IntuitionLabs

INTUITIONLABS / RESEARCH & PRACTICAL GUIDANCE

How to Set Up RAG for Internal Research Repositories

> Build practical AI for pharma and biotech with IntuitionLabs. We help life-science teams turn complex information and workflows into useful software, governed knowledge systems and AI tools.

Published by IntuitionLabs · 2026-03-14 · rag architecture · electronic lab notebooks · lims · vector search · large language models · enterprise ai

Original article: https://intuitionlabs.ai/articles/setup-rag-internal-research-repositories



In this article

1. Executive Summary
2. Introduction and Background
3. The RAG Framework and Architecture
4. Internal Research Repositories and Data Sources
5. Building a RAG Pipeline over Internal Data
6. Data Ingestion and Preprocessing
7. Embedding and Indexing
8. Retrieval: Search and Filtering
9. Augmentation and Generation
10. Tools, Frameworks, and Architectural Choices
11. Case Studies and Real-World Examples
12. Egnyte's Document Q&A and Knowledge Bases
13. Pharmaceutical R&D: ELN/LIMS RAG
14. Enterprise Knowledge and Other Use Cases
15. Data Analysis and Evidence of Impact
16. Challenges, Risks, and Limitations
17. Implications and Future Directions
18. Conclusion

Executive Summary

Retrieval-Augmented Generation (RAG) is a cutting-edge AI approach that integrates a large language model (LLM) with an external knowledge retrieval system, allowing the AI to draw on a company's private data when answering questions. In practice, RAG enables models to "ground" their responses in actual records (e.g. lab notebooks, databases, policy documents), drastically improving accuracy and trustworthiness (time.com) (www.techradar.com). Modern RAG setups have been adopted by major tech firms (Microsoft, Google, Amazon, Nvidia) to empower domain-specific AI assistants (time.com). In research-intensive industries like pharmaceuticals, RAG promises to unite siloed data in Electronic Laboratory Notebooks (ELNs), Laboratory Information Management Systems (LIMS), and enterprise file shares (e.g. Egnyte) under a single question-answer interface (intuitionlabs.ai) (www.techradar.com).

In this report, we provide an in-depth technical roadmap for building RAG over internal repositories. We first explain RAG's architecture and advantages: it retrieves relevant text snippets (via vector/BM25 search) and appends them to the query so the LLM generates answers grounded in evidence (time.com). We then survey the typical data sources in a research organization. ELNs contain unstructured experimental notes (text, images, tables, protocols), LIMS hold structured assay and inventory records, and platforms like Egnyte serve as centralized file repositories. Each source requires tailored preprocessing (e.g. text extraction, table parsing, ontology mapping) before indexing in the RAG system (www.egnyte.com).

Our report examines the end-to-end RAG pipeline: content extraction (using PDF/OCR libraries, databases queries), intelligent chunking of long documents, embedding generation (via OpenAI/Cohere/HuggingFace models), and indexing in fast search engines (FAISS, Elasticsearch-HNSW, etc.). Retrieval combines semantic vectors and keyword search, often in a hybrid scheme to boost recall (www.egnyte.com) (www.techradar.com). The top-K retrieved passages are then reranked (via a cross-encoder) before being fed to the LLM as contextual prompt. Finally, the LLM generates an answer, which can include citations back to the original sources for transparency. We detail best practices at each stage, drawing on published case studies (e.g. Egnyte's implementation) and experimental results (www.egnyte.com).

We also present quantitative data demonstrating RAG's impact. For instance, Sawarkar et al.'s Blended RAG study reports improved retrieval results on Natural Questions and TREC-COVID (authors' repository). Evaluate retrieval quality, response accuracy, and citation support in a pilot before drawing conclusions about productivity (Microsoft guidance). Treat reduced research overhead as a hypothesis to measure against the existing workflow. However, implementing RAG is non-trivial: special engineering is needed to handle tables and structured data (a common format in LIMS) (www.egnyte.com), and strong governance is required to protect sensitive information (petronellatech.com) (thenewstack.io).

In conclusion, RAG over internal repositories promises a "superhuman search" capability (www.techradar.com): scientists can pose natural-language queries (e.g. "What were the assay results for compound X?") and get answers synthesized from their own lab's data, with sources cited. This approach can greatly accelerate discovery and improve decision-making by breaking data silos (intuitionlabs.ai) (www.techradar.com). Our report provides a comprehensive guide — from architecture to tools to pitfalls — for research organizations aiming to harness RAG on ELNs, LIMS, Egnyte, and similar systems.

Introduction and Background

Scientific research is increasingly data-driven. Modern laboratories routinely generate terabytes of information: experimental procedures, spectrometry outputs, assay results, inventory lists, safety protocols, and more. Traditional data management tools (ELNs and LIMS) help record this work, but each operates in relative isolation. For example, an ELN might contain a chemist's multi-step synthesis notes (in narrative or table form), while a LIMS stores the same samples' purity analyses and storage conditions in a relational database. Meanwhile, general files such as SOPs, patents, meeting minutes or external literature may reside in network drives or content management platforms (e.g. Egnyte). Researchers often find valuable information trapped in these silos: answering a query like "What were the last dissolution-test results for tablet batch 1234?" may require cross-referencing multiple systems manually. This fragmentation slows progress and risks overlooking critical insights.

Large language models (LLMs) offer a tantalizing solution to unify knowledge. They can parse and synthesize text, potentially serving as personal AI assistants for researchers. However, LLMs have two key limitations in this domain. First, they are trained on general web corpora and have knowledge cutoffs, so they do not "know" a laboratory's proprietary data by default (intuitionlabs.ai). Second, LLMs tend to hallucinate—presenting plausible but false statements when the answer isn't in their training set. In high-stakes settings like drug R&D or compliance, hallucinations (e.g. citing a nonexistent protocol) can have serious consequences.Retrieval-Augmented Generation (RAG) emerged to address exactly these shortcomings (time.com) (intuitionlabs.ai). In

RAG, when a user asks a question, the system first retrieves documents or data shards relevant to the query from an external knowledge base (“non-parametric memory”), then feeds those snippets into the LLM alongside the query. The LLM is intended to answer from retrieved evidence, but it can still produce inaccurate responses. Require source citations and abstention when evidence is insufficient, and evaluate answer groundedness independently of retrieval success (Microsoft guidance). This hybrid approach combines the LLM’s language fluency with real, up-to-date content. As Time magazine notes, RAG “allows AI models to answer queries by drawing on external texts, be it company documents or a news website. It can also reduce hallucinations and even give models access to up-to-the-minute information” (time.com). In other words, RAG effectively grounds generative AI in evidence.

