CIKLUM

RAG
Intro

October 2025

CIKLUM

Speaker

Maksym Lypivskyi
Head of Cloud Platforms & AI Director

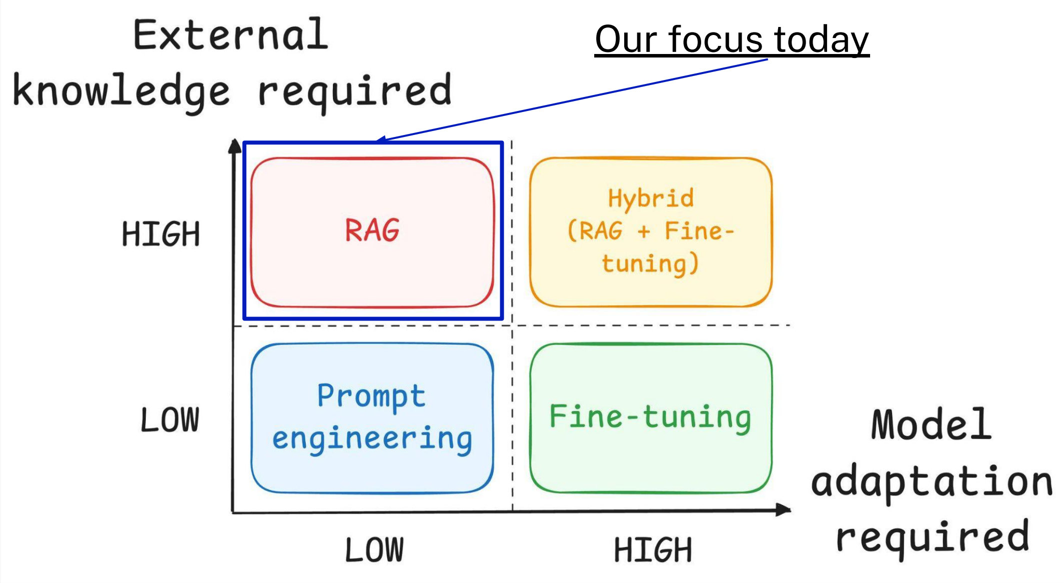
- 9 years at Ciklum driving large-scale cloud and software delivery initiatives
- 3 years specializing in AI
- Core interest: making AI systems reliable, production-ready, and business-impactful

Agenda

The AI Adaptation Landscape

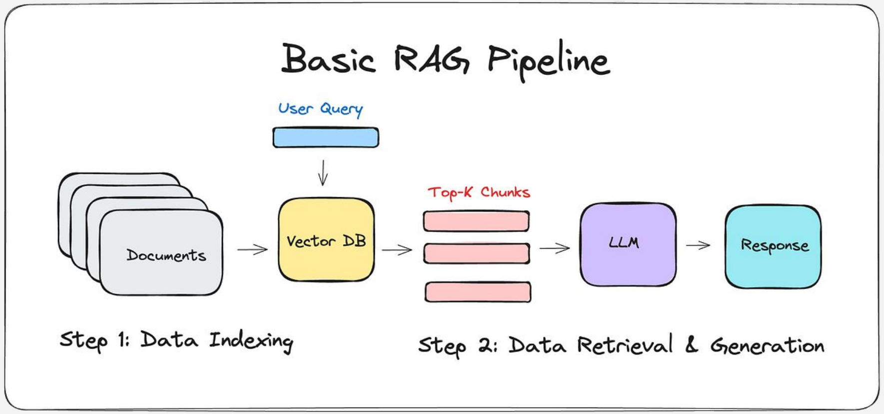
Where Does RAG Fit in Your AI Strategy?

- Need fresh, proprietary knowledge
- Don't want to retrain models
- Cost-effective scaling



What is RAG?

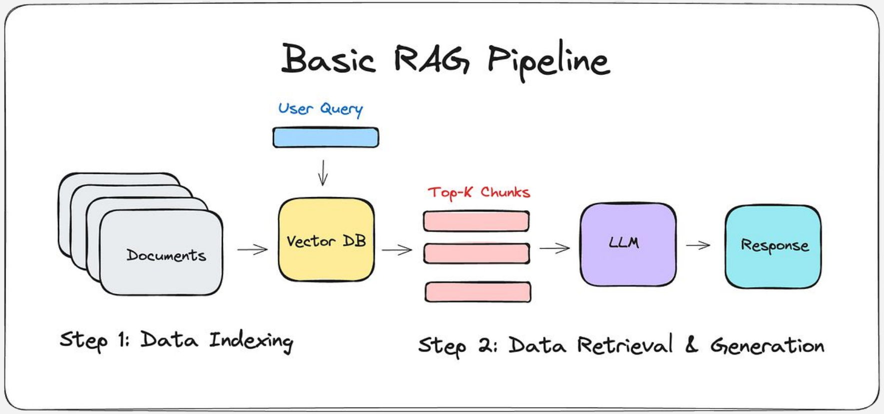
Retrieval-augmented generation (RAG) is a pattern that augments an LLM by retrieving relevant information from external sources at query time and injecting it into the prompt.



