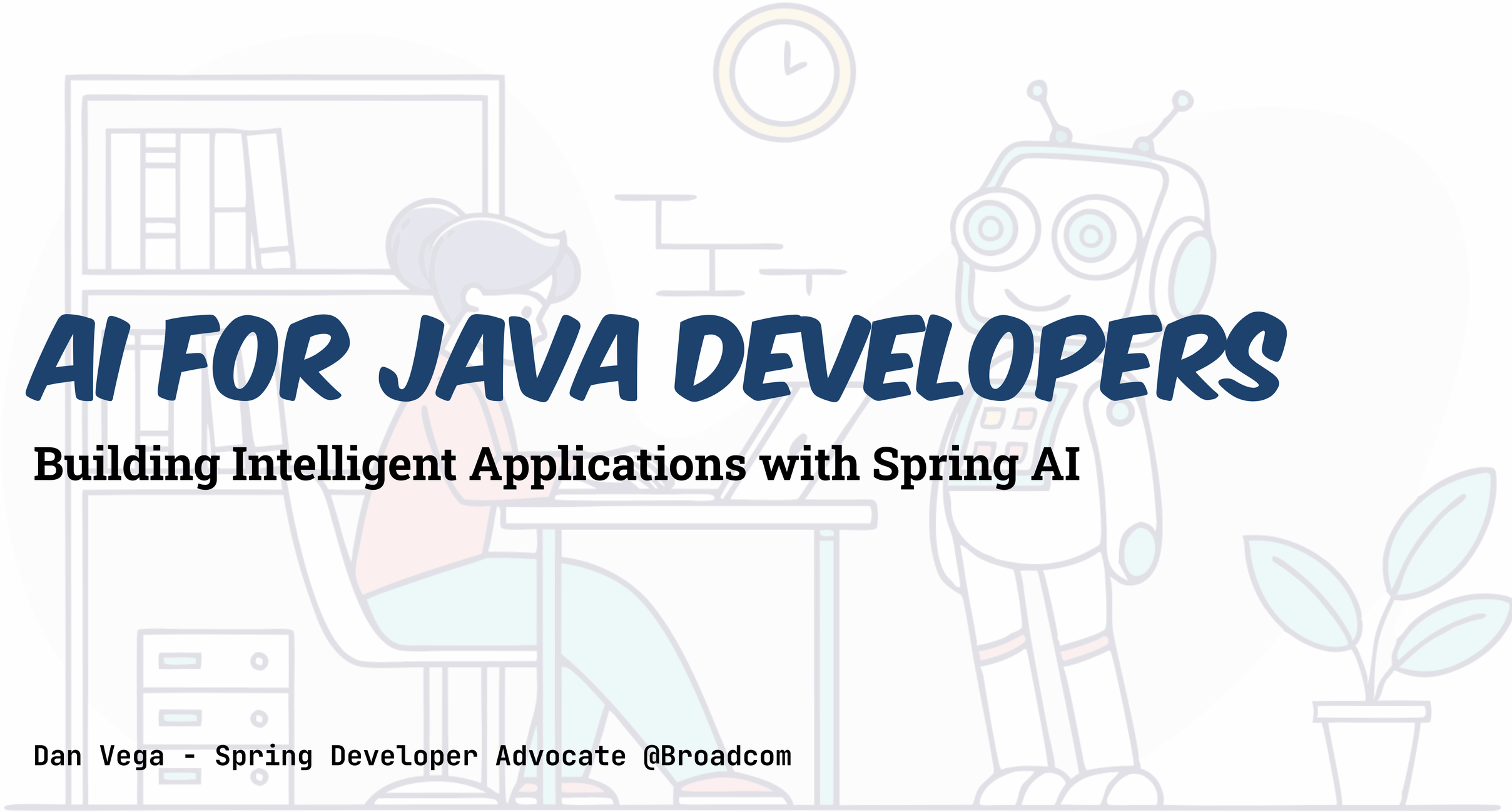


AI FOR JAVA DEVELOPERS

Building Intelligent Applications with Spring AI

Dan Vega - Spring Developer Advocate @Broadcom

https://spring.io/blog/

spring by VMware Tanzu

Why Spring | Learn | Projects | Academy | Community | Tanzu Spring

Spring blog | All Posts | Engineering | Releases | News and Events | RSS feeds

Spring AI 1.0 GA Released

RELEASES | MARK POLLACK | MAY 20, 2025 | 2 COMMENTS

On behalf of the Spring AI engineering team and everyone who contributed to this release, I am very excited to announce the general availability of Spring AI 1.0. We have a great release blog lined up for you.

Getting Started

All the new bits are in maven central. Use the provided bom to import the dependencies.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.ai</groupId>
            <artifactId>spring-ai-bom</artifactId>
            <version>1.0.0</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Checkout the Upgrade Notes for the latest breaking changes and how to upgrade. NOTE: You can automate the upgrade process to 1.0.0-GA using an OpenRewrite recipe. This recipe helps apply many of the necessary code changes for this version. Find the recipe and usage instructions at Arconia Spring AI Migrations.

You can get started creating 1.0 GA apps on the Intialzr website and read our Getting Started section in the reference documentation.

Friends and Family

Second are a selection of blogs created for the 1.0 GA release from the many friends and family we have been working with over the past two years, showing how to use Spring AI in various ways:

- Microsoft Azure Blog and Video. A special thanks to Asir Selvasingh who helped us launch Spring AI back at the Spring One conference in Vegas in 2023.
- AWS - Blog - Spring AI 1.0 Brings AI to the Developer Masses
- Google - Blog - Google Cloud and Spring AI 1.0
- Cloud Foundry - Video - Spring AI and CloudFoundry: Bootiful, Agentic, Production-Worthy, Cloud-Native Systems and