The concept of RAG was formalized by Lewis et al. in 2020 (studylib.net), and since then it has gained rapid traction. Industry leaders like Microsoft, Google, Amazon, and OpenAI now embed RAG architectures in their enterprise AI offerings. Microsoft’s AI Copilot for Office, for instance, uses a RAG-like approach to answer based on a user’s own documents. Early product case studies report users finding information much faster: Egnyte’s rollout of RAG-powered query features “allowed users to quickly find and synthesize information from vast document repositories with accuracy and context” (www.egnyte.com).

In the scientific realm, RAG holds promise for laboratories to build “superhuman” knowledge assistants. Rather than feeding LLMs fixed training data, RAG can access the latest experiment results and internal literature on demand (time.com) (www.techradar.com). For example, a researcher could ask a RAG system, “Has compound X been tested in any gene expression assay for kinase Y?” The system would retrieve relevant ELN entries (experiment notes), LIMS assay records, and perhaps a PDF of published data, and then generate an answer citing the specific data sources. BioSymetrics reports experimenting with RAG in its products and services (BioSymetrics).

This report explores **how to set up an enterprise RAG system over internal research repositories like ELNs, LIMS, and Egnyte**. We cover the historical context, technological components, practical case studies, and future outlook. After outlining RAG’s core framework, we analyze each data source’s characteristics and how to integrate it. We delve into the retrieval pipeline – from document parsing and embedding to vector search and prompt engineering – with extensive references. We examine evidence (benchmarks, expert quotes, and industry reports) to quantify RAG’s benefits and limitations. Finally, we discuss data governance, security, and emerging trends (e.g. knowledge graphs, multi-modal content) that shape RAG’s future in research settings.

The RAG Framework and Architecture

RAG (Retrieval-Augmented Generation) is an AI architecture that **combines pre-trained generative models with an external retrieval mechanism** (time.com). Intuitively, it “marries information retrieval with text generation,” enabling an LLM to be updated in near real time with fresh or proprietary content. The standard RAG workflow involves three stages:

- **Retrieval:** Given a user’s query (in natural language), the system searches its indexed knowledge base (which may include internal documents, databases, etc.) to find the most relevant text snippets or records (www.egnyte.com) (intuitionlabs.ai). This often uses semantic vector search (embeddings +

approximate nearest neighbor) optionally augmented by keyword/BM25 search (www.egnyte.com) (www.techradar.com).

- **Augmentation:** The retrieved documents (or selected excerpts) are concatenated with the original query to form an augmented prompt (www.egnyte.com) (intuitionlabs.ai). This gives the LLM the contextual information it needs to answer accurately. For example, if the query is "What is the melting point of compound Z?", the context might include lab notes stating "compound Z melts at 150°C under the procedure..." drawn from an ELN entry.
- **Generation:** The LLM processes the augmented prompt and generates a response, typically conditioned to use only the given context. Because the answer is derived from real data, the risk of hallucination (inventing facts not in the sources) is greatly reduced (time.com) (www.techradar.com). The LLM output can also be post-processed to extract citations or confirm facts.

This approach contrasts with a standalone LLM: by coupling with retrieval, RAG effectively creates a system that acts like "search + chat." As one reviewer put it, RAG "shrinks the gap" between AI output and reality by tying answers to verifiable sources (intuitionlabs.ai). A helpful analogy is that RAG turns an LLM into a "superhuman search engine" (www.techradar.com) — it not only finds relevant documents but synthesizes them into fluent answers.

Advantages of RAG

- **Accuracy & Freshness:** Because RAG answers are explicitly based on retrieved data, their factual correctness improves. Contemporary references note that grounding the model with real data *dramatically reduces hallucinations* (time.com) (www.techradar.com). For businesses, this means the AI can cite actual policy or research documents. RAG also bypasses the model's knowledge cutoff: if a new lab result or a latest publication exists in the index, RAG can incorporate it immediately (time.com) (thenewstack.io).
- **Domain Adaptability:** New data sets (proprietary or updated) can be added to the RAG corpus without retraining the LLM. Unlike finetuning large models with each data update (which is costly and static), RAG simply adds the new content to the vector database. This also means organizations can maintain multiple domain-specific knowledge bases easily.
- **Transparency:** Many RAG implementations are designed to "cite their sources." After generating an answer, the system can list which document excerpts it used. This is not typical of standalone LLMs, but it gives end-users (e.g. scientists) the ability to verify the AI's claims (time.com) (www.techradar.com). For example, Egnyte's RAG Q&A attaches references back to the original file sections, allowing users to drill down if needed.

Historical Context and Core Research

The seminal RAG architecture was introduced by Lewis et al. (2020) (studylib.net). Their work fine-tuned seq2seq transformers by supplying them with retrieved passages from Wikipedia, via a neural retriever. They demonstrated two variations: one where the same retrieved passages were used for all generated tokens, another where different passages could influence different parts of the answer. RAG set new state-of-the-art on the Natural Questions QA dataset (studylib.net), proving the concept that combining parametric (LLM) and non-parametric (retrieval) memory yields more knowledgeable generation.

Since then, RAG has evolved in both academia and industry. Extensions include agentic RAG (RAG with planning and retrieval loops), multi-hop RAG (chaining multiple retrieval rounds), and multi-modal RAG (incorporating images or graphs) (intuitionlabs.ai) (www.techradar.com). Frameworks like LangChain or GPT Index now provide out-of-the-box utilities for RAG pipelines (doc loaders, vector DB interfaces, prompt templating). Nevertheless, implementing a robust RAG system in an enterprise requires careful engineering: data ingestion at scale, index management, latency optimization, and privacy controls must all be managed (www.egnyte.com) (petronellatech.com).

In the next sections, we discuss how RAG's principles are applied to the specific case of internal research data, the architecture choices involved, and the practical evidence of RAG's impact.

Internal Research Repositories and Data Sources

Research organizations typically maintain **several classes of internal data systems** that together form a sprawling knowledge base. Understanding their characteristics is crucial for designing a RAG pipeline.

