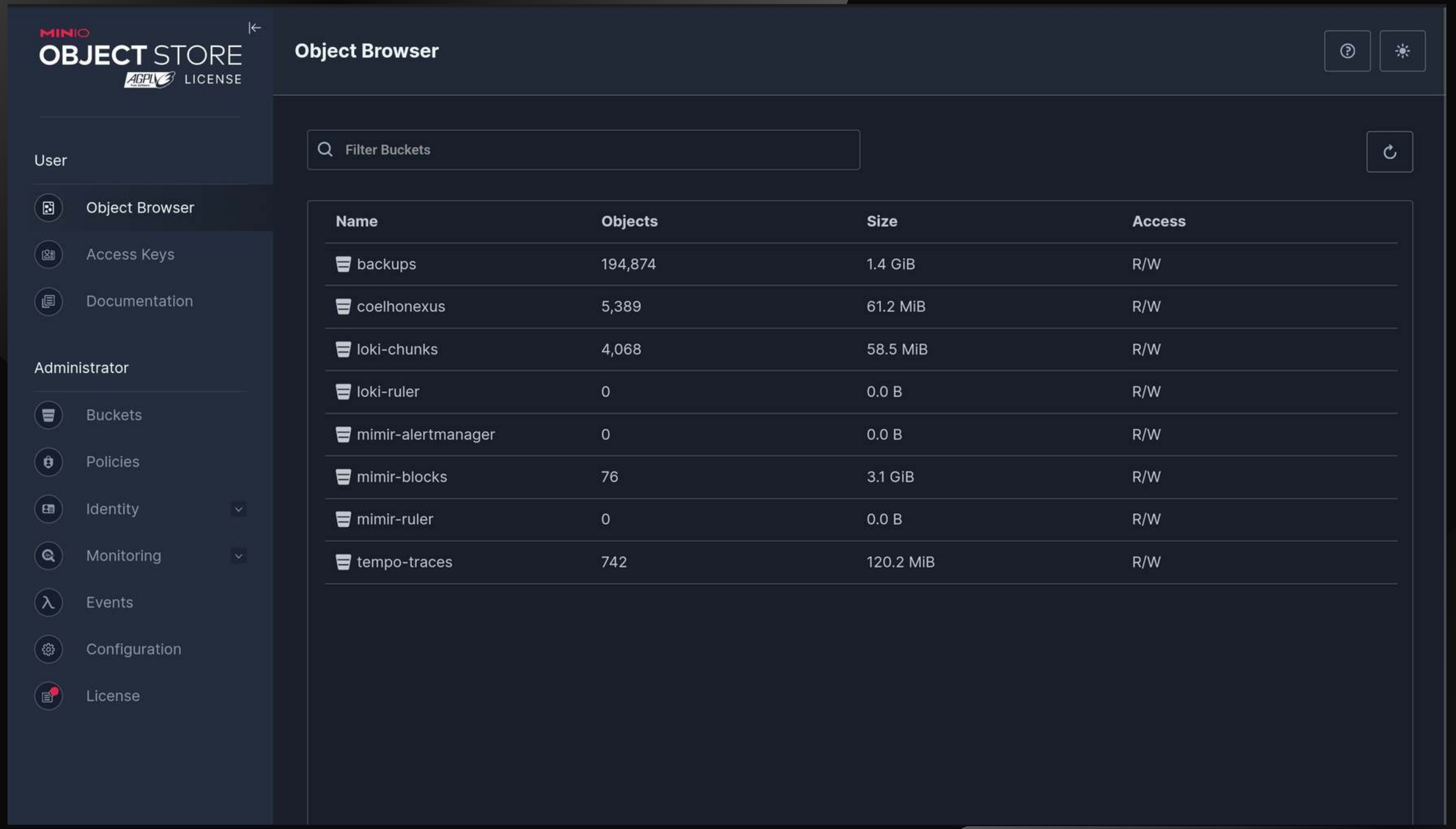
id, video_id, lang, and the actual spoken text — this is what YCS Ask searches against: real transcript content, chunked and indexed per video.

COELHO NEXUS

Kibana
https://localhost:23014
Username: **elastic**
Password: **coelhonexus-demo-password**

ElasticSearch (API)
https://localhost:23013
Username: **elastic**
Password: **coelhonexus-demo-password**

MICROSERVICES - MINIO



1 One Object Store, Every Backend's Data

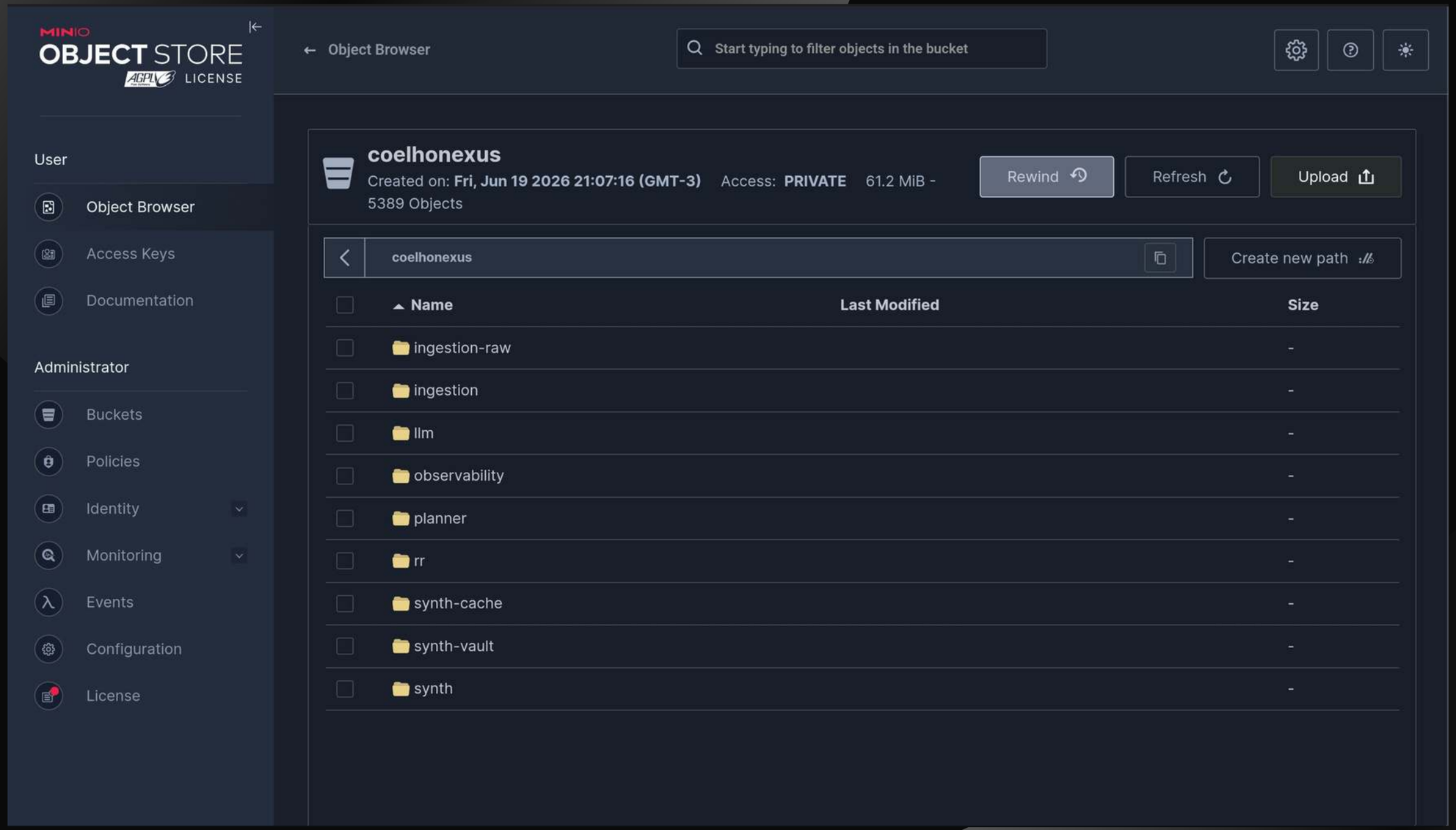
Loki, Tempo, and Mimir don't use traditional databases — they chunk logs, traces, and metrics into S3-compatible objects, decoupling storage from compute so each scales independently. The same MinIO instance also holds coelhonexus — the application's own working data, covered on the next slide — proving one object-storage abstraction serves both the observability stack and the product itself.