Get the Spring newsletter

Stay connected with the Spring newsletter

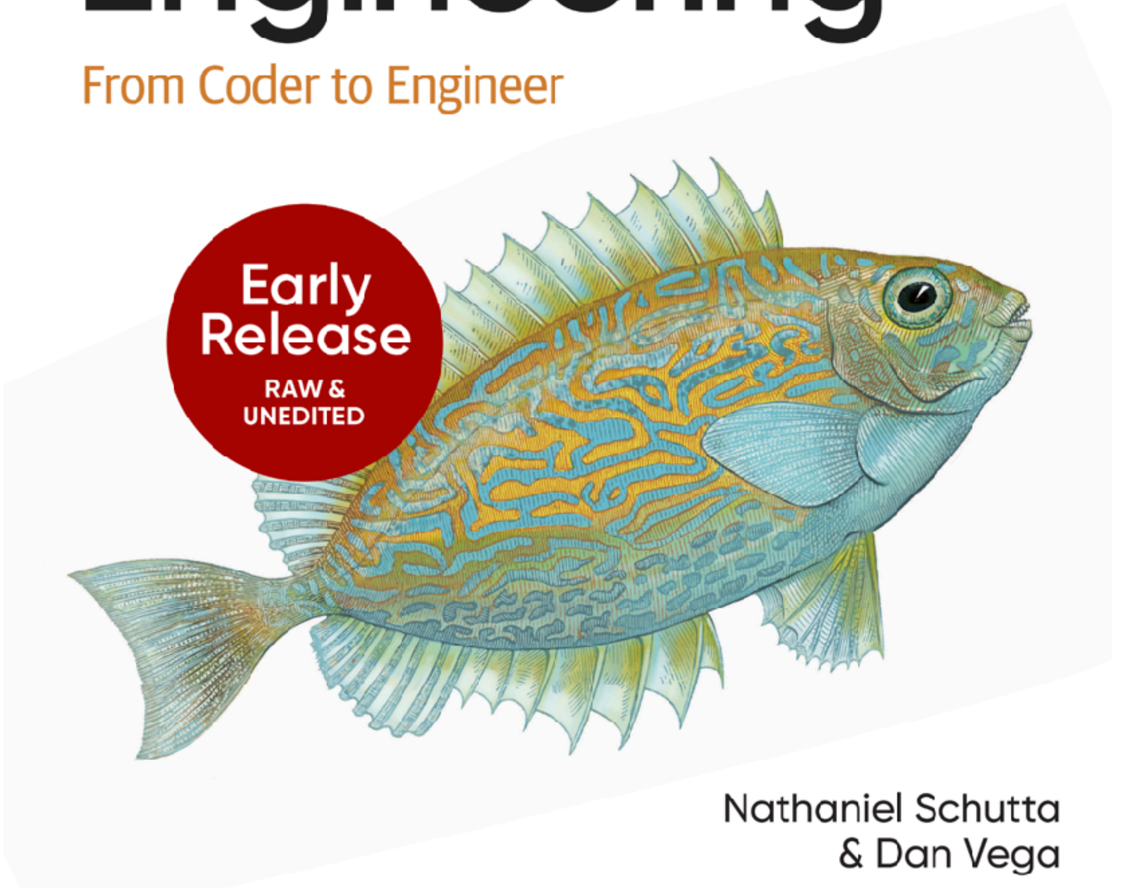
SUBSCRIBE

INTRODUCTION

ABOUT ME

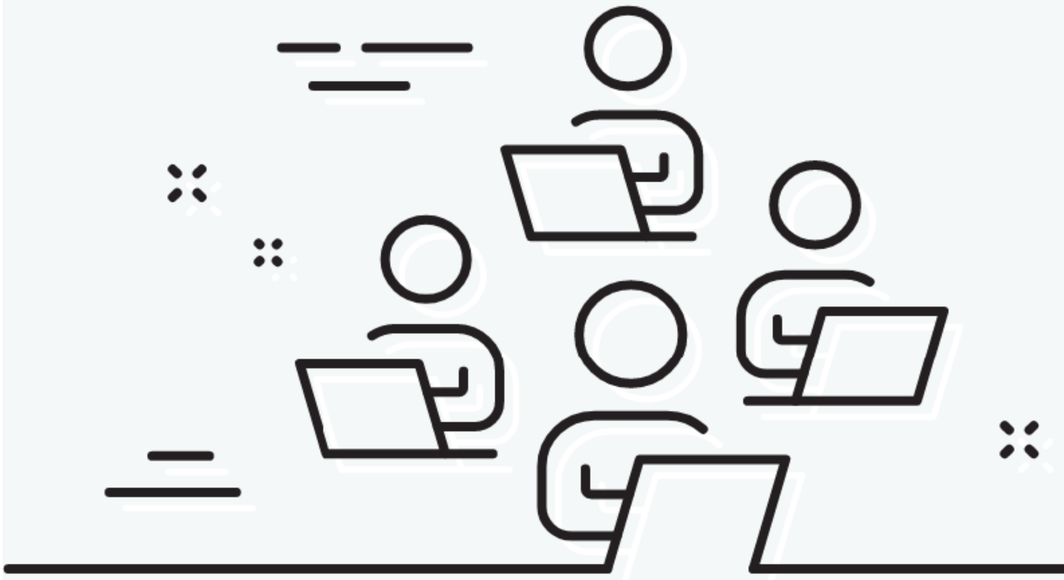
Learn more at danvega.dev

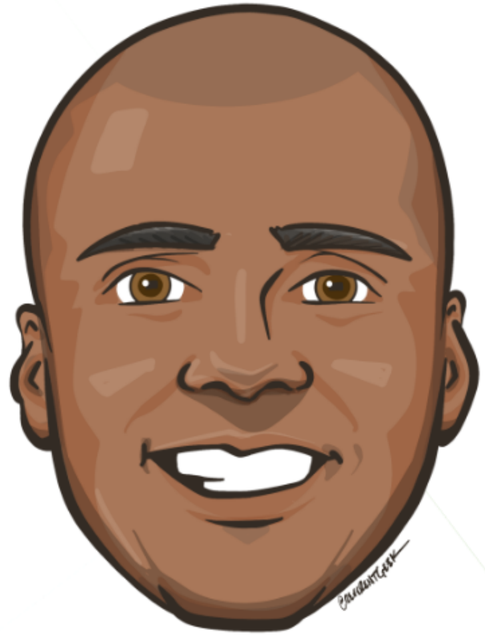
- 👨‍👩‍👧‍👦 Husband & Father
- 🏠 Cleveland
- ☕ Java Champion
- 👨‍💻 Software Development 23 Years
- 🍃 Spring Developer Advocate
- 📖 Author (Soon to be)