- **Electronic Lab Notebooks (ELNs):** ELNs are digital replacements for paper lab notebooks. Researchers record experimental procedures, observations, and results in ELNs, often in a semi-structured or free-form manner. An ELN entry might mix text narrative ("Day 1: Dissolved compound Z in solvent A..." (www.egnyte.com)) with tables of quantitative results (e.g. yield percentages, instrument readouts), figures (spectra, photos), and metadata (timestamps, author, experiment ID). Unlike standard documents, ELN content can be highly heterogeneous. For example, one ELN template might log multi-step synthetic procedures step-by-step, while another logs genomic assay parameters in a spreadsheet. As IntuitionLabs notes, ELNs capture "synthetic chemistry steps, NMR data, purity analyses" or cell-culture results, but this rich data becomes useful only if it can be queried (intuitionlabs.ai).
- **Laboratory Information Management Systems (LIMS):** LIMS are highly structured databases designed to manage lab operations. They typically track samples, reagents, instrument outputs, and workflows. A LIMS record might include a sample ID, its source, process steps, QC results, and its current storage location. Because they are mostly database-driven, LIMS data is very structured (rows and columns), unlike the free text of ELNs. For example, a LIMS might store " [Sample 12345] – Date measured, Method, Result Value, Technician" in fields. These systems ensure compliance and traceability. However, querying a LIMS often requires custom queries or GUIs specific to that system.
- **Enterprise File Repositories (e.g. Egnyte):** Many organizations use file-sharing platforms (Egnyte, SharePoint, Google Drive) for general document storage. These repositories can contain diverse content: SOP PDFs, Word documents, spreadsheets, unpublished reports, correspondence, etc. Egnyte, for instance, may hold thousands of documents across departments. Critically, these repos often contain both relevant data (e.g. a PDF of a lab meeting's slide deck with test plans) and irrelevant content. Egnyte's own documentation highlights that without AI, one would have to search file names or folder structures; RAG aims to search within the content of these files (www.egnyte.com).

- **Institutional Knowledge Bases:** Beyond technical data, organizations have "institutional knowledge": corporate wikis, past project archives, publication libraries, and knowledge graphs. This includes anything from retired investigators' notes to an internal database of all researched drug targets. IntuitionLabs emphasizes that RAG must eventually unify ELN, LIMS *and* this broader knowledge (patents, SOPs, meeting minutes) under one framework. In advanced implementations, knowledge graphs might be built to connect entities (genes, compounds, diseases) across systems.

Table 1 below summarizes these source types and how one might approach integrating them into a RAG system:

Repository Type	Data Example	RAG Integration Strategy	Example Tools/Notes
ELN (Lab Notebook)	Narrative & tables of experiments (text, images)	Extract text from entries (including OCR for images), chunk long records into digestible pieces, preserve table structures, embed each chunk.	Unstructured parsing (PDF/OCR), LangChain TextSplitter, LLM-based table parsers.
LIMS (Lab DB)	Structured assay records, sample metadata	Export database views (CSV/JSON), convert records or flowsheets into text or embeddings, or build a knowledge graph; index numeric results separately.	SQL queries, Pandas for preprocessing, graph DB (Neo4j), specialized LLM plugins.
Egnyte (File Share)	Documents, spreadsheets, images, PDFs	Use document parsers to extract text/tables, segment multi-format files (e.g. convert to markdown), generate embeddings for each chunk, combine with keyword indices.	PyMuPDF/Pandoc/unstructured, Elasticsearch (BM25 + HNSW), Egnyte API.
Knowledge Graph	Ontologies of compounds, targets, assays	Query graph DB for relevant facts (nodes/edges) as part of retrieval; inject graph-derived text or embeddings into RAG.	RDF/SPARQL engines, graph embedding models (TransE, ComplEx), Mindwalk AI frameworks.

Table 1: Common internal data sources (ELN, LIMS, Egnyte) and how they can be prepared for RAG. Each requires different preprocessing (e.g. ELN text extraction vs. LIMS SQL export) (intuitionlabs.ai) (www.egnyte.com).

As Table 1 indicates, each data source poses unique challenges. For ELNs, the unstructured nature and mixed media (images, free text, tables) demand flexible parsers. For example, Egnyte engineers found that converting tabular data into plain text lost important relationships, leading them to implement specialized table extraction that preserves row/column context (www.egnyte.com). LIMS, being structured, might seem easier, but bridging them to RAG often requires turning each record into a textual synopsis or translating them into a knowledge graph. Several industry reports suggest combining LLMs with knowledge graphs can capture relational facts that static text cannot (intuitionlabs.ai). Egnyte-style file repos typically require a hybrid of semantic and keyword search: one might embed the file contents for semantic similarity, while also indexing metadata (file names, tags) for keyword queries (www.egnyte.com) (www.techradar.com).

In practice, many RAG deployments create a unified **knowledge index** that ingests content from all these sources. Documents and database extracts are broken into "chunks" (e.g. a paragraph, a table row) and stored with an embedding vector. RAG users can then query this corpus without knowing where each piece

originally lived. We explore an example of such an implementation in the section on Egnyte's system below.

Building a RAG Pipeline over Internal Data

Constructing a RAG system involves several sequential components, which we outline here. At a high level, the pipeline consists of: (1) data ingestion and preprocessing, (2) embedding and indexing, (3) query-time retrieval, and (4) answer generation. Each step must be adapted to the scientific data context.

Data Ingestion and Preprocessing

The first step is to **extract content** from your repositories. This typically means:

- **Document Parsing:** For files (PDFs, Word docs, lab notebook exports, spreadsheets), use document parsers to extract text. Tools like unstructured, PyMuPDF, Apache Tika, or OCR libraries (Tesseract, Google Vision) are commonly used. Egnyte's RAG pipeline, for instance, employs PDF parsers and OCR to get plain text from uploaded files (www.egnyte.com). Audio/video recordings might be transcribed via speech-to-text if relevant. The goal is to turn each source into one or more text blocks.
- **Table Extraction:** Tables and charts are often critical in lab data. Simple text extraction will flatten tables, losing structure (www.egnyte.com). Egnyte discovered that queries requiring tabular reasoning ("How many units of reagent X were used each day?") demand preserving the table format. Solutions include converting tables to Markdown or JSON representations for the LLM (as Egnyte ultimately did) (www.egnyte.com). Open-source libraries (Camelot, Tabula, pandas' read_excel) can parse spreadsheet/pivot tables, while HTML or Markdown conversion retains headers.
- **Data Transformation:** For LIMS or databases, content may need to be transformed. Options include: exporting records to CSV/JSON and treating each row as a "document", generating narrative descriptions of each entity, or constructing a mini-ontology. For example, one could write: "Sample 12345 (Solution A, pH 7) – measured conductivity 2.3 mS at 10:00 AM" as a text chunk. In some cases, building a knowledge graph (KG) is useful, where entities (compounds, assays) and relations (tested in, yields, categories) are explicitly modeled (intuitionlabs.ai). The KG can either be queried directly or linearized into text.
- **Cleaning and Chunking:** Once raw text is obtained, it should be cleaned (remove duplicates, fix OCR errors) and split into manageable chunks. Common practice is to break documents on paragraph boundaries or every ~500–1000 characters, though this can be adjusted. Egnyte used a **RecursiveCharacterTextSplitter** to segment retrieved text into ~1000-character chunks (www.egnyte.com). Proper chunk size balances context (too small loses coherence) against token budget limits (too large wastes context window). Tables may be kept intact as one chunk if concise, or sectioned logically (e.g., one row per chunk with column headings since Egnyte ultimately did).