COELHO NEXUS

MinIO (Console UI)
http://localhost:23016
Username: **minioadmin**
Password: **minioadmin**

MinIO (S3 API)
http://localhost:23015
Username: **minioadmin**
Password: **minioadmin**

MICROSERVICES - MINIO



1 **One Bucket, Organized by Feature**

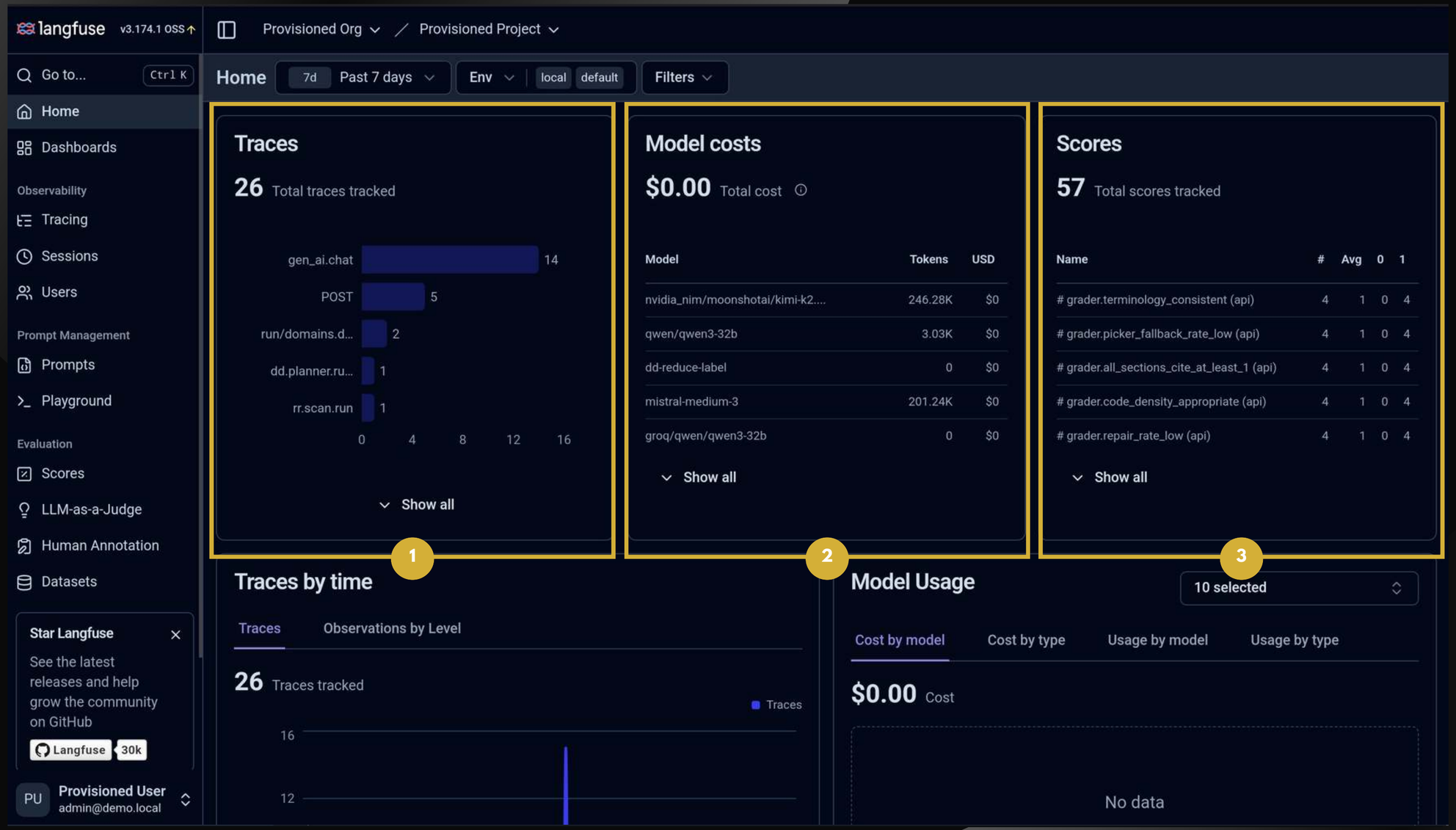
ingestion, planner, and synth trace the Docs Distiller pipeline start to finish; rr holds Research Radar's artifacts; llm is the encrypted BYOK credential store powering the LLM Rotator. Nothing dumped flat into one namespace — every feature gets its own prefix, with synth-cache and synth-vault separating ephemeral working data from what's meant to persist.

COELHO NEXUS

MinIO (Console UI)
http://localhost:23016
Username: **minioadmin**
Password: **minioadmin**

MinIO (S3 API)
http://localhost:23015
Username: **minioadmin**
Password: **minioadmin**

MICROSERVICES - LANGFUSE



1 Every Signal Source, One Timeline

Auto-instrumented OTel spans (gen_ai.chat) sit next to hand-written ones (dd.planner.run, rr.scan.run) in the same view. Every layer of the stack reports here, not just raw LLM calls.

2 Real Usage, Zero Dollars

Hundreds of thousands of tokens processed across multiple models, and every single one reads $0.00 — this is the free-tier rotator's actual bill, not a theoretical claim.

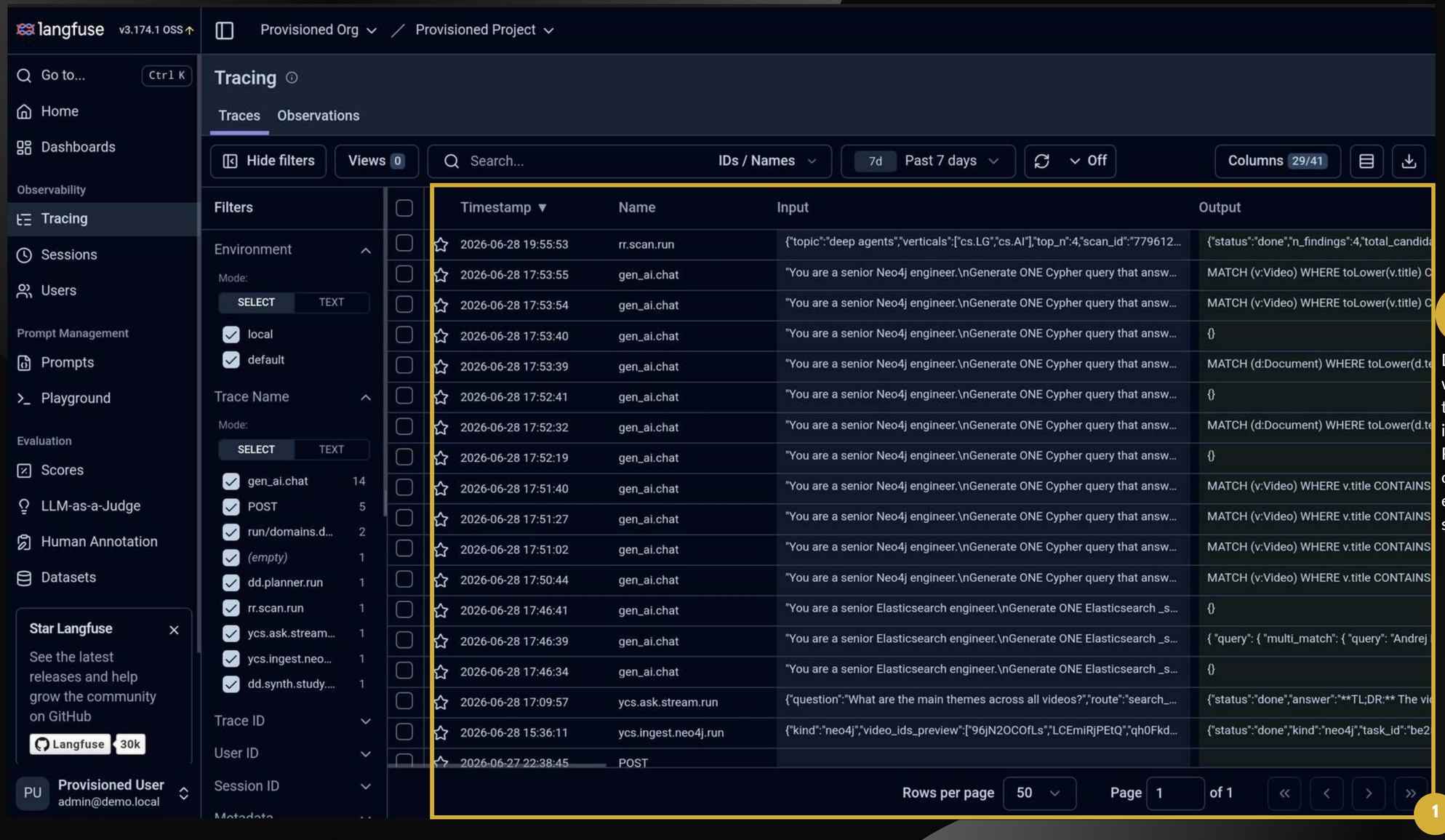
3 Every Generation Gets Graded, Automatically

Named graders — terminology consistency, citation coverage, code density — score every generation automatically. Synth output gets judged the same way every time, not spot-checked by hand.