spring

OFFICE HOURS

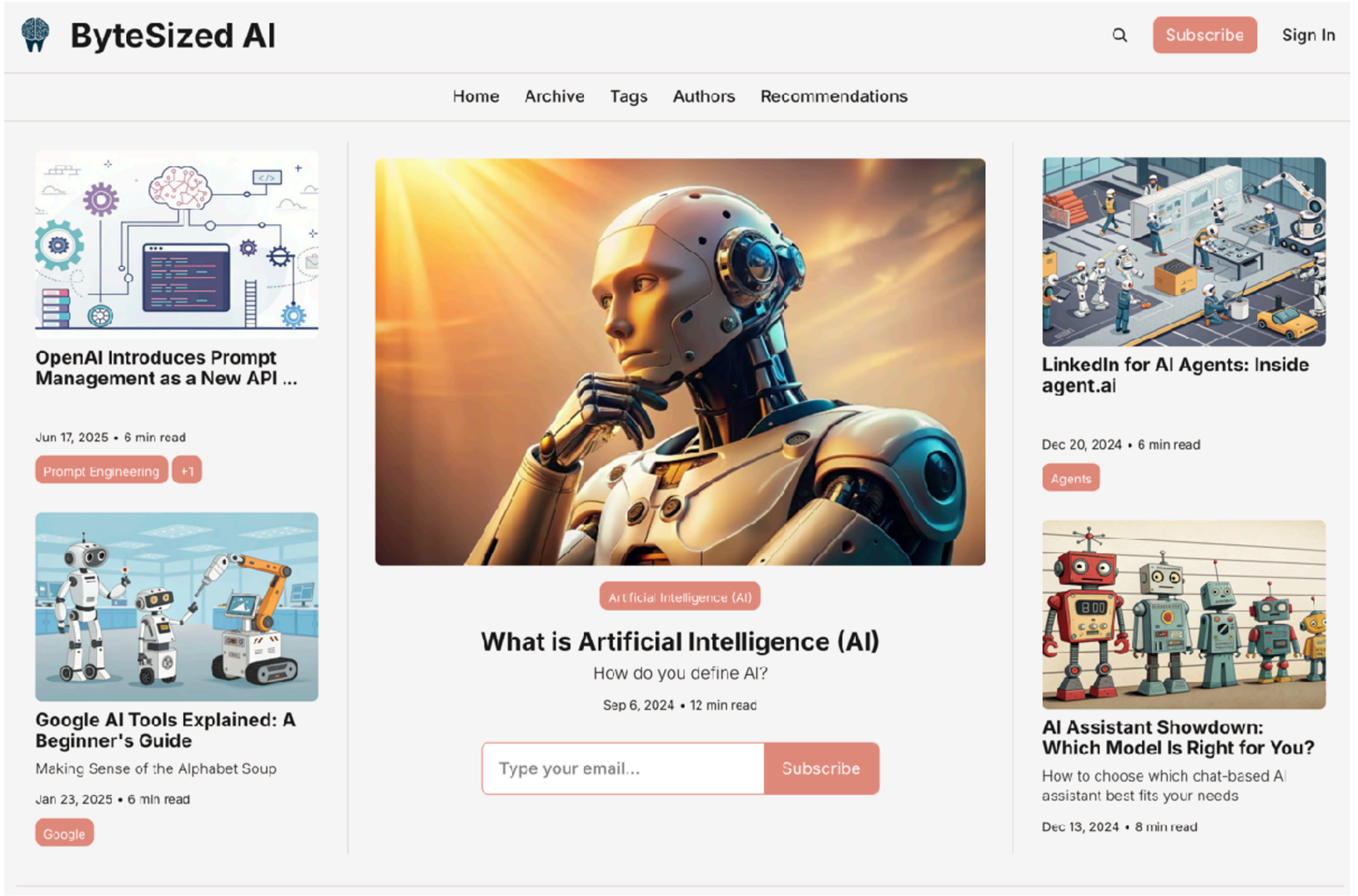




https://www.springofficehours.io

ByteSized AI

www.bytesizedai.dev



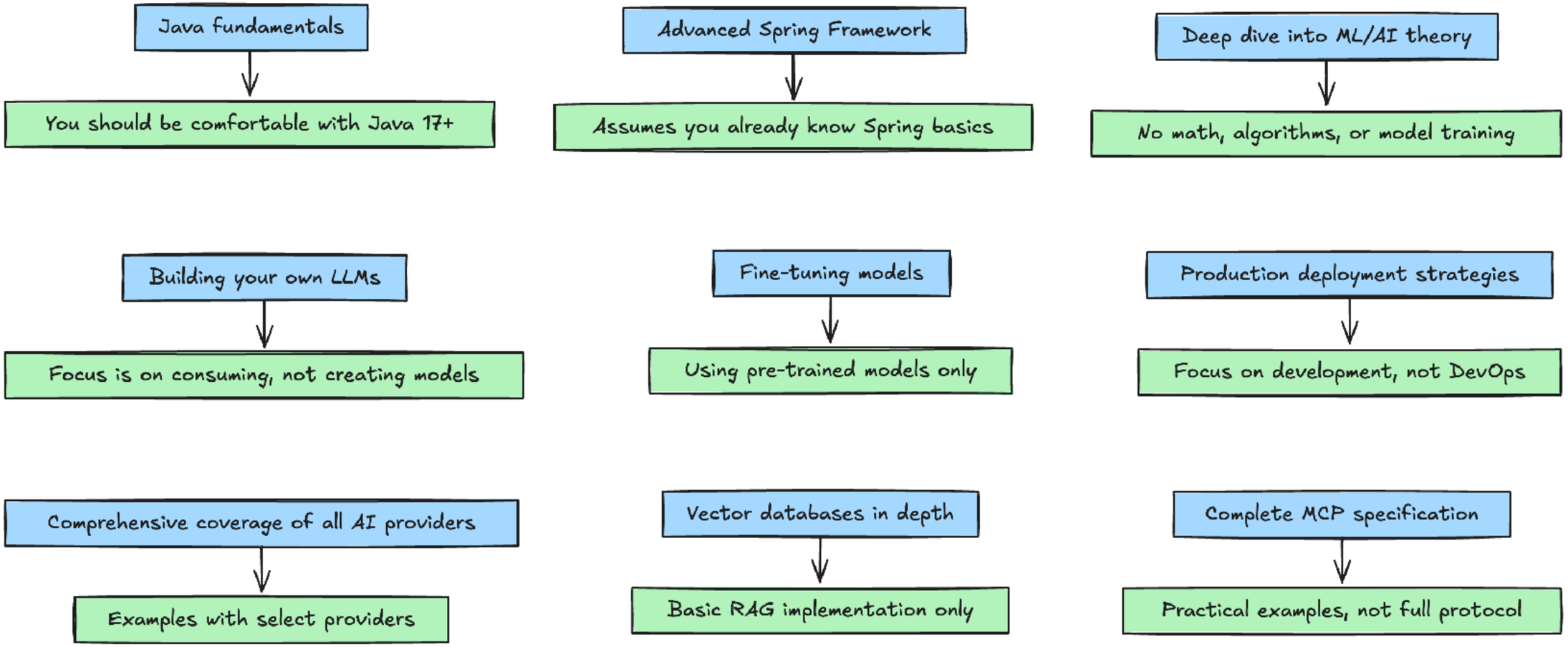
AGENDA

What we will cover in this course

- What is AI?
- Getting Started
- Spring AI Features
- Open Source vs Proprietary Models
- LLM Limitations
- Observability
- Evaluations
- Conclusion