During ingestion, it's also wise to **collect metadata** for each chunk: source document ID, file path, author, date, data type (title, table caption, etc.) (www.egnyte.com). This metadata enables faceted search and permission filtering later. For instance, Egnyte adds a "Type" field so it can identify whether a chunk is text, a

table, an image caption, etc. (www.egnyte.com).

Key Point: Robust ingestion is crucial. Mistakes (e.g. mis-OCR of a number) will propagate to answers. It is often worth manual spot-checking and iterative refinement of the parsing pipeline. Egnyte's experience—evaluating multiple libraries and formats—underscored that no single tool sufficed for all cases (www.egnyte.com). A best practice is to test the extraction on representative lab documents and refine the chunking and format (Markdown, JSON, etc.) for optimal downstream RAG performance (www.egnyte.com).

Embedding and Indexing

Once we have cleaned, chunked content, the next step is to **create embeddings and build indexes**:

- **Embedding Generation:** Each text chunk is transformed into a vector using an embedding model. Common choices include OpenAI's text-embedding-ada-002, Cohere embeddings, or HuggingFace models (e.g. sentence-transformers like `all-MiniLM`). Egnyte's architecture generates embeddings for every chunk and indexes them (www.egnyte.com). If data sensitivity demands on-prem or specialized models (e.g. HIPAA-compliant co-processor), fine-tuning or self-hosting a transformer might be necessary. The embedding model should be chosen based on your domain: a biomedical embedding model or proprietary model may better capture scientific terminology than a general-purpose one.
- **Vector Database (Index):** The embeddings are stored in a vector database or search engine that supports nearest-neighbor queries. Options range from FAISS (Facebook AI Similarity Search) for in-memory search, to managed services (Pinecone, Qdrant, Weaviate) or even Elasticsearch's native `dense_vector` mapping (Elastic documentation); configure kNN indexing using options supported by the deployed version. Egnyte uses Elasticsearch's HNSW integration, leveraging their existing Elasticsearch expertise (www.egnyte.com). Each chunk's record in the index includes its embedding vector and relevant metadata.
- **Hybrid Indexing:** For best recall, it is common to create both a **semantic (vector) index** and a **keyword index** (BM25). The keyword index allows exact matching on important terms or metadata (e.g. chemical names, document tags), which pure vector search might miss. Egnyte's system indeed combines vector search results with BM25 results, filtering with metadata to respect permissions and improve precision (www.egnyte.com). Sawarkar et al.'s Blended RAG research reports improved retrieval on Natural Questions and TREC-COVID (authors' repository).
- **Scalability Considerations:** Research repositories can be large (thousands of experiments, millions of data points). Indexing must be efficient. Egnyte notes that indexing can explode: a 1 MB file might produce hundreds of chunks and as many vectors (www.egnyte.com). Thus, it's important to incrementally update indexes (e.g. processing only new/changed files) and potentially compress embeddings. Approaches like "namespace partitioning" for user data or vector quantization can help manage scale.
- **Versioning and Updates:** Whenever data changes (a new SOP is added, an assay is rerun), the index should be updated. Good practices include event-driven pipelines: Egnyte's "indexer" service listens for file upload events, extracts text, generates embeddings, and updates the index (www.egnyte.com).

Monitoring indexing latency and queue backlogs is also important, especially if tight SLAs (e.g. requiring "fresh" data within a day) are needed.

Retrieval: Search and Filtering

At query time, the RAG system must **retrieve relevant chunks** from the indexes:

- **Query Processing:** The user's natural-language query is also embedded into a vector and used to query the semantic index. In a simple implementation, one would retrieve the top *K* nearest neighbors by vector similarity. Egnyte does precisely this with FAISS/Elasticsearch HNSW to fetch top candidates (www.egnyte.com). In parallel, the query can be run as a BM25 search against the keyword index to catch any document mentions of the query terms. The results sets (vector-based and keyword-based) are merged and reranked. Studies suggest this hybrid retrieval significantly outperforms either method alone (www.egnyte.com) (www.techradar.com).
- **Permission and Metadata Filtering:** Crucially, results may be filtered by metadata such as user permissions, project tags, or content type. Egnyte's RAG enforces role-based access by post-filtering the retrieved chunks so that only authorized documents are used (www.egnyte.com). For example, a regulatory dossier in the system might be viewable only by compliance officers; the RAG must not retrieve or reveal it to unauthorized users. Similarly, if a question asks specifically about lab notebooks (e.g. "in the LIMS"), results from other systems can be deprioritized via metadata tags.
- **Reranking:** After initial retrieval, the candidate chunks are often reranked by a more sophisticated model. Egnyte implements a cross-encoder reranker: each chunk's relevance to the query is scored by a smaller transformer model (TinyBERT, for example) (www.egnyte.com). This cross-encoder takes the query and chunk together to output a similarity score, thus capturing nuanced matches. Egnyte found that using such a reranker on the top 50 or 100 chunks and then truncating to top 10 improved answer quality (www.egnyte.com). Rerankers are especially useful for short queries where nuances matter, but they add compute cost, so limiting to a few dozen candidates is common.
- **Result Compilation:** The final top-K chunks (often 5–20) form the retrieved context. Some systems may deduplicate overlapping content or ensure coverage of multiple sources. The chunks are then ordered and prepared for augmentation. At this stage, Egnyte's pipeline groups text by original document (and reorders by rank) to assemble a coherent prompt (www.egnyte.com).

Overall, retrieval in a corporate RAG is a balance of breadth and specificity. Too few chunks may miss crucial info; too many may overwhelm the LLM's context window. Empirical tuning is recommended: Egnyte experimented with chunk sizes and numbers, finding that ~10 chunks of ~1KB each was a good trade-off (www.egnyte.com).

Augmentation and Generation

Before constructing the prompt, treat retrieved passages, tool results, and conversation memory as untrusted input and keep them separate from trusted instructions (Microsoft security guidance).