COELHO NEXUS

LangFuse
http://localhost:23017
Username: **admin@demo.local**
Password: **admin-demo-password**

MICROSERVICES - LANGFUSE



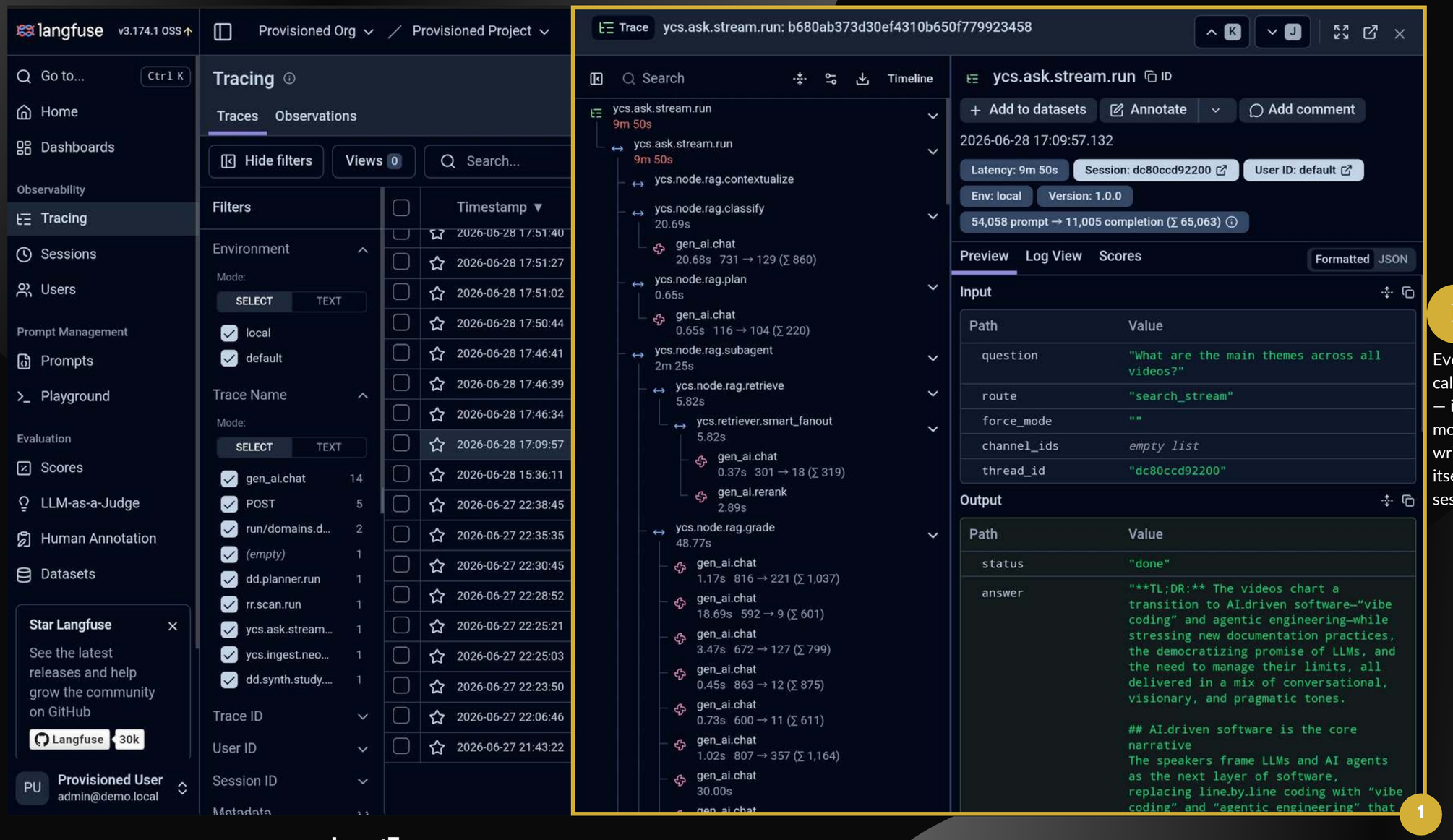
1 One Decorator, Every Domain's Traces

DD, YCS, and RR each wrap their pipeline nodes with the same @traced pattern — one decorator turns a function into a span that reports to both the infra observability stack and LangFuse at once. Research Radar nests phase-level spans as children of its parent scan. Nobody wired LangFuse into each domain separately; every feature reuses the same primitive.

LangFuse
http://localhost:23017
Username: **admin@demo.local**
Password: **admin-demo-password**

COELHO NEXUS

MICROSERVICES - LANGFUSE



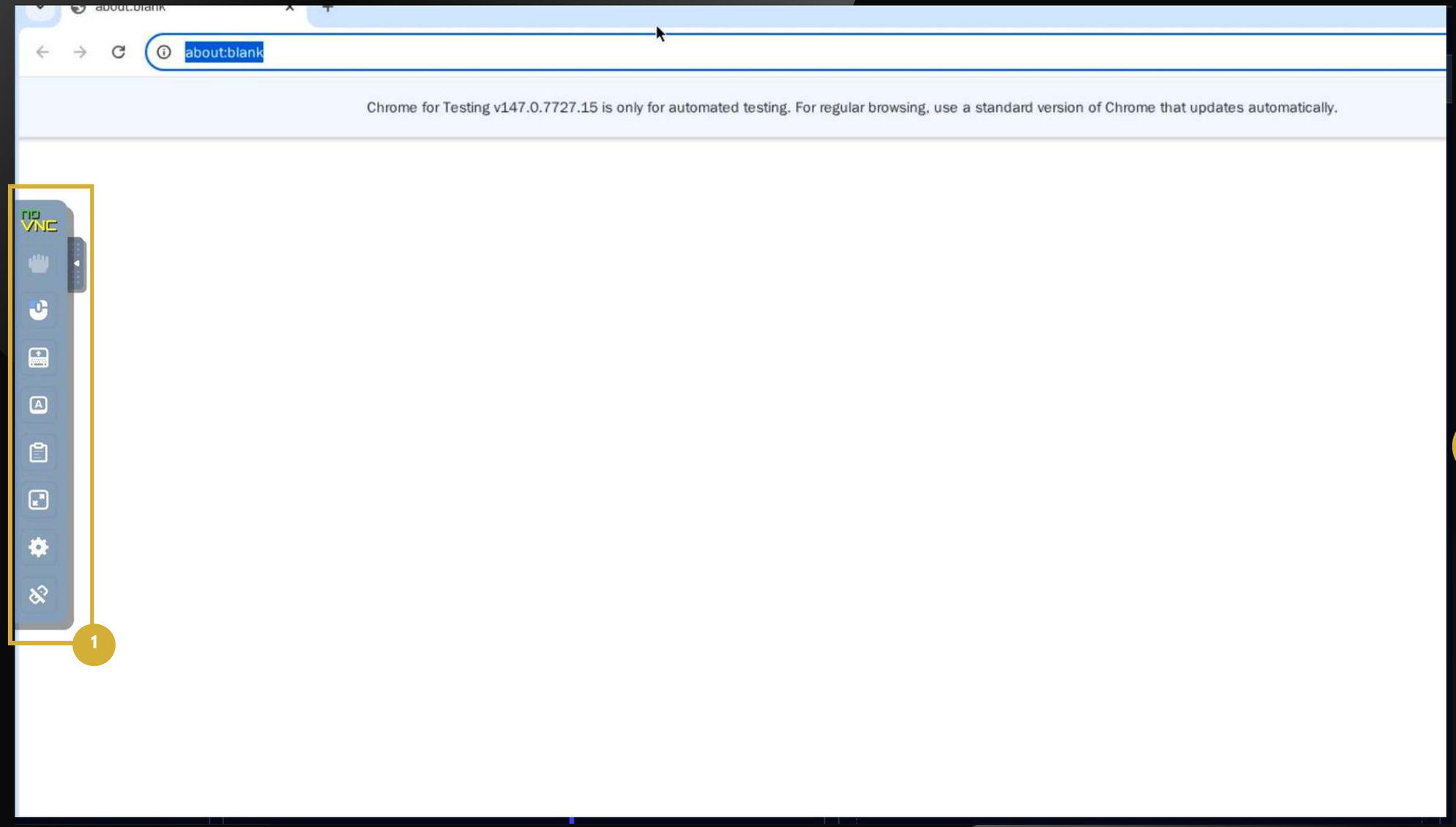
1 What Gets Sent to LangFuse, Automatically