Why RAG Matters?

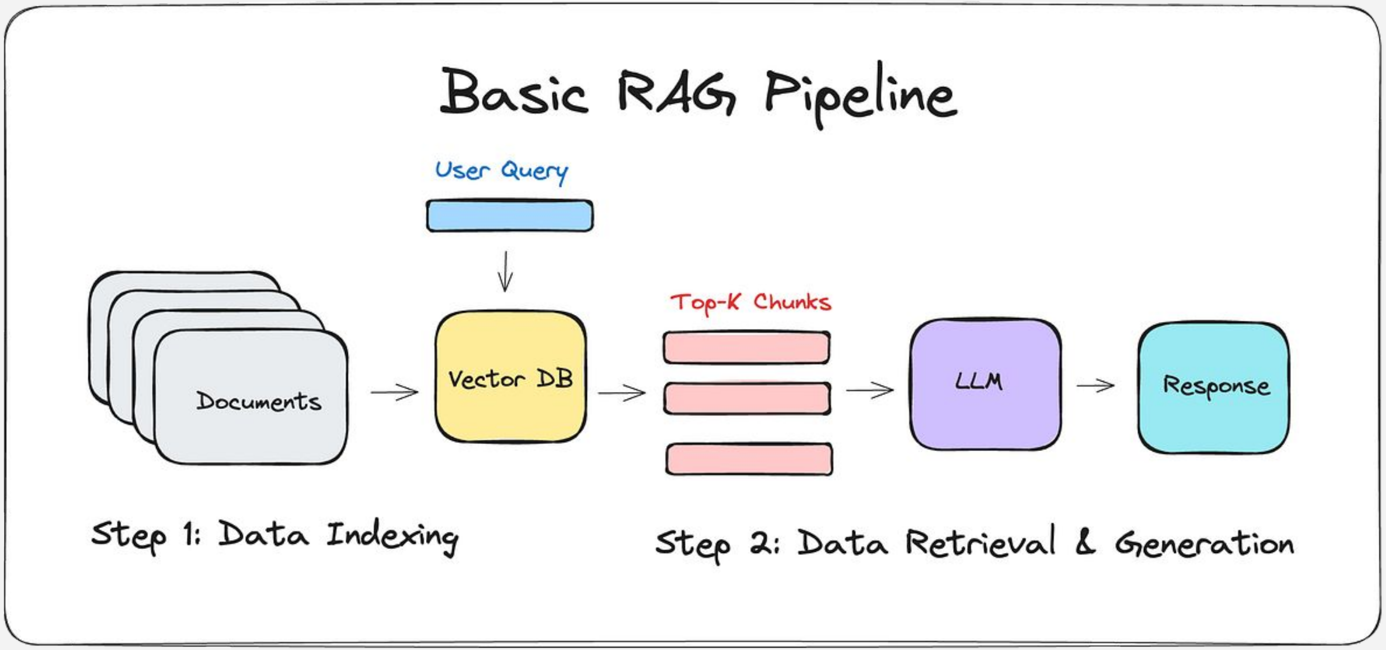
CIKLUM

- **Fresh Additional Knowledge.** RAG lets you ground answers in recent documents, internal wikis, or databases without retraining models.
- **Better Accuracy.** By retrieving authoritative evidence, the model generates more accurate answers and can cite sources.
- **Adaptable.** Effectively handles novel and niche queries that weren't in the model's training data.
- **Increases Efficiency.** RAG grounds prompts with smaller, targeted chunks of information that streamline retrieval and generation.



RAG Pipeline Deep Dive

CIKLUM



Chunking - The Critical Decision

CIKLUM

Why Chunking Matters

LLMs have limited context windows (32k-200k tokens). Large documents must be divided into smaller pieces.

The Challenge:

- Too large → loses specificity, poor retrieval
- Too small → loses context, fragmented information

```
Original Document (5000 words)
│
├─ Bad Chunking: Split mid-sentence, loses context
│  "...the company reported $500M revenue in Q4. The CEO"
│  "stated that growth would continue through..."
│
└─ Good Chunking: Complete semantic units
   "The company reported $500M revenue in Q4. The CEO
    stated that growth would continue through 2025 with
    expansion into European markets."
```

Three Main Chunking Strategies

CIKLUM

Strategy	How It Works	Pros	Cons	When to Use
Fixed-Size	Split at 512 tokens, 15% overlap	Simple, fast, predictable	May break mid-sentence	General purpose, starting point
Semantic	Use ML to identify coherent units	Preserves meaning, high accuracy	More compute, slower	Technical docs, complex content
Recursive	Split using separators (\n\n, \n, space) repeatedly until desired size	Respects structure, better boundaries	More complex than fixed-size	Documents with headings, paragraphs