AGENDA

What we will NOT cover in this course



HOW TO GET THE MOST OUT OF THIS COURSE

- **Use timestamps** - Jump to specific topics as needed
- **Access all resources** - Links, documentation, and code samples provided
- **Learn → Build → Share** - Apply each concept immediately in your own project
- **Code along actively** - Don't just watch, open your IDE and follow along
- **Pause and experiment** - Try variations of the examples shown
- **Join the Spring AI community** - Ask questions and share your implementations
- **Start with simple use cases** - Build a basic chatbot before complex features
- **Keep a learning journal** - Note ideas for your own projects as you watch
- **Watch in segments** - One section per day for better retention
- **Prepare your environment first** - Have API keys and tools ready before starting
- **Build a portfolio project** - Create one substantial AI-powered app by course end
- **Cross-reference documentation** - Keep Spring AI docs open while watching
- **Share your progress** - Tweet/post about what you're building for accountability
- **Focus on your domain** - Think how each feature applies to your specific industry/work

PREREQUISITES AND RESOURCES

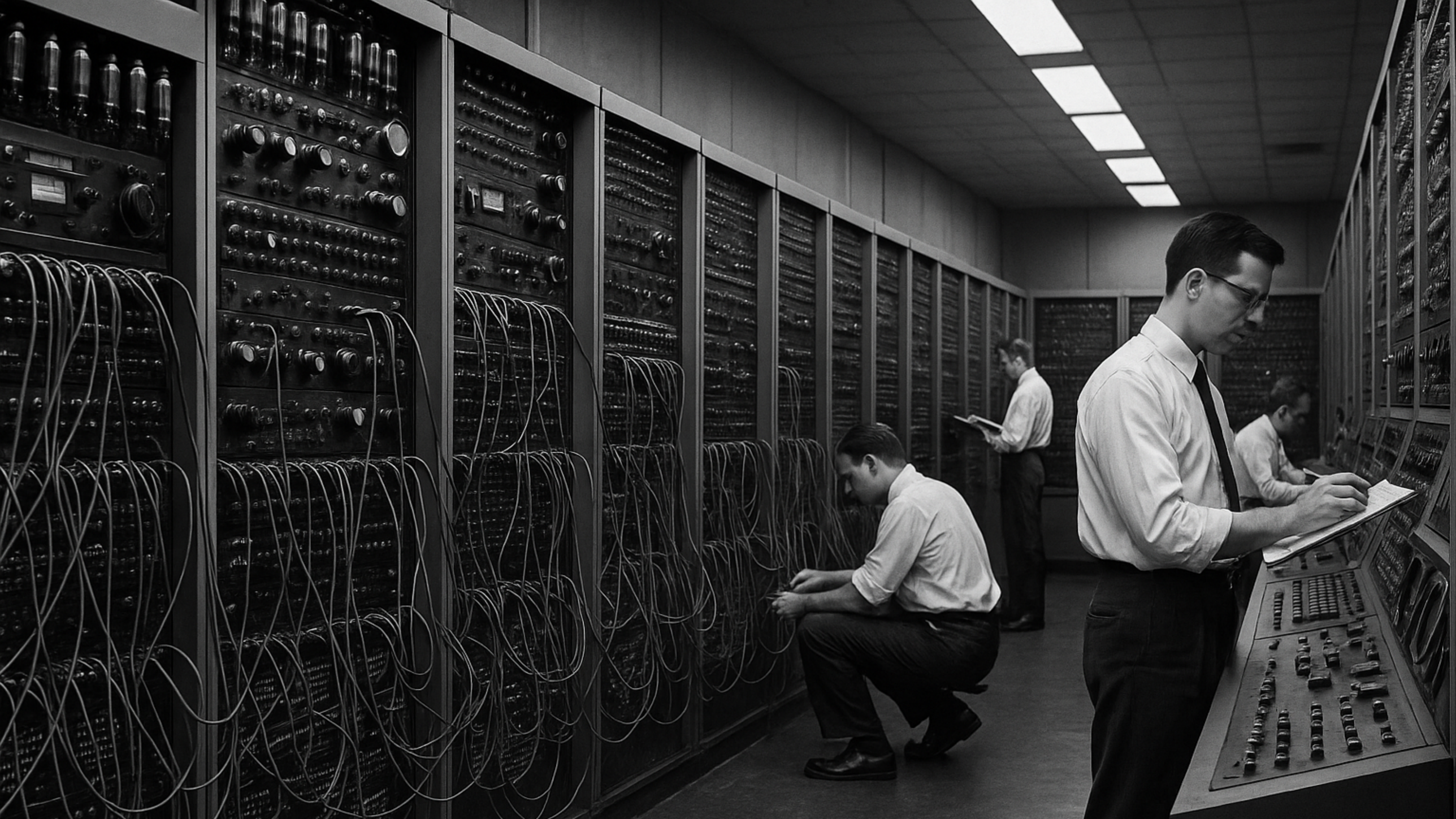
Prerequisites

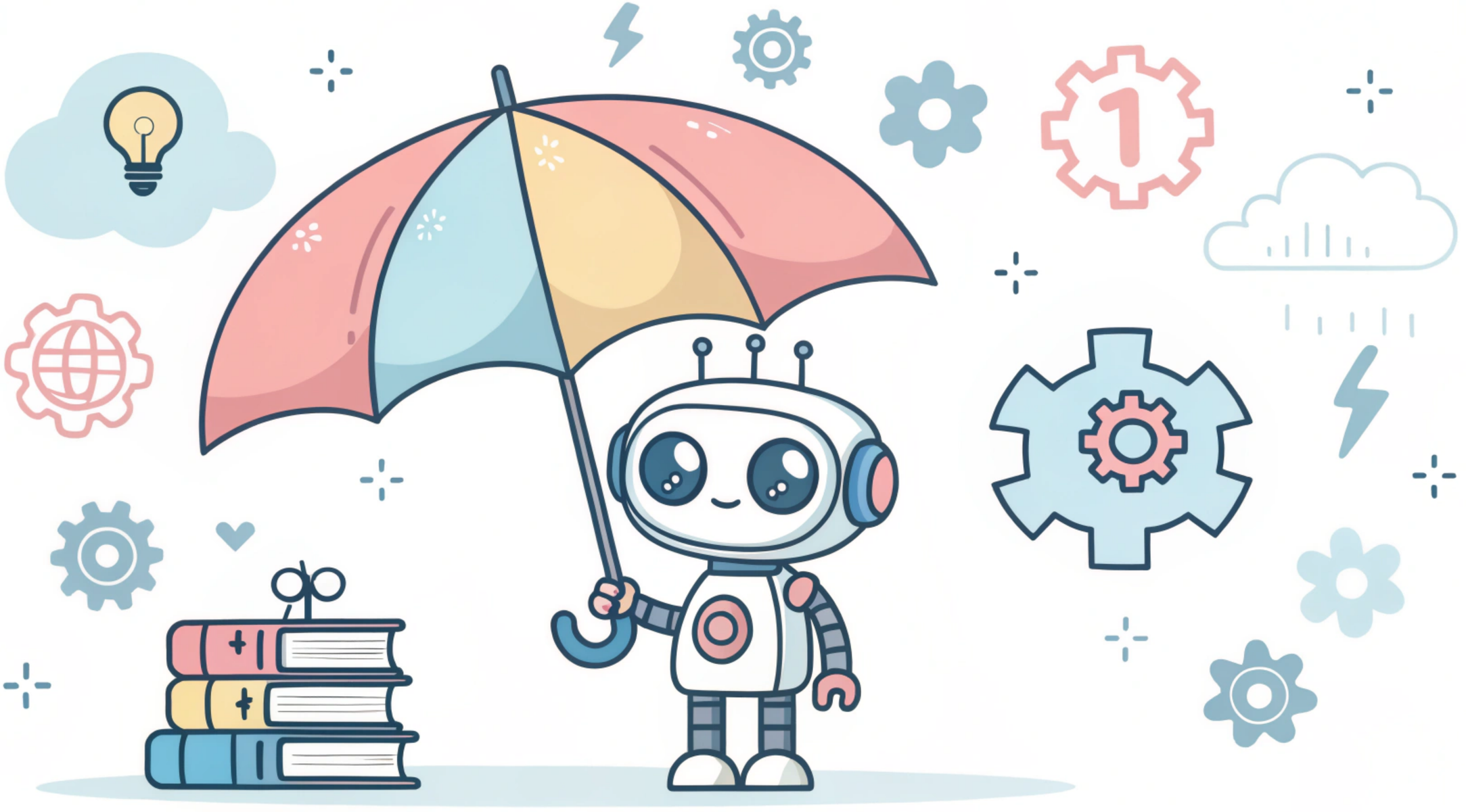
- Familiar with the Java programming Language
- Familiar with Spring (but not Spring AI)
- IDE (IntelliJ)