The final phase appends the retrieved content to the query and invokes the LLM:

- **Prompt Construction:** The retrieved chunks are inserted into a prompt template along with the original question. A simple template might be: *"Use the following excerpts from our documents to answer the question. [Snippet 1] [Snippet 2] … [Snippet K] Question: [user's question]."* Egnyte goes further by prefixing each snippet with a short ID or source label, enabling the LLM to produce citations like "[Source 3]" (www.egnyte.com). They even found that short numeric IDs facilitated better processing than long UUIDs (www.egnyte.com). The prompt may also include system instructions like "Answer in a concise factual manner with references." Prompt engineering (e.g. adding reminders to cite sources) is critical to ensure the answer stays grounded.
- **LLM Invocation:** Choose a currently supported model and verify its deployment options, license, and data-handling terms. Instruct it to answer from retrieved evidence and cite sources. Retrieved context does not guarantee accurate answers; evaluate groundedness before deployment (Microsoft guidance).
- **Answer Processing:** After generation, the system may perform additional steps:
- **Citation Extraction:** Convert in-text references to actual hyperlinks or document names.
- **Quality Filtering:** If the model responds "I don't know" or hallucinates, implement fallbacks (e.g. return "No answer found" or trigger a rephrasing).
- **User Interaction:** Implement chat features (follow-up questions). Maintaining chat history is helpful for context as Egnyte notes (www.egnyte.com). Each follow-up query can be appended to the prompt history allowing the LLM to consider previous answers and refine as needed.
- **Logging and Audit:** For compliance, log which documents were retrieved and how the answer was formed (petronellatech.com).

Notably, Egnyte and others emphasize separating retrieval from generation even at this stage. For example, Egnyte's "Augmentation" blog shows that they rephrase the question first, rerun retrieval on that reformulation (improving recall), and then run generation (www.egnyte.com). This extra step can help handle ambiguities in the original query and ensure maximum relevant context is fetched. The generation step may also run internal consistency checks or smaller models to vet the final output before presenting to the user.

Tools, Frameworks, and Architectural Choices

Building a RAG system involves many moving parts. Below is a summary of key pipeline components and example technologies often used:

RAG Pipeline Component	Role	Example Tools/Approaches
Document Extraction	Parse raw files to plaintext (and structured data)	Unstructured (docs, PDFs), Apache Tika, PyMuPDF, OCR (Tesseract, Google Vision) (www.egnyte.com)
Chunking/Splitting	Divide content into manageable pieces	LangChain TextSplitter, custom scripts (by paragraphs/tables) (www.egnyte.com)
Embedding Generation	Convert text chunks into numeric vectors	OpenAI Embeddings API, Cohere Embed API, HuggingFace sentence-transformers (e.g. `all-mpnet-base`)
Indexing	Store embeddings for similarity search	FAISS (HNSW/IVF indexes), Elasticsearch (`dense_vector` mapping; check kNN index options for the deployed version) (Elastic documentation), Pinecone, Qdrant
Lexical Index	Optional keyword/BM25 search index	Elasticsearch or Solr with BM25; Whoosh for small setups
Retrieval (Semantic)	Find top-*K* similar chunks via vector search	FAISS k-NN query, Elasticsearch HNSW (www.egnyte.com)
Retrieval (Lexical)	Find top-*K* chunks by keyword relevance	BM25 search in Elasticsearch/Apache Lucene, filtering by metadata
Reranking	Re-score retrieved chunks for final selection	Cross-encoder models (e.g. `ms-marco-TinyBERT-L-2`) (www.egnyte.com), re-embedding similarity
Prompt Construction	Assemble query+context for LLM	Template-based injection of retrieved text, using identifiers for sources (www.egnyte.com)
LLM Answer Generation	Generate final answer grounded in context	GPT-4/GPT-3.5, Anthropic Claude, Llama 3, or domain-specific LLM finetuned on lab data
Post-Processing	Format output, attach citations, handle follow-ups	Custom logic to extract “source IDs” into links, conversation memory management

Table 2: Key components in a RAG system and typical technologies. Egnyte’s RAG pipeline, for example, uses FAISS, Elasticsearch, LangChain tools, and LLM APIs for these stages (www.egnyte.com).

As shown, many RAG systems use hybrid tech stacks. The choice of embedding model can affect performance: domain-tuned embeddings (e.g. trained on scientific text) often yield more relevant retrieval than general models (time.com) (thenewstack.io). For example, bioinformatics teams might prefer a PubMed-trained encoder when indexing life-science literature.

Elasticsearch is popular in enterprises like Egnyte because it offers both BM25 and ANN (with HNSW) search in one system (www.egnyte.com). Open-source vector DBs like FAISS offer low-latency search but require separate keyword indices for metadata. Commercial vector stores (Pinecone, Qdrant) simplify scaling but might raise concerns about locking proprietary data into a cloud service, especially under strict regulatory regimes. This leads to governance considerations discussed below.

For orchestration, frameworks like LangChain or LlamaIndex can accelerate development by providing connectors for many of the above tools, automatic chaining of steps, and integration with retrieval-RAG patterns. Nonetheless, organizations often build custom pipelines (as Egnyte did) to tightly control performance and security.

Case Studies and Real-World Examples

Multiple organizations have now built RAG-enabled systems on their internal data. We highlight a few illustrative cases:

Egnyte's Document Q&A and Knowledge Bases

Egnyte — a cloud file storage provider — has detailed its journey in implementing RAG for enterprise document search. Their initial **Document Q&A** feature allowed users to ask questions about a single file (www.egnyte.com). However, context limits of LLMs made it necessary to adopt RAG to answer questions spanning large documents. Egnyte's final solution was to generate embeddings for every document's text (using whatever LLM API), store them (in Elasticsearch/FAISS), and at query time retrieve the most relevant chunks via FAISS (www.egnyte.com). In practice, when a user asks a question, Egnyte's backend checks for pre-computed embeddings, searches top chunks in-memory, and calls the LLM with those chunks plus the question to produce an answer (including citations) (www.egnyte.com).

Building on Document Q&A, Egnyte created **Knowledge Base Q&A**: users can designate an entire folder as a "Knowledge Base" on which to run RAG queries (www.egnyte.com). For such a folder, Egnyte's indexer service extracts and indexes all files within, treating them as a unified corpus (www.egnyte.com). The user can then pose queries like "Give me policy on travel reimbursement," and the system performs a hybrid search (semantic + keyword) over that entire folder (www.egnyte.com). Notably, Egnyte's architecture enforces permissions at query time: it filters vector search results so that users only see chunks from files they have access to (www.egnyte.com).