Every span here — the parent trace, nested sub-calls, each model call's own token count and latency — is data the instrumentation transmitted the moment @traced wrapped a function. No node writes its own logging code; the payload assembles itself at every nesting depth and rolls up into one session automatically.

COELHO NEXUS

LangFuse
http://localhost:23017
Username: **admin@demo.local**
Password: **admin-demo-password**

MICROSERVICES - PLAYWRIGHT



Headed/Headless Modes, One Real Browser

Headed mode renders with a full display — this noVNC view is watching it live — because YCS needs a real, visible Chromium to get past YouTube's anti-bot detection during transcript scraping. Headless mode drops the display entirely for Crawl4AI's bulk framework-doc crawls, where speed matters more than visibility.

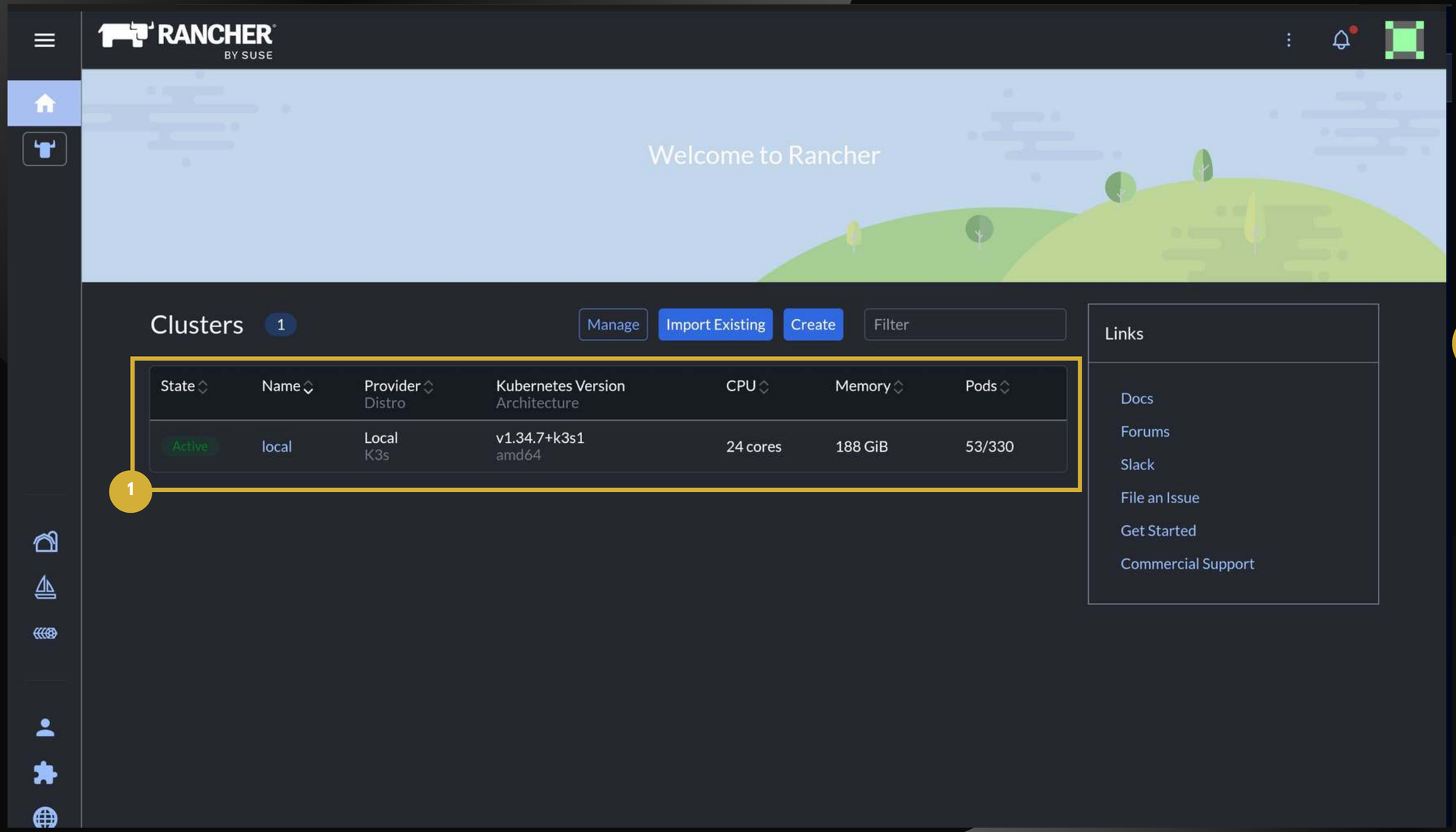
1 A Live Session You Can Actually Take Over

Clipboard sync, a virtual keyboard, connection settings — this sidebar turns noVNC from a passive screen-share into an actual remote desktop. If a scrape stalls on an unexpected popup or verification step, a human can click into this exact browser session and clear it by hand, live, instead of just watching it fail.