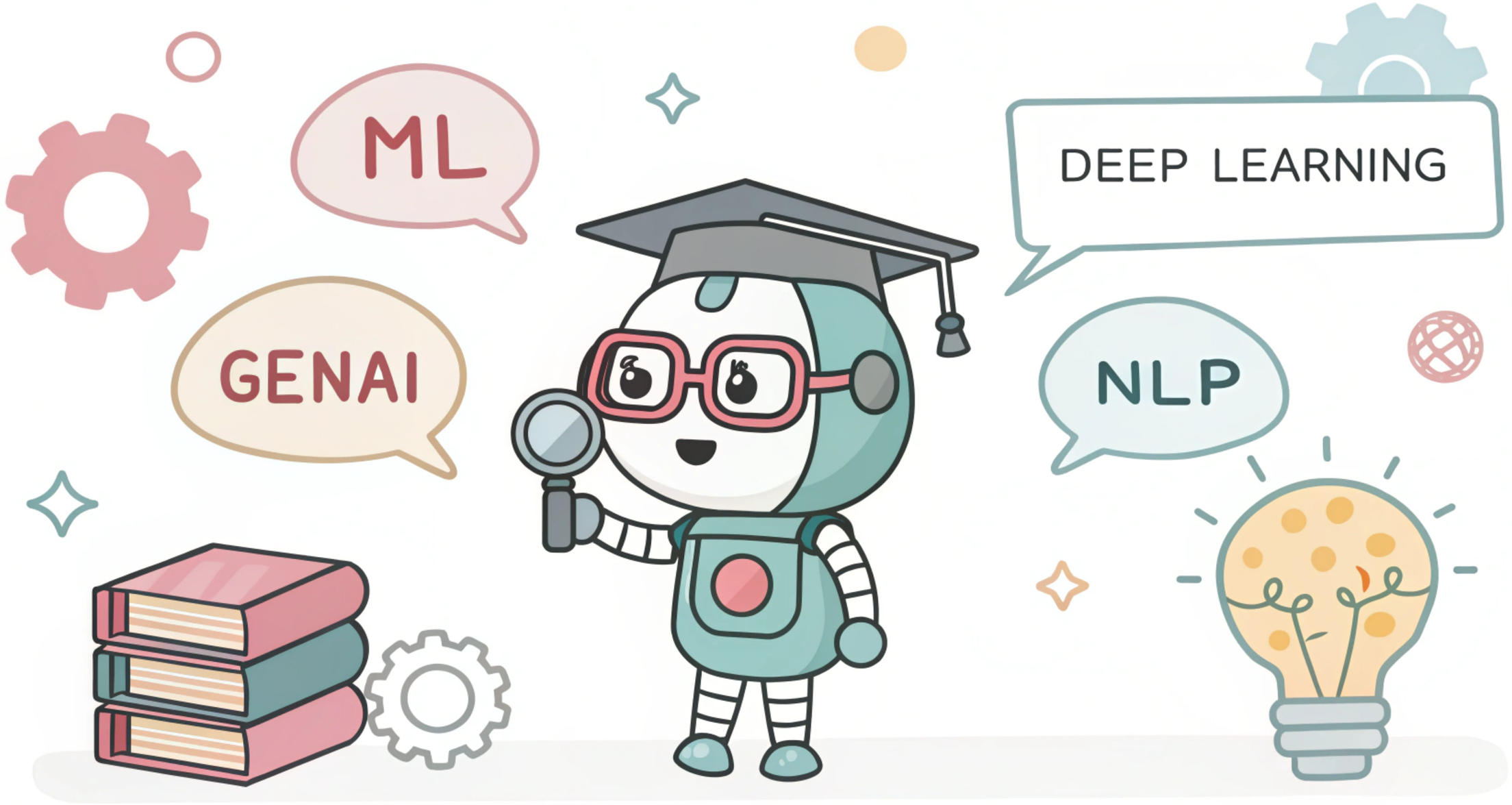
Resources

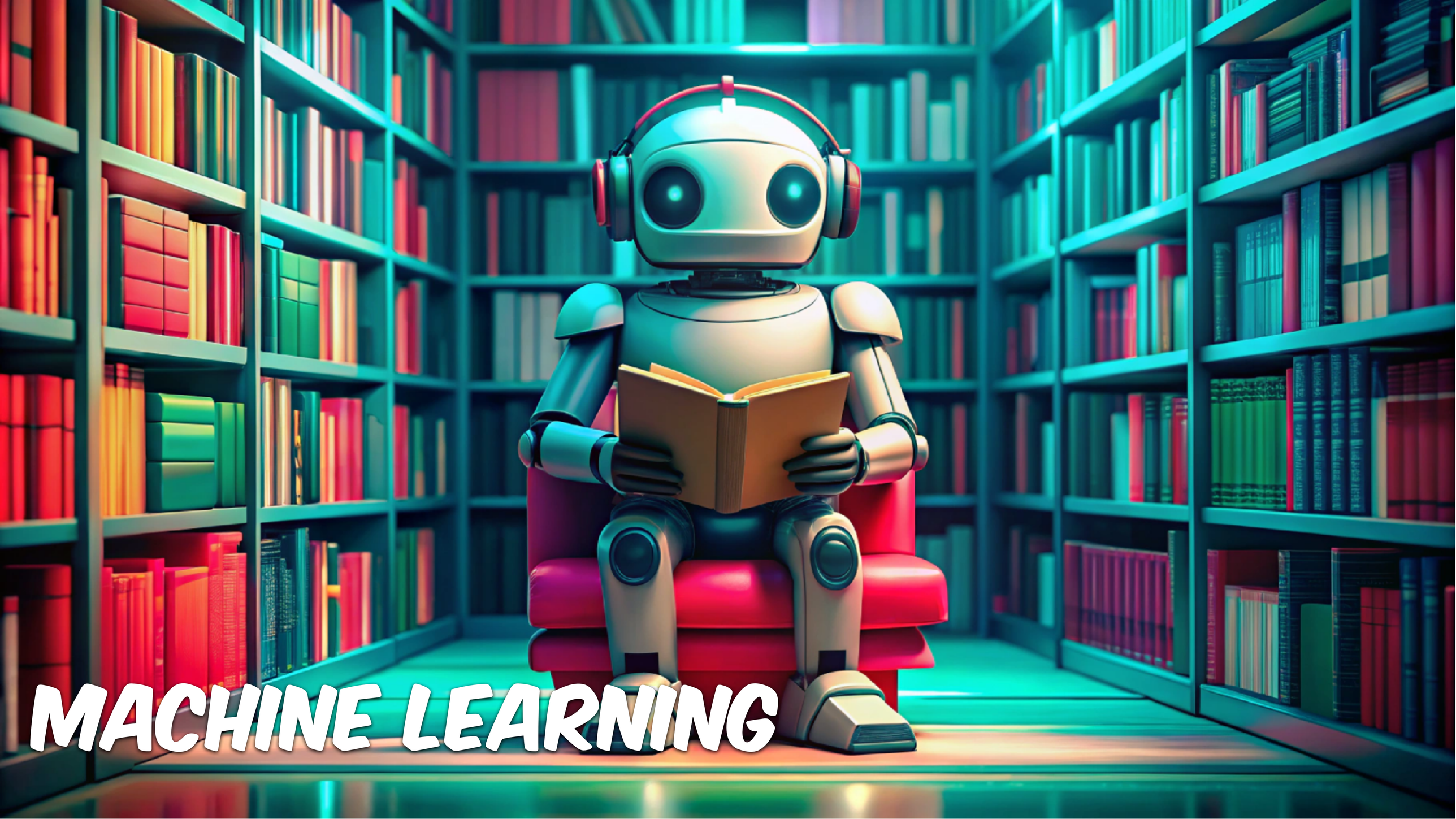
- http://www.danvega.dev
- https://docs.spring.io/spring-ai/reference
- https://github.com/danvega/spring-ai-workshop

WHAT IS ARTIFICIAL INTELLIGENCE (AI)






