Retrieval-Augmented Generation (RAG) is one of the most exciting developments in the world of LLMs—especially for someone like you who’s deep into semantic workflows and reproducibility. Let’s break it down in a structured, example-driven way:

**🧠 What Is RAG?**

**Retrieval-Augmented Generation (RAG)** is a hybrid architecture that combines:

* **Information retrieval** (like semantic search from a vector database)
* **Text generation** (via a large language model)

Instead of relying solely on the LLM’s static training data, RAG dynamically pulls in relevant external content—documents, PDFs, web pages, etc.—to enrich the prompt before generating a response2.

**🔧 How RAG Works (Step-by-Step)**

1. **Document Chunking & Embedding**
   * External data (e.g., PDFs, HTML, Markdown) is split into chunks.
   * Each chunk is converted into a vector using an embedding model (like OpenAI, Cohere, or Google’s EmbeddingGemma).
   * These vectors are stored in a vector database (e.g., Qdrant, Pinecone, FAISS).
2. **Query Embedding & Retrieval**
   * A user query is embedded into a vector.
   * The system searches the vector DB for semantically similar chunks.
3. **Prompt Augmentation**
   * Retrieved chunks are injected into the LLM prompt using prompt engineering.
   * This augmented prompt gives the LLM fresh, context-specific knowledge.
4. **Response Generation**
   * The LLM generates a response using both its internal knowledge and the retrieved context.