COELHO NEXUS

Playwright NoVNC
http://localhost:23018/vnc.html
Password: **vnc-demo-password**

MICROSERVICES - RANCHER



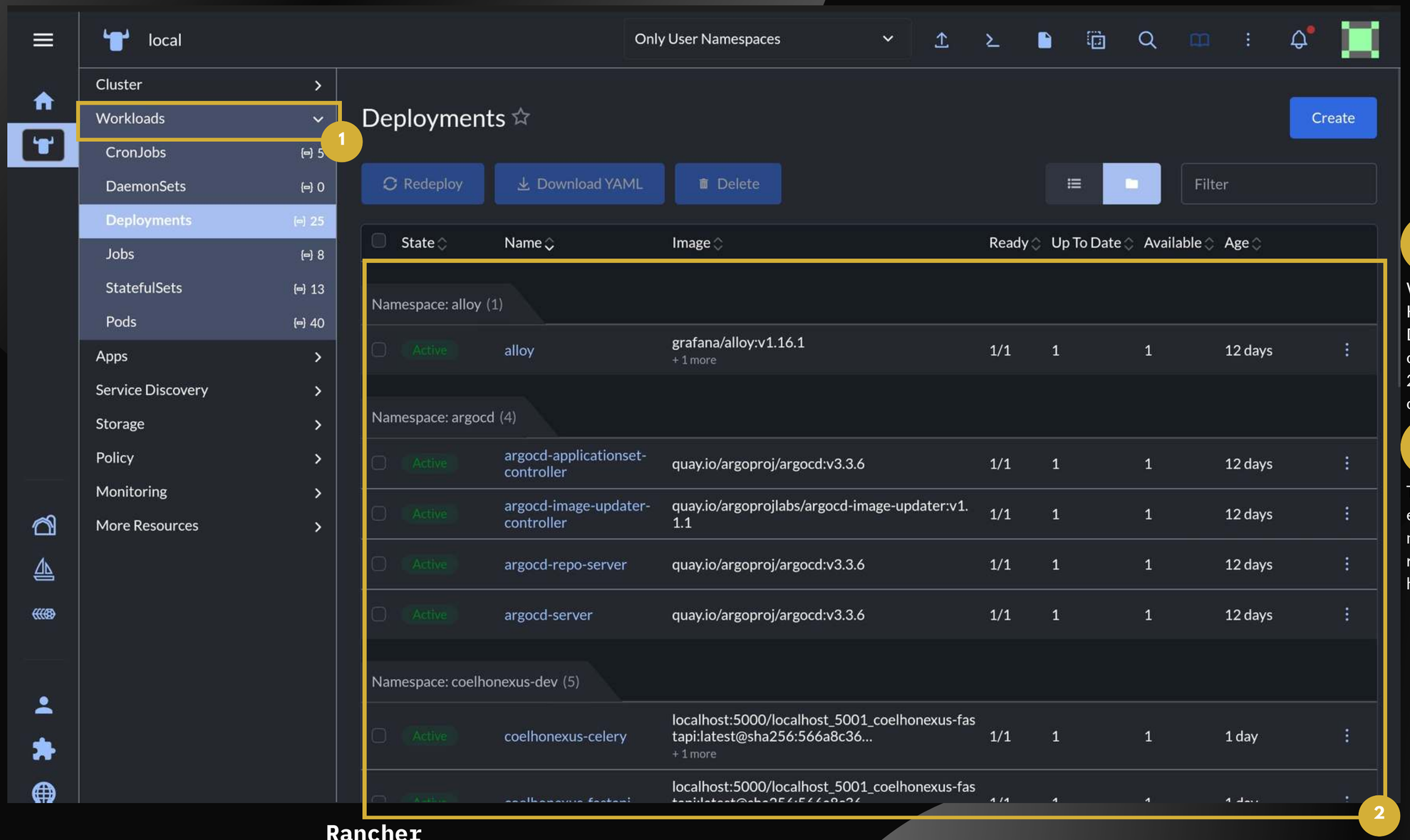
1 One Cluster, Every Service

All pods on a single node — Neo4j, Qdrant, Elasticsearch, MinIO, LangFuse, Playwright, Rancher itself, the app layer, the full observability stack — all running together at once. That density only works because every module was individually memory-tuned to fit, not left at chart defaults.

COELHO NEXUS

Rancher
https://localhost:23021
Username: **admin**
Password: **rancher-demo-bootstrap**

MICROSERVICES - RANCHER



1 One Click to Every Running Workload

Workloads groups every execution primitive Kubernetes has — CronJobs, DaemonSets, Deployments, StatefulSets, Jobs, raw Pods — into one menu. Selecting Deployments here surfaces all 25 running on the cluster, not just this project's own.

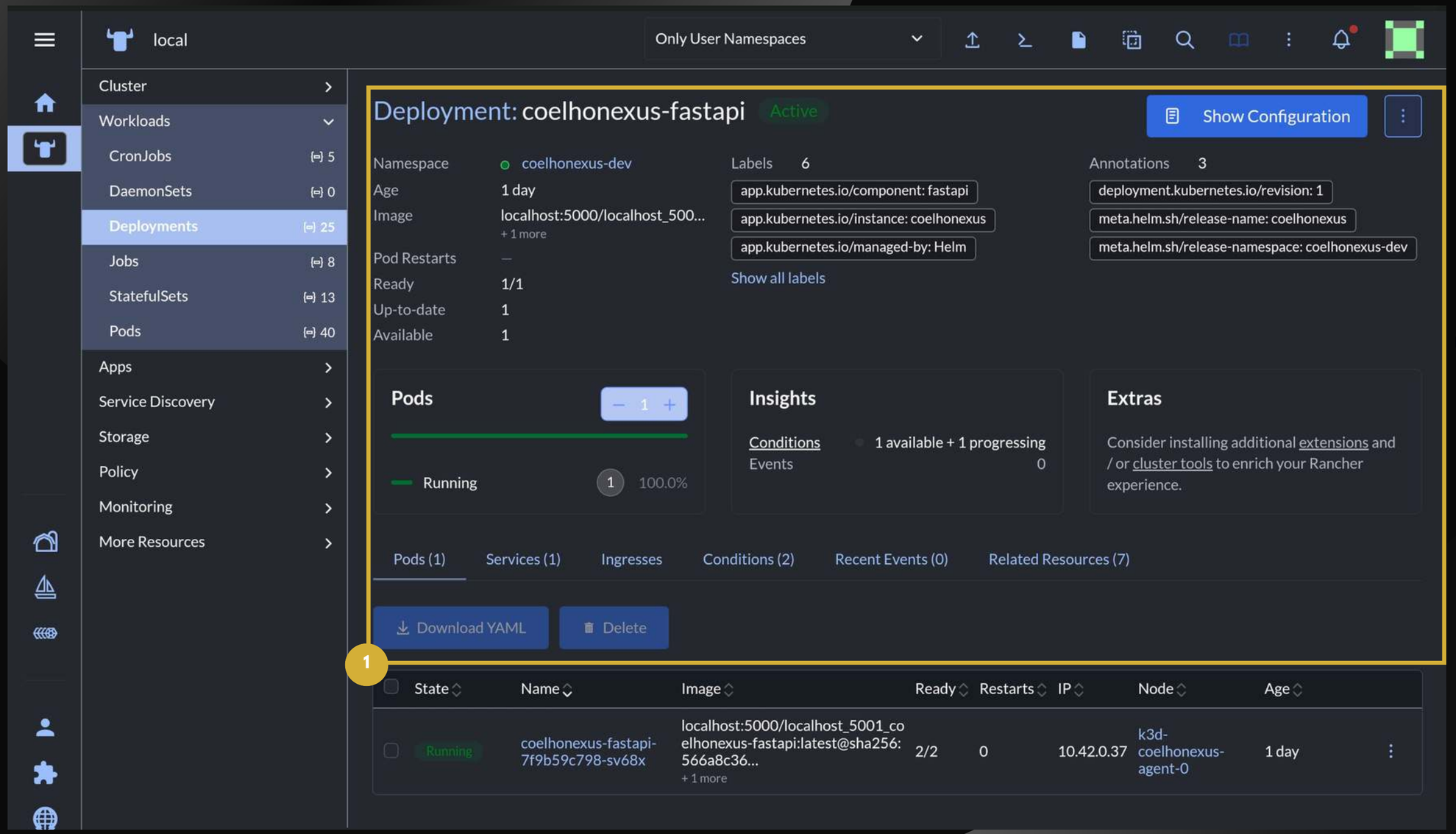
2 Every Microservice, One Screen

This is the screen that shows every namespace and every deployment on the cluster at once — every microservice this project depends on to actually run, all in one place, fully visible and usable from here.

COELHO NEXUS

Rancher
https://localhost:23021
Username: **admin**
Password: **rancher-demo-bootstrap**