ML
GENAI
DEEP LEARNING
NLP


MACHINE LEARNING

MACHINE LEARNING

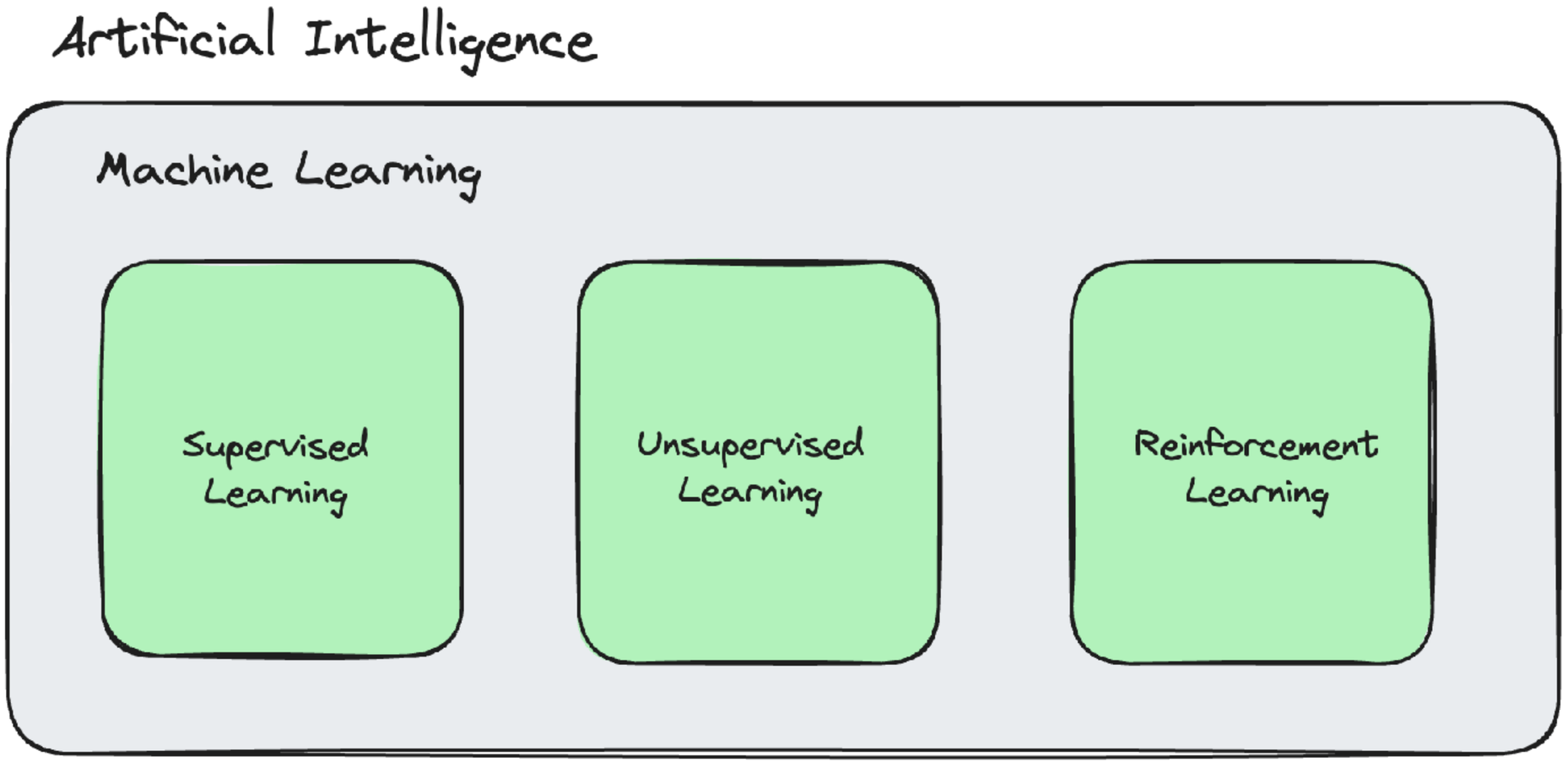
Artificial Intelligence

Machine Learning

Unlike traditional programming, where explicit instructions are provided for every scenario, ML systems learn patterns from data, allowing them to make predictions or decisions without being explicitly programmed for each possibility.

MACHINE LEARNING

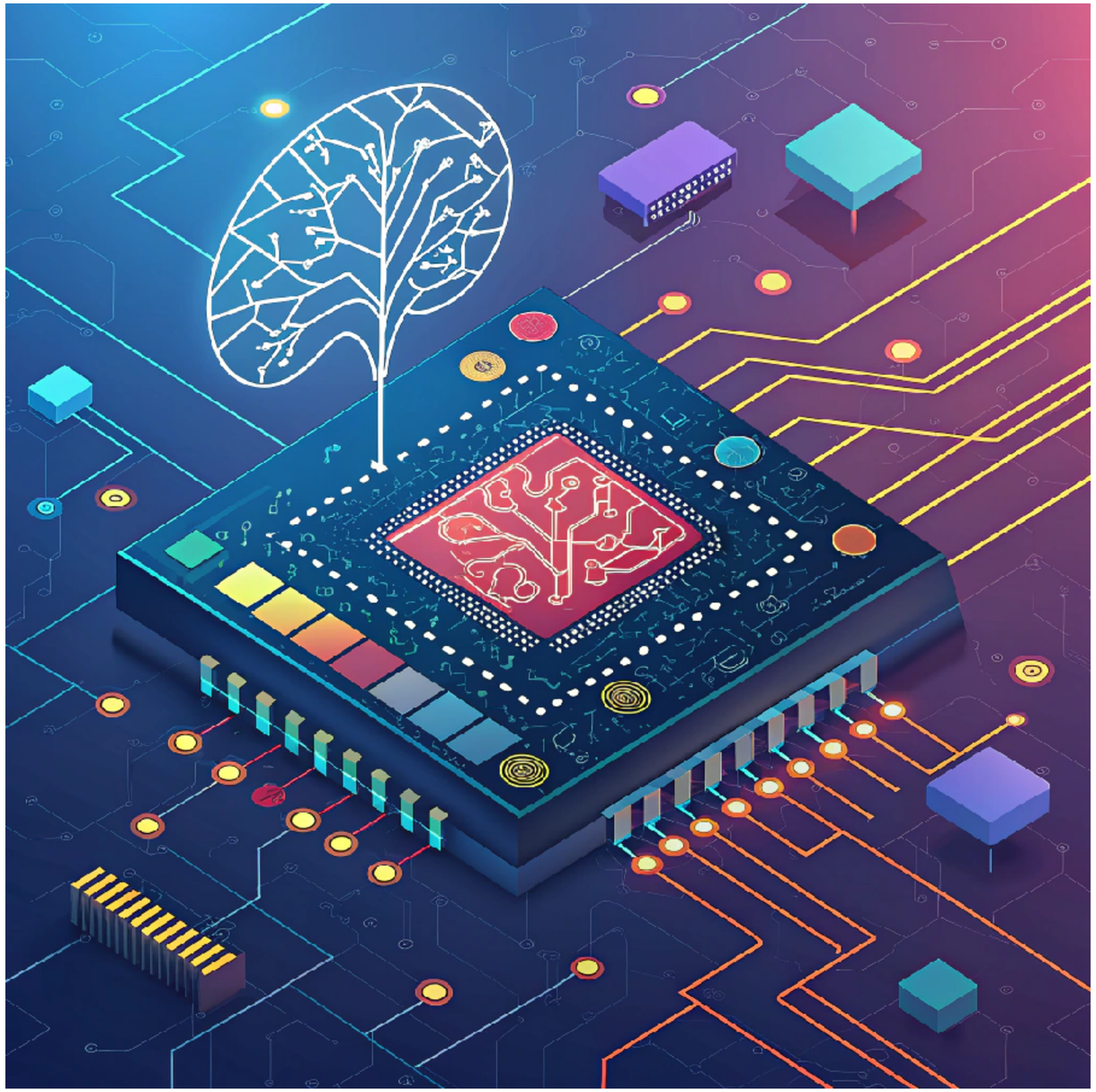
Machine Learning encompasses various techniques and types of tasks, including:



MACHINE LEARNING

Use Cases

- Facial Recognition
- Recognize Tumors on x-ray scans
- Abnormality on ultrasounds
- Self-drive mode
- Fraud Detection
- Credit Score / Loan Approval
- Product Recommendations (YouTube)
- Spam Filtering
- Search Ranking