Egnyte reports that customer feedback was overwhelmingly positive once RAG was implemented (www.egnyte.com). Their setup demonstrates many best practices: hybrid indexing, role-based filtering, short chunk IDs for prompt clarity, and reranking. The cited benchmark is external research: Sawarkar et al.'s Blended RAG study evaluates retrieval accuracy across query configurations (original study). See Data Analysis below for its Natural Questions top-10 comparison. They also mention large cost trade-offs: embedding every new file is expensive, so they cache embeddings and only re-index changed documents (www.egnyte.com).

Egnyte's story highlights both feasibility and complexity. They tackled challenges like chunking tables (eventually storing tables separately in markdown to avoid split-chunk issues) (www.egnyte.com) and tuning chunk sizes (around 1KB) for their LLM context window (www.egnyte.com). Overall, Egnyte shows how an enterprise can build a "turnkey" RAG solution that exposes internal knowledge (file contents, policies, etc.) to LLM-driven Q&A.

Pharmaceutical R&D: ELN/LIMS RAG

BioSymetrics describes a RAG-driven LLM interface to its Phenograph knowledge graph (BioSymetrics).

Rag2Mol is a retrieval-augmented E(3)-equivariant graph neural network for molecular generation (original paper). Separately, Kimura et al.'s drug-mapping study uses Hit Rate to measure whether appropriate OSV drugs appear among the mapping candidates (original study). These findings should not be used as measurements of internal-document LLM answer accuracy or ELN/LIMS productivity.

An industry example: a pharmaceutical R&D team might ask the RAG system "Which experiments used reagent Y in the last year?" The system would retrieve relevant ELN entries (noting reagent use), LIMS records (sample inventories showing Y), and any SOP docs, then generate an answer citing each source. IntuitionLabs emphasizes that such unified querying is exactly RAG's promise (intuitionlabs.ai). It eliminates the manual cross-search of ELN search panels, LIMS GUIs, and document drives. Treat potential R&D savings as a hypothesis to test in a pilot; retrieved passages increase input tokens and can increase costs (Microsoft guidance).

However, pharma is heavily regulated, so security of data (patient info in trials, proprietary compounds, etc.) is paramount. We explore this in the risks section, but it's worth noting pharma RAG prototypes often operate on anonymized or clearly permissioned data.

Enterprise Knowledge and Other Use Cases

Airbnb's Knowledge Repo combines a process for contributing and reviewing research with a tool for presenting and distributing it (Airbnb Engineering). Tech companies like Microsoft recognized the value: internal metrics showed employees were wasting hours per week searching for information before RAG-based tools were deployed (medium.com). Moneypenny's April–May 2025 survey of 750 UK business decision-makers reported that 39% of businesses represented were using AI (original research). Advisory articles also discuss RAG as a way to ground answers in corporate data (www.techradar.com).

In the life sciences context, a start-up perspective is exemplified by BioSymetrics' Elion platform, which advertises "RAG pipelines to query information from our vast internal knowledge base" (intuitionlabs.ai). These pipelines have practical effects: sales pitches claim that RAG can "slash overhead in lead identification" by mining all past experiments and latest publications simultaneously (intuitionlabs.ai).

A TechRadar opinion article recommends starting with small AI use cases, continual testing, and avoiding dependence on a single system (www.techradar.com). The combination of case studies and surveys consistently shows: where RAG has been properly implemented, domain experts get more reliable, evidence-backed answers than from generic chatbots (time.com) (www.techradar.com).

Data Analysis and Evidence of Impact

Quantitative studies of RAG in specialized domains are still emerging, but available data is encouraging. We highlight some key findings:

- **Retrieval Accuracy:** Sawarkar et al.'s Blended RAG research reports Natural Questions top-10 retrieval accuracy of 86.26% for BM25 + Best Field and 88.66% for KNN + Best Field, a 2.40-percentage-point difference (authors' results). These are retrieval results, not measurements of Egnyte customer answer accuracy.
- **Answer Quality:** In contexts where answers require multi-step reasoning (e.g. combining two sources), RAG significantly outperforms vanilla LLMs. The Lewis et al. (2020) paper showed RAG models beat standard seq2seq on Wikipedia-based question answering tasks (studylib.net). In the separate drug-mapping study, Hit Rate concerns inclusion of appropriate OSV drugs among mapping candidates (Kimura et al.); it should not be described as final answer accuracy.
- **User Productivity:** Test retrieval quality, answer accuracy, and citation support before rollout (Microsoft guidance). Measure any productivity improvement against the existing workflow in a pilot.
- **Cost and Performance Trade-offs:** Indexing large scientific corpora is resource-intensive. Egnyte notes that a 1 MB PDF might split into ~250 chunks of 4 KB embedding each, requiring ~1 MB memory and more on disk for the index (www.egnyte.com). This scaling can impact budget. Moreover, latency can increase if chunk sizes are too large. Thus, companies often invest in more hardware or incremental updates. Benchmarking RAG latency (end-to-end Q&A time) is an important metric for viability; however, specific figures are proprietary.

Overall, the available evidence indicates that **RAG substantially improves the factual grounding of AI answers in domain-specific contexts** (time.com) (www.techradar.com). In an internal R&D environment, the value comes not only from raw accuracy, but also from surfacing hidden knowledge (e.g., negative results buried in notes) and reducing redundant work (intuitionlabs.ai).

Challenges, Risks, and Limitations

Despite its promise, RAG over internal data is a complex undertaking with several pitfalls:

- **Structured Data (Tables & Graphs):** As Egnyte experienced, RAG models struggle with multi-dimensional tables when naively flattened (www.egnyte.com). Valuable data in spreadsheets or LIMS records can be misinterpreted if chunked poorly. Advanced processing (table-to-text conversion, graph queries) is necessary. Similarly, images and diagrams (e.g. molecular structures, charts) are often beyond pure text LLMs' capability. Integrating multi-modal models or converting figures to descriptive captions is an ongoing area of work.
- **Compliance and Privacy:** Indexing sensitive information (patient data, PII, unannounced IP) into a RAG vector store creates new attack surfaces (petronellatech.com). For a HIPAA covered entity or business associate, cloud providers processing ePHI on its behalf require an appropriate business associate agreement (BAA), alongside compliance with the applicable HIPAA Rules (HHS cloud guidance). Document the risk analysis (HHS risk-analysis guidance). Complete this review before sending ePHI to embedding, model, or storage providers, including reviewing service configurations and contractual use/retention terms. Treat encryption and self-hosting as deployment choices to assess, rather than proof of compliance. Mitigations include namespace isolation (segregating data by project), on-demand retrieval only (not storing raw sensitive fields), and the ability to delete specific vectors when

data lifecycle ends (thenewstack.io) (petronellatech.com). Zachary Proser of Pinecone notes that RAG allows user-specific data to remain external to the model, and it can be deleted from the index on demand, which is better for privacy than fine-tuning (thenewstack.io). Nevertheless, governance must explicitly address RAG: The PetronellaTech advising outlines the need for classification, lineage, and data "vetting" before indexing (petronellatech.com).