MICROSERVICES - RANCHER



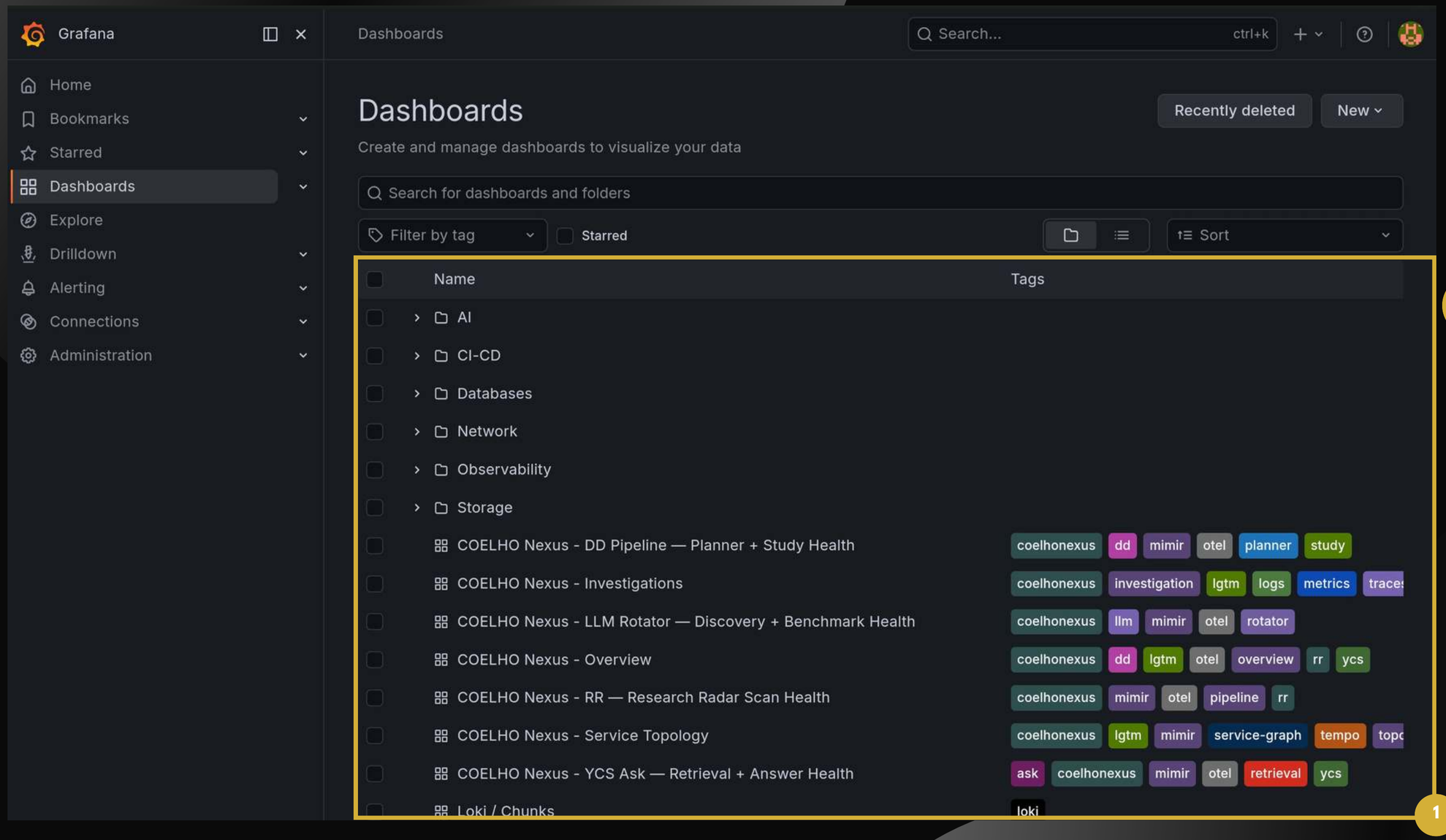
1 One Microservice, Every Detail

This is the screen that shows everything about a single deployment — its exact container image, every label and annotation, live pod health, and direct links to its Services, Ingresses, and Events. One microservice, fully inspectable, without leaving the page.

COELHO NEXUS

Rancher
https://localhost:23021
Username: **admin**
Password: **rancher-demo-bootstrap**

MICROSERVICES - GRAFANA



1

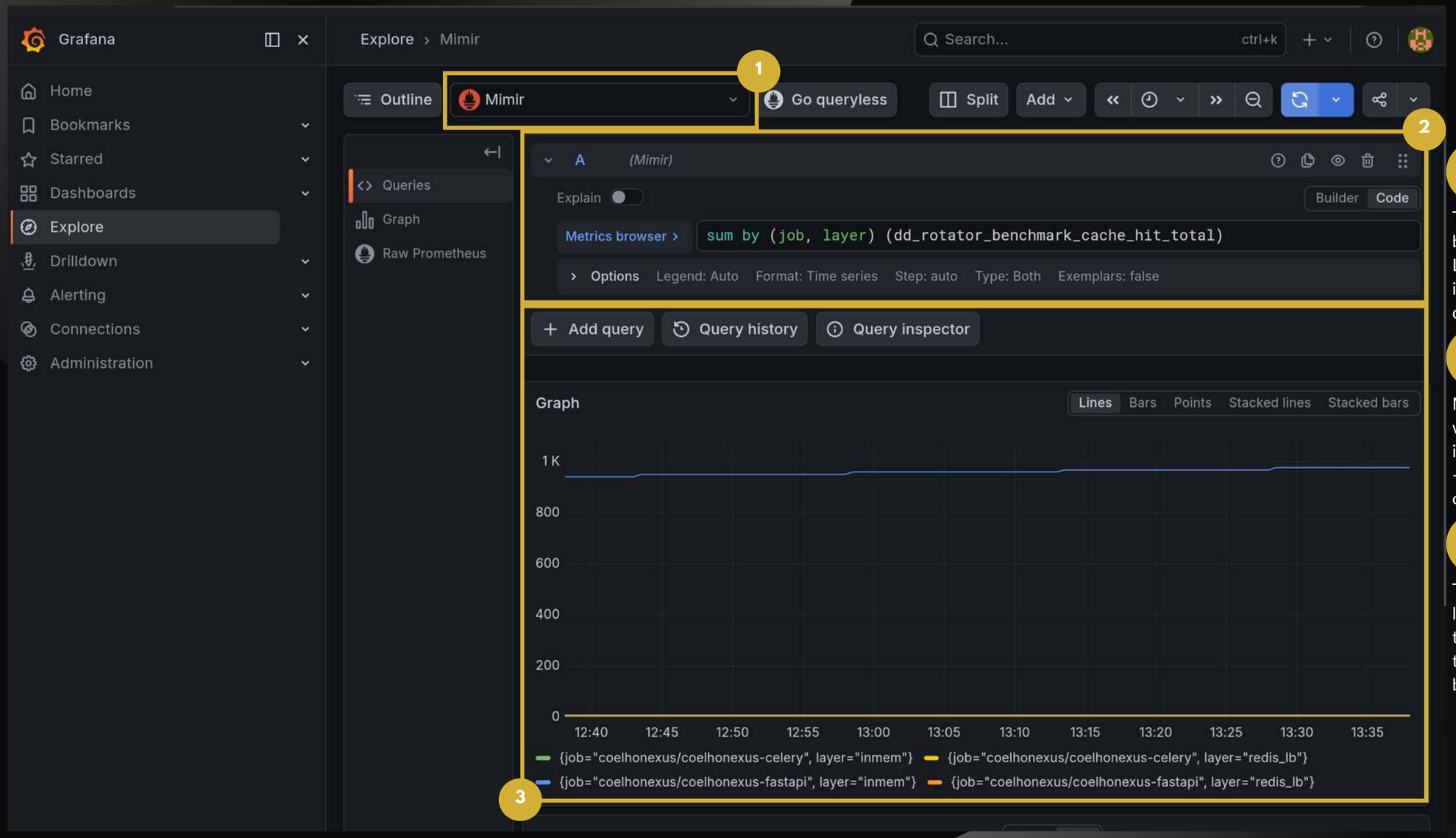
1 Every Signal, Cross-Linked

One dashboard per feature, plus a cross-signal Investigations view — but the real design choice is that they're wired together, not just co-located: a slow metric has an exemplar link straight to the exact trace that caused it, and every trace links onward to its own logs. These dashboards ship inside the application's own Helm chart, not a separate infra concern, so they version alongside the features they actually monitor.

COELHO NEXUS

Grafana
http://localhost:23022
Username: admin
Password: **admin**

MICROSERVICES - GRAFANA



1 One Explorer, Three Backends

This dropdown is the only thing that changes between querying metrics, logs, or traces — Mimir, Loki, and Tempo all live behind the same Explore interface, same shortcuts, same UI, just a different datasource selected.

2 PromQL You Write, With Help Finding What Exists

Metrics Browser helps you discover what's available without memorizing exact names, but the query itself is hand-written PromQL — sum by (job, layer) — full query-language control, not a simplified GUI-only builder.