SUPERVISED LEARNING

Labeling the training data

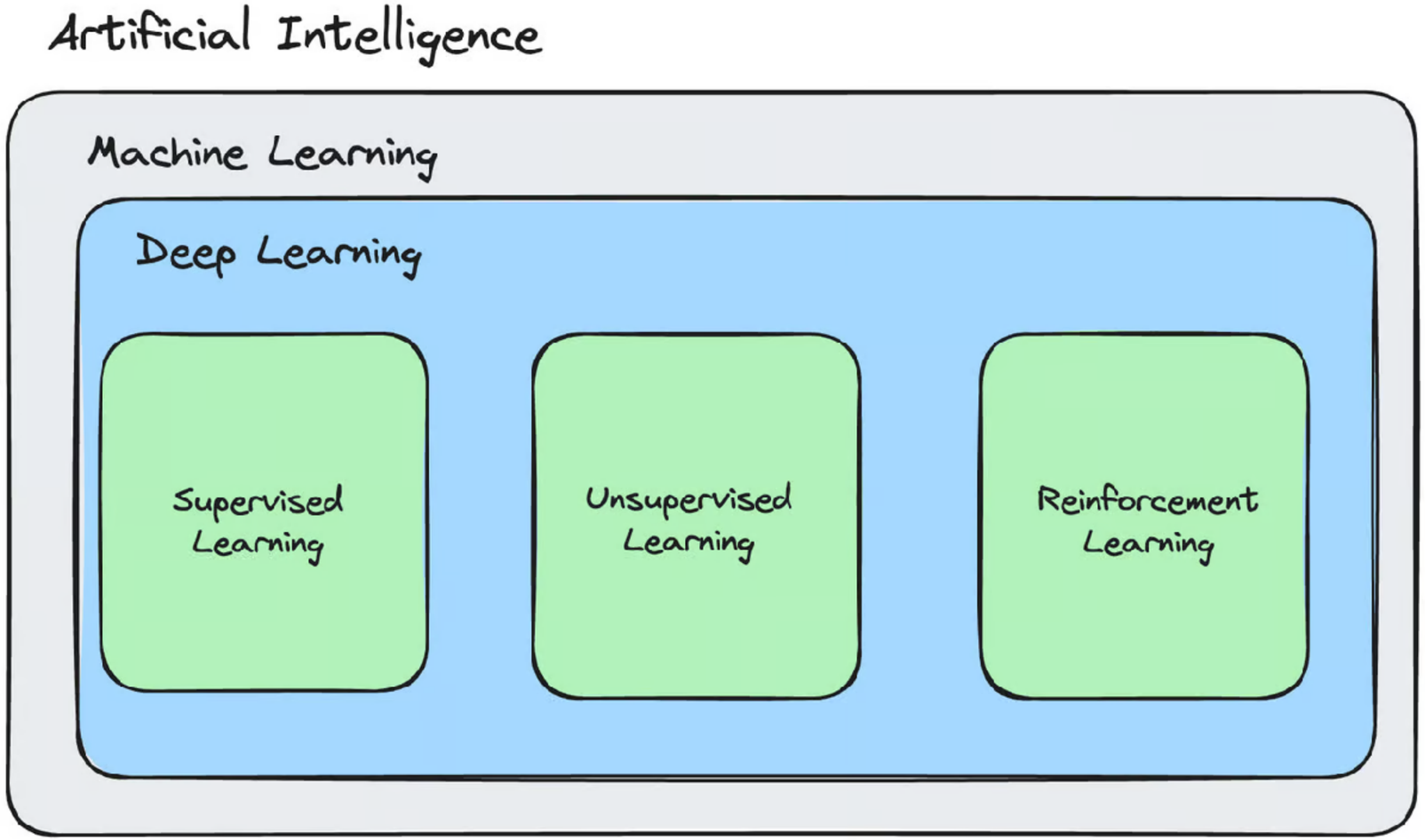
→ Dan Vega

→ Dan Vega

→ Dan Vega


DEEP LEARNING

DEEP LEARNING AND NEURAL NETWORKS



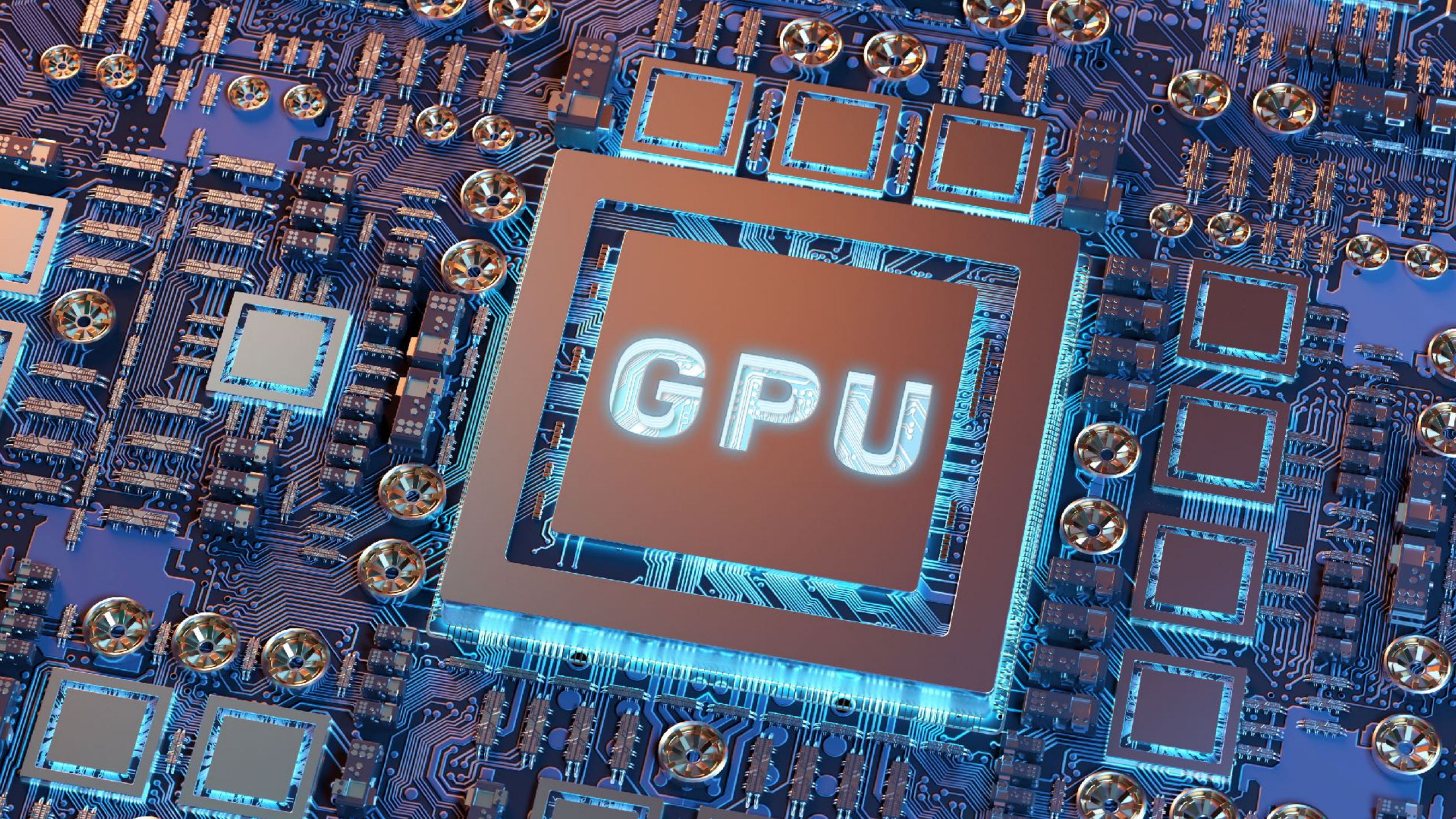
NEURAL NETWORKS