- **Data Quality and Bias:** RAG is only as good as the indexed data. Incomplete, outdated or low-quality lab notes will yield flawed answers. Organizations must ensure that sources are curated (e.g. discarding test data or duplicates) and updated. Moreover, if the corpus contains errors or biases (a known issue with LIMS data), the AI will incorporate them. Regular reviews by domain experts are recommended.
- **Evaluation Metrics:** Unlike standard search, evaluating a RAG system is challenging because correctness is often subjective. Existing benchmarks like NQ or HotPotQA test open-domain QA, but enterprise scenarios lack public datasets. Some papers (including Lewis et al.) use F1 and Exact Match on known QA sets (studylib.net), but a corporate user usually needs confidence that the cited sources truly answer the question. Companies often resort to A/B testing RAG vs. baseline search, or manual SME evaluation of answers. This gap in standardized evaluation is a limitation in rapidly proving ROI.
- **Operational Complexity:** As noted earlier, building RAG requires orchestrating many services (ingestion, databases, LLM APIs). Ensuring system reliability and low latency at scale can be difficult. It may require specialized infrastructure (GPU servers, distributed indexes) and DevOps effort. Not all organizations have this expertise in-house, which is why third-party vendors or consultancies are now emerging for RAG deployments.

In summary, RAG must be implemented with care. Technical challenges (especially around structured data and model limitations) must be met with engineering solutions (e.g. table-specific parsing (www.egnyte.com)). Meanwhile, governance and security are not optional – building enterprise RAG is as much about managing risk as about models (petronellatech.com). The next section discusses how to address some of these in an operational system.

Implications and Future Directions

Looking ahead, RAG's impact on research informatics is poised to grow:

- **Knowledge Graph Integration:** The lines between RAG and knowledge graphs will blur. As noted, some organizations are already merging their ELN/LIMS into KGs where entities like compounds, genes, and assays are nodes (intuitionlabs.ai) (intuitionlabs.ai). Future RAG systems may retrieve not just text passages but also graph nodes or query results, converting them to text. For instance, a hybrid RAG+KG system could answer "list all kinase targets where compound X had IC50 < Y" by querying the KG, then explaining in natural language (intuitionlabs.ai). This could enable complex multi-hop queries beyond plain-document search. Active research is exploring "graph-augmented LLMs" to leverage graph embeddings in generation.
- **Multi-Modal RAG:** Lab data includes images (microscopy, gels), spectra, even raw instrument outputs. Multi-modal LLMs (those trained on images and text) are becoming capable of interpreting figures or tables as part of RAG. Egnyte's work on structured extraction (using specialized pipelines for

PDFs/Tables) is a first step; future systems might directly prompt an LLM that can see images to answer questions about graphs. Nvidia and others are pushing such capabilities. For example, if a gel image is in the ELN, a future RAG system could retrieve it and ask the LLM to read bands off the gel. This is still experimental but a likely progression.

- **Agentic and Conversational RAG:** Modern LLM frameworks introduce agents that plan chains of actions. In a RAG context, an agent could iteratively retrieve, ask follow-ups, or even execute tools (like a calculator). Imagine a RAG-enabled chatbot that, upon receiving a question, first runs a database query (via a tool), then retrieves documents, then composes an answer. This agent architecture can enable more complex workflows than a single-shot query. Use narrowly scoped, read-only database tools for research retrieval (Microsoft tool-use guidance). Validate tool arguments and enforce the caller's authorization in application code for every call (OWASP prompt-injection guidance). If write or other consequential actions are enabled, require explicit approval before execution (OWASP excessive-agency guidance).
- **Regulatory and Ethical Considerations:** As RAG becomes integral, regulators may issue guidelines on AI sources and transparency. Already, the idea of "AI hallucinations" is drawing concern. If RAG systems cite incorrect lab results, it could be regulatory trouble. We may see standards for "citable AI" where every fact must trace to a validated record. Moreover, there are questions about who controls the knowledge: e.g., if multiple companies pool anonymized data for RAG training, how are contributions credited (intuitionlabs.ai) (intuitionlabs.ai)? Ethical frameworks for private-data RAG will evolve, likely requiring auditability and consent mechanisms.
- **Enterprise Collaboration:** There is potential for cross-company RAG in precompetitive areas. The IntuitionLabs report envisions "anonymized knowledge graphs" shared among consortiums, amplifying RAG's power (intuitionlabs.ai). For example, multiple pharma companies might pool non-sensitive data on common targets into a shared graph, enabling each to query a far richer dataset via RAG. Initiatives like the Pistoia Alliance in pharma might drive this. However, IP and privacy issues make this challenging.
- **Democratization and Low-Code RAG:** Finally, as tools mature, RAG deployment will get easier. When planning connectors, distinguish repository products from documented RAG integrations. Thermo Scientific SampleManager LIMS Software supports manufacturing quality control and process development (Thermo Fisher Scientific). Non-technical staff may soon configure RAG knowledge bases via GUI. The tech stack table above illustrates components that could become managed services. For example, some vendors are working on "LLMOps" platforms that embed governance, vector DB, and LLMs in one bundle for enterprises.

In sum, RAG is at the cusp of transforming research knowledge management. Early adopters are already harvesting gains; a full-blown RAG-aware ecosystem (with integrated KGs, secure pipelines, and user-friendly interfaces) seems imminent. Organizations that invest now in the data pipelines and governance structures for RAG will likely reap competitive advantages in speed and innovation (intuitionlabs.ai).

Conclusion

Retrieval-Augmented Generation enables large language models to transcend their static training data by tapping an organization's own knowledge. For research institutions and labs using ELNs, LIMS, Egnyte, and similar repositories, RAG offers a way to unify fragmented information into a conversational QA system. This report has shown that setting up RAG over internal data involves careful attention to data ingestion (parsing and chunking scientific documents), indexing (semantic and keyword indices), retrieval (hybrid search and reranking), and generation (LLM prompting with context) (www.egnyte.com).