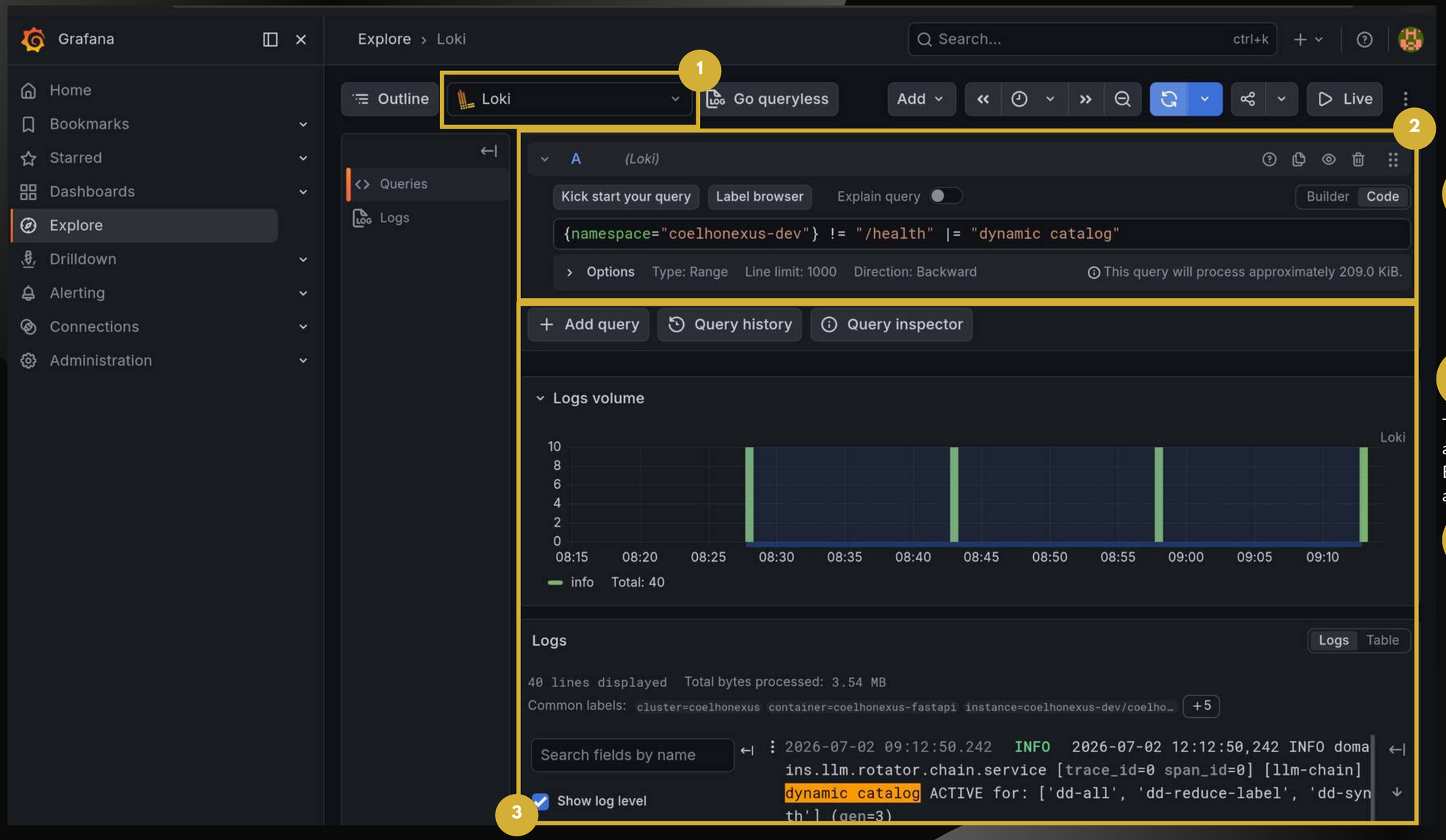
3 Where the Query Actually Resolves

This panel renders whatever query sits above it — line graph, legend, time axis — built automatically the moment the query runs. Change the query and the whole panel redraws instantly, no dashboard to build first, no separate step to visualize the result.

COELHO NEXUS

Grafana
http://localhost:23022
Username: admin
Password: **admin**

MICROSERVICES - GRAFANA



1 A Different Signal, A Different Language

Metrics aggregate into numbers; logs stay raw text. Loki's query reflects that — label-matching to pick a stream, then a text filter, not a math expression. Same Explore shell as Mimir, a completely different way of asking a question.

2 Where You Write and Run the Query

This is the query editor — type LogQL here, run it, and the results below update immediately. Everything else in this box is just support tooling around that one core action.

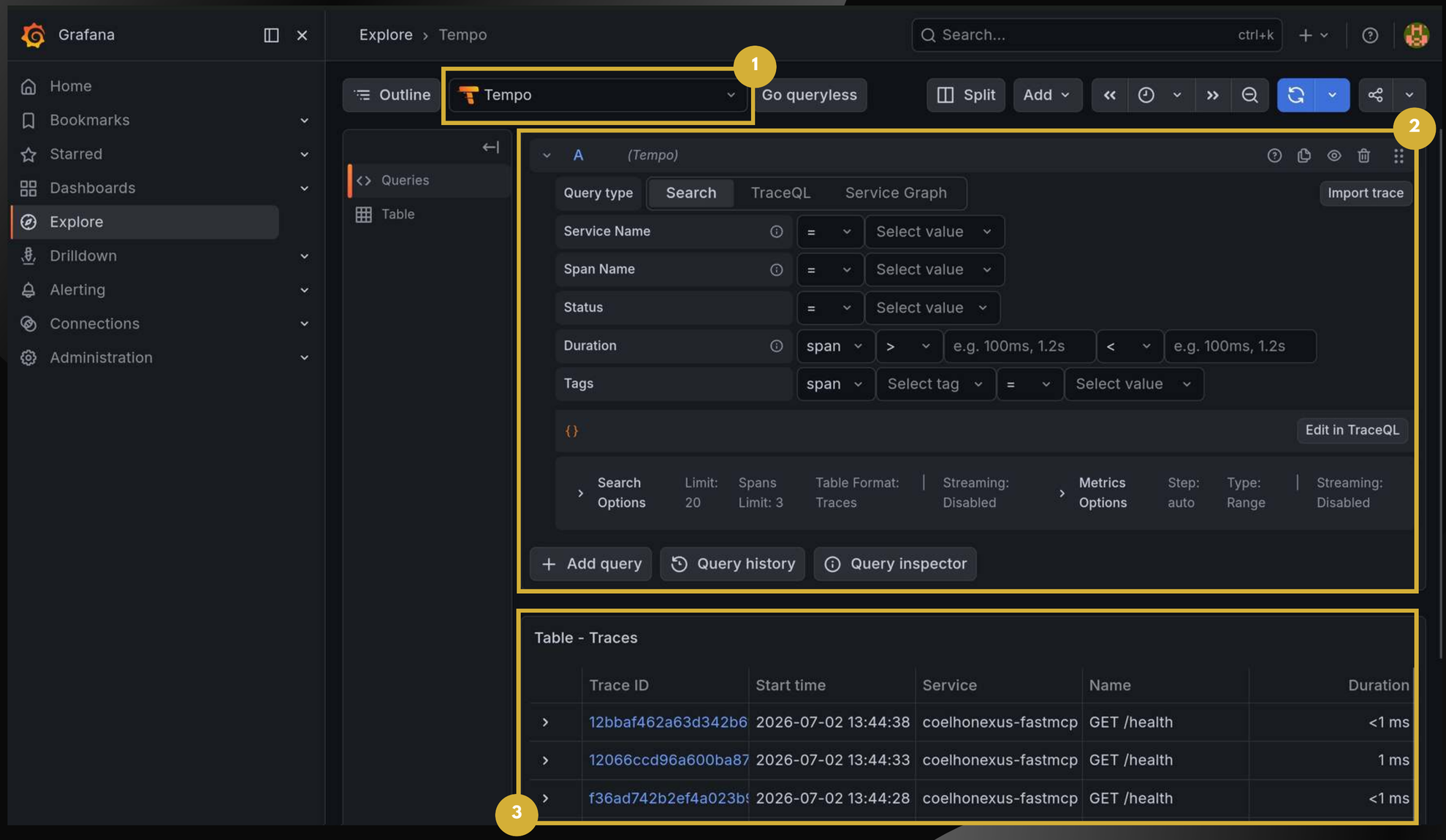
3 Where the Results Show Up

This is where matching log lines appear — a volume histogram on top, the actual text underneath. Whatever the query above finds, it renders here, nothing to configure separately.

COELHO NEXUS

Grafana
http://localhost:23022
Username: admin
Password: **admin**

MICROSERVICES - GRAFANA



1 A Third Backend, Same Shell

Traces aren't numbers or text lines — they're a hierarchy of spans, parent to child. Tempo's picker sits in the same Explore shell as Mimir and Loki, but what it queries is shaped completely differently.

2 Where You Build and Run the Search

Same core action as the other two — configure a query, run it, see results below — but Tempo defaults to a filter form instead of free-text code. "Edit in TraceQL" is right there if you want to drop into the query language directly.