Chunking Strategy Comparison

CIKLUM

[Fixed-Size]

```
Chunk 1: "The quarterly earnings report revealed a 23%
         increase in revenue. This growth was primarily
         driven by our new cloud services division,"
Chunk 2: "which expanded to serve 150 additional enterprise
         clients. The expansion required significant"
```

[Recursive]

```
Chunk 1: "The quarterly earnings report revealed a 23%
         increase in revenue."
Chunk 2: "This growth was primarily driven by our new
         cloud services division, which expanded to serve
         150 additional enterprise clients."
Chunk 3: "The expansion required significant infrastructure
         investment but resulted in improved margins."
```

[Semantic]

```
Chunk 1: "The quarterly earnings report revealed a 23%
         increase in revenue. This growth was primarily
         driven by our new cloud services division, which
         expanded to serve 150 additional enterprise clients."
         [Topic: Revenue Growth & Driver]

Chunk 2: "The expansion required significant infrastructure
         investment but resulted in improved margins."
         [Topic: Investment & Outcome]
```

Using Chunking Libraries

You Don't Need to Build from Scratch

CIKLUM

LangChain

Broad LLM application framework

Modular workflows where chunking is one piece of the puzzle

- Flexible TextSplitters
- Easy integration with agents
- Part of larger system

LlamaIndex

RAG-specific pipeline

High-performance, data-centric retrieval systems

- Sophisticated NodeParsers
- Produces optimized "Nodes"
- Built for ingestion/retrieval

RAG real world use cases

CIKLUM

- **AI Chatbots:** RAG provides accurate answers from internal knowledge bases (e.g., support wikis, legal documents). OpenAI emphasises that RAG is valuable when the content is not part of the base model's knowledge .
- **Search & discovery:** Search systems combine keyword and vector search to surface relevant documents in e-commerce, research and legal discovery.
- **AI Copilots:** Tools like Supabase AI Copilots use vector databases to ground responses in proprietary data and maintain multi-tenant isolation .
- **Long-context reasoning:** Databricks' long-context benchmark shows that Google's Gemini 2.5 models can maintain consistent performance on RAG tasks up to two million tokens (longer than most models), whereas OpenAI's GPT 5 models achieve state-of-the-art accuracy up to 128k tokens .

Common Challenges

- **Chunking & context windows:** If chunks are poorly defined, the retrieved information may miss critical context or include too much irrelevant text. Research by Analytics Vidhya notes that fixed-size chunking can break context while semantic-based chunking preserves meaning but requires more compute .
- **Model context length:** Models can only ingest a finite number of tokens. Databricks' benchmark observed that performance of LLMs like Llama-3.1 and GPT-4 starts to degrade when context windows exceed 32–64 k tokens .
- **Retrieval quality:** The quality of the vector database and retrieval algorithm determines recall. Missing relevant documents leads to hallucinated answers.
- **Latency & cost:** Large vector databases and embedding models can be expensive and introduce latency.

Do’s and don’ts for RAG

CIKLUM

Do’s

- ✅ Start simple, iterate based on metrics
- ✅ Use metadata filtering (product, language, permissions)
- ✅ Combine vector + keyword search (hybrid approach)
- ✅ Monitor retrieval quality (recall@k, precision)
- ✅ Keep embeddings synchronized with documents
- ✅ Evaluate with domain-specific questions

Don’ts

- X Rely solely on vector search
- X Ignore security and access controls
- X Overload the LLM context window
- X Neglect continuous updates
- X Skip evaluation frameworks

Three Main Chunking Strategies

Scenario	Stack	Key Reason
Learning	LangChain + Chroma + sentence-transformers + Ollama	• Learn fundamentals • Risk-free • Runs on laptop
MVP	LangChain + Qdrant (self-host) + OpenAI/Gemini embeddings + GPT-4o-mini	Professional quality at startup budget
Enterprise	LangGraph + LangSmith + Pinecone + OpenAI/Gemini + GPT-4	• Agentic workflows • Observability • SLAs

Default Configuration

(Works for 80% of cases):

- ✅ Chunk size: 512 tokens
- ✅ Chunk overlap: 15% (~75 tokens)
- ✅ Top-k retrieval: 3-5 chunks
- ✅ Embedding dimensions: 768-1536

Key Takeaways

What You Should Remember

CIKLUM

- ✅ RAG = Open-book exam for AI - Retrieves external knowledge at query time
- ✅ Chunking is critical - Start with 512 tokens, 15% overlap, then iterate
- ✅ Hybrid search > vector-only - Combine vector and keyword search
- ✅ Start simple - Use Chroma + LangChain for learning, scale as needed
- ✅ Always evaluate - Track recall, precision, and answer quality

CIKLUM

Thank you!