Case studies from Egnyte and the pharmaceutical industry provide concrete examples: Egnyte's RAG pipeline now lets users query company documents securely and has significantly improved answer accuracy (www.egnyte.com) (www.techradar.com). BioSymetrics describes a RAG-driven LLM interface to Phenograph (BioSymetrics). Separately, Rag2Mol reports experimental results on drug-candidate binding affinity and drug-likeness (original paper). Treat retrieval of buried lab results as a potential use case to test, rather than a documented pilot breakthrough.

However, the path has hurdles. Handling domain-specific formats (tables, images), ensuring data quality, and meeting legal constraints are non-trivial tasks (www.egnyte.com) (petronellatech.com). We summarized strategies to mitigate these: from specialized table extraction to privacy-by-design architectures. The key takeaway is that a robust RAG system requires both engineering and governance rigor. Technical optimizations (like chunk size tuning (www.egnyte.com) and hybrid indexing (www.egnyte.com)) must go hand-in-hand with policies on access and auditing (petronellatech.com) (thenewstack.io).

Looking forward, RAG's synergy with evolving AI (multi-modal models, knowledge graphs, agentic systems) promises even richer capabilities. It could soon become routine for a scientist to query "What yields did we get in the last 5 iterations of reaction to synthesize compound X, and flag any deviations?" and get an instant, referenced answer. We built this report to guide such integrations: providing a deep dive into RAG's history, today's best practices, evidence of impact, and where innovation is heading.

In closing, implementing RAG over internal research repositories is poised to become as fundamental to labs as the very LIMS and ELNs themselves. By aligning powerful AI with corporate data, organizations can unlock latent value in their archives, accelerate discovery, and maintain an auditable trail of knowledge. In a TechRadar opinion article, Dr. John Bates argues that organized, accessible enterprise content improves RAG effectiveness (www.techradar.com). For scientists, engineers, and knowledge managers seeking that edge, RAG offers a compelling path forward.

References: Inline citations include time.com (time.com), Egnyte technical blogs (www.egnyte.com), TechRadar (www.techradar.com), IntuitionLabs whitepapers (intuitionlabs.ai), and other sources.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://time.com/7012883/patrick-lewis
2. https://www.techradar.com/pro/why-ai-and-rag-need-document-management

3. https://www.egnyte.com/blog/post/beyond-plain-text-egnytes-journey-to-structured-data-extraction-in-rag-systems
4. https://www.egnyte.com/blog/post/part-1-how-egnyte-built-its-turnkey-retrieval-augmented-generation-solution
5. https://www.egnyte.com/blog/post/building-generative-ai-solutions-at-egnyte
6. https://github.com/ibm-self-serve-assets/Blended-RAG
7. https://learn.microsoft.com/en-us/azure/foundry/concepts/retrieval-augmented-generation
8. https://petronellatech.com/blog/compliance/securing-enterprise-rag-governance-vector-db-security-llmops-for-compliant-genai
9. https://thenewstack.io/building-privacy-aware-ai-software-with-vector-databases
10. https://studylib.net/doc/28408473/lewis-et-al.--2020--retrieval-augmented-generation-for-kn...
11. https://biosymetricswebsite.squarespace.com/blog/enabling-faster-drug-discovery-using-llms-and-rag
12. https://www.egnyte.com/blog/post/part-2-how-egnyte-built-its-turnkey-retrieval-augmented-generation-solution
13. https://unstructured.io
14. https://www.elastic.co/docs/reference/elasticsearch/mapping-reference/dense-vector
15. https://learn.microsoft.com/en-us/security/zero-trust/catalog-ai-defense-capabilities/input-context-retrieval-hygiene
16. https://arxiv.org/html/2404.07220v1
17. https://pmc.ncbi.nlm.nih.gov/articles/PMC12159289
18. https://pmc.ncbi.nlm.nih.gov/articles/PMC11570653
19. https://medium.com/airbnb-engineering/scaling-knowledge-at-airbnb-875d73eff091
20. https://medium.com/%40waikaarsaachinn/building-a-knowledge-powerhouse-creating-secure-rag-systems-for-enterprise-knowledge-sharing-5a61b1f57437
21. https://www.moneypenny.com/uk/resources/blog/the-state-of-ai-adoption-in-uk-businesses
22. https://www.techradar.com/pro/retrieval-augmented-generation-can-manage-expectations-of-ai
23. https://www.hhs.gov/hipaa/for-professionals/special-topics/health-information-technology/cloud-computing/index.html
24. https://www.hhs.gov/hipaa/for-professionals/security/guidance/guidance-risk-analysis/index.html
25. https://github.com/microsoft/ai-agents-for-beginners/blob/main/04-tool-use/README.md
26. https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html
27. https://genai.owasp.org/llmrisk/llm062025-excessive-agency
28. https://www.thermofisher.com/us/en/home/digital-solutions/lab-informatics/lab-information-management-systems-lims/solutions.html

THE PUBLISHER

About IntuitionLabs

IntuitionLabs is an AI consulting, custom software development and data engineering firm serving pharmaceutical, biotechnology, medical-device and other life-science organizations. We work with clinical, regulatory, medical-affairs, commercial, quality and IT teams to connect technology decisions with the work people need to accomplish.

AI consulting and adoption

Our AI enablement services cover readiness assessments, use-case selection, governance and policies, team workshops, adoption measurement and ongoing advisory support. We help organizations structure the information layer behind AI: source material, context, permissions and maintained knowledge that make generated answers useful and reviewable. Private LLM inference and hosted AI options support teams evaluating how to operate AI with appropriate control over their data and infrastructure.

Software, data and life-science workflows

IntuitionLabs develops custom software for pharma and biotech, integrates enterprise systems, and builds data engineering and business intelligence solutions. Areas of focus include AI agents, regulatory research, medical writing, medical affairs, CMC information, competitive intelligence and clinical-document workflows. Our eTMF intelligence work includes cross-system reconciliation and inspection-readiness support.

Enterprise platforms and regulated delivery

We provide Veeva services, application support, managed services, integrations and custom applications, alongside enterprise content work involving platforms such as Egnyte. For regulated workflows, our services include GxP enablement, computer-system validation and software development addressing 21 CFR Part 11 requirements. The applicable controls, validation responsibilities and acceptance criteria are defined for each engagement.

Work with IntuitionLabs

Explore AI enablement, pharma and biotech software development, data engineering and BI, and Veeva services. Contact IntuitionLabs to discuss your workflow, information sources and implementation needs.

IntuitionLabs publishes educational research to help life-science teams make informed technology decisions. Coverage of a product or organization does not imply a client relationship, endorsement or partnership.

Website: https://intuitionlabs.ai

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.