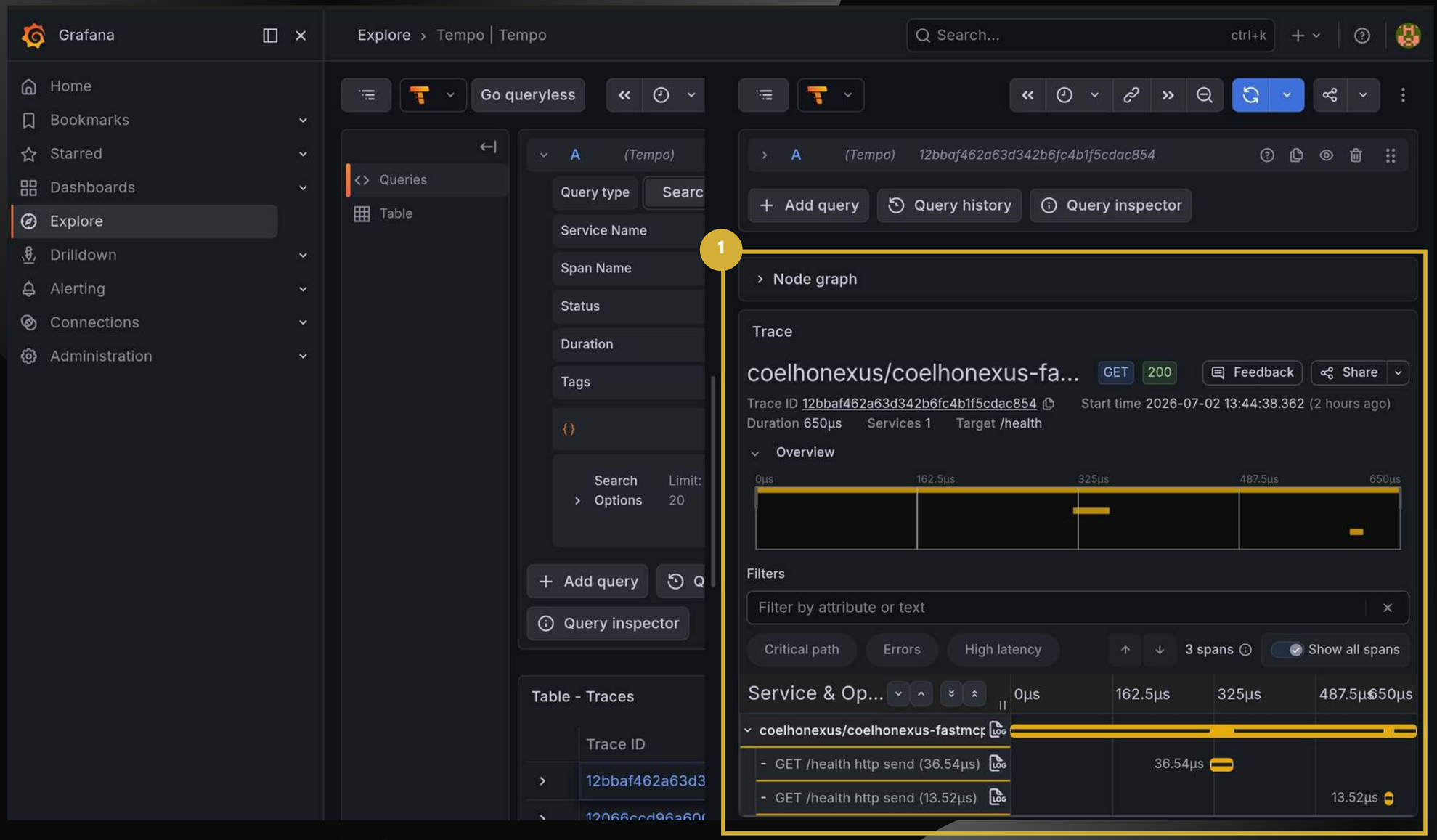
3 Where the Results Show Up

This is where matching traces appear — one row per trace, service, name, and duration at a glance. Click any row to expand its full span waterfall.

COELHO NEXUS

Grafana
http://localhost:23022
Username: admin
Password: **admin**

MICROSERVICES - GRAFANA



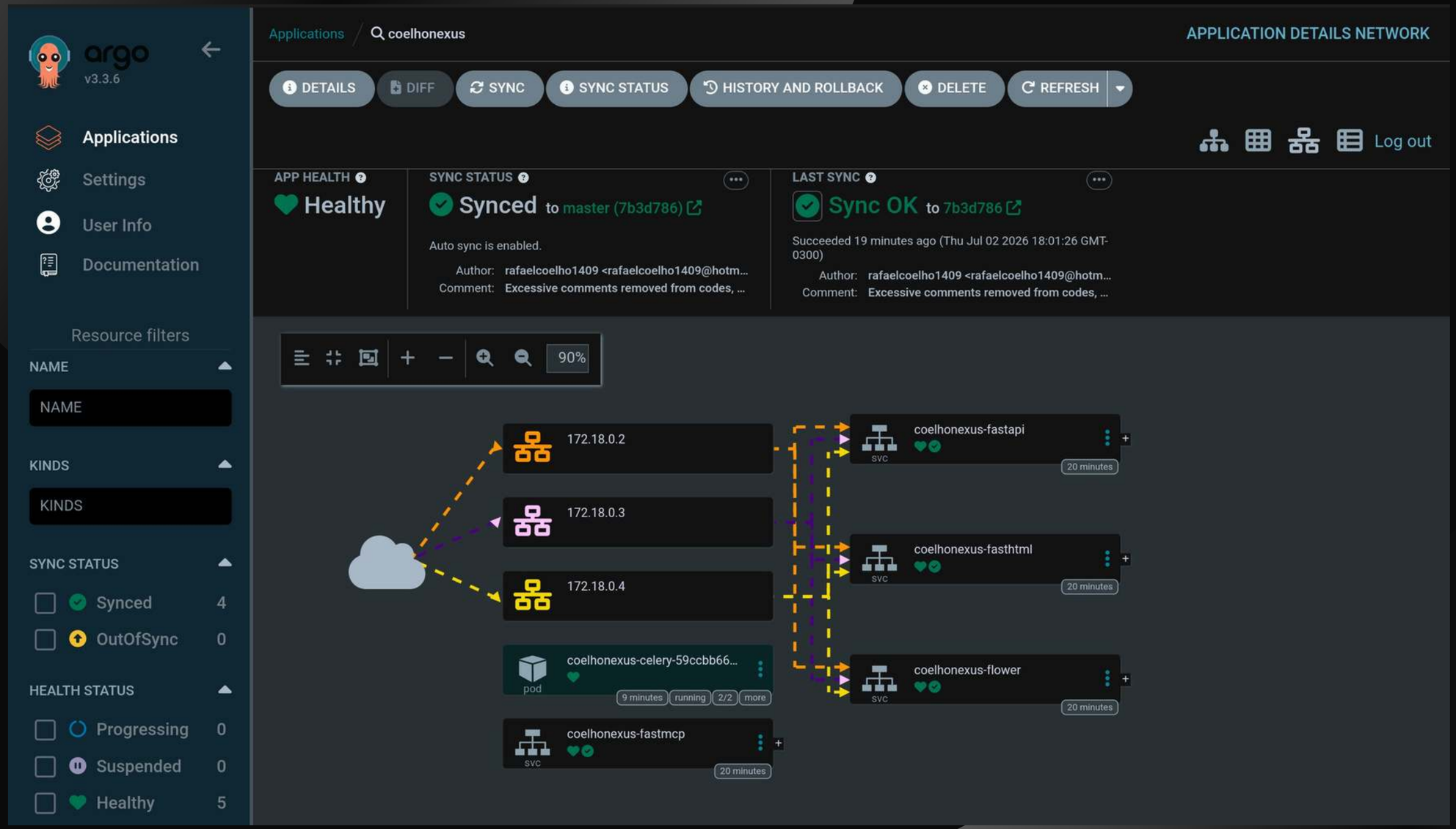
1 One Trace, Every Span, Expanded

Clicking a row from the results table opens this — the parent request and every child span nested underneath it, each with its own exact duration. Same waterfall view for any trace in the system, a health check or a full pipeline run alike.

COELHO NEXUS

Grafana
http://localhost:23022
Username: admin
Password: **admin**

MICROSERVICES - ARGOCD



The Graph Proves What the Code Says

Three services trace back through Docker-network IPs to the cloud icon — those are the ones with a LoadBalancer path out. FastMCP sits apart with no such line, because its Service is deliberately ClusterIP-only: an internal gateway other pods call, never meant to be reached directly. The topology isn't decorative — it's the same internal/external split the Service manifests define, just drawn instead of read.

COELHO NEXUS

ArgoCD
http://localhost:23023
Username: **admin**
Password: **admin**

LET'S WORK TOGETHER

https://rafaelcoelho.pages.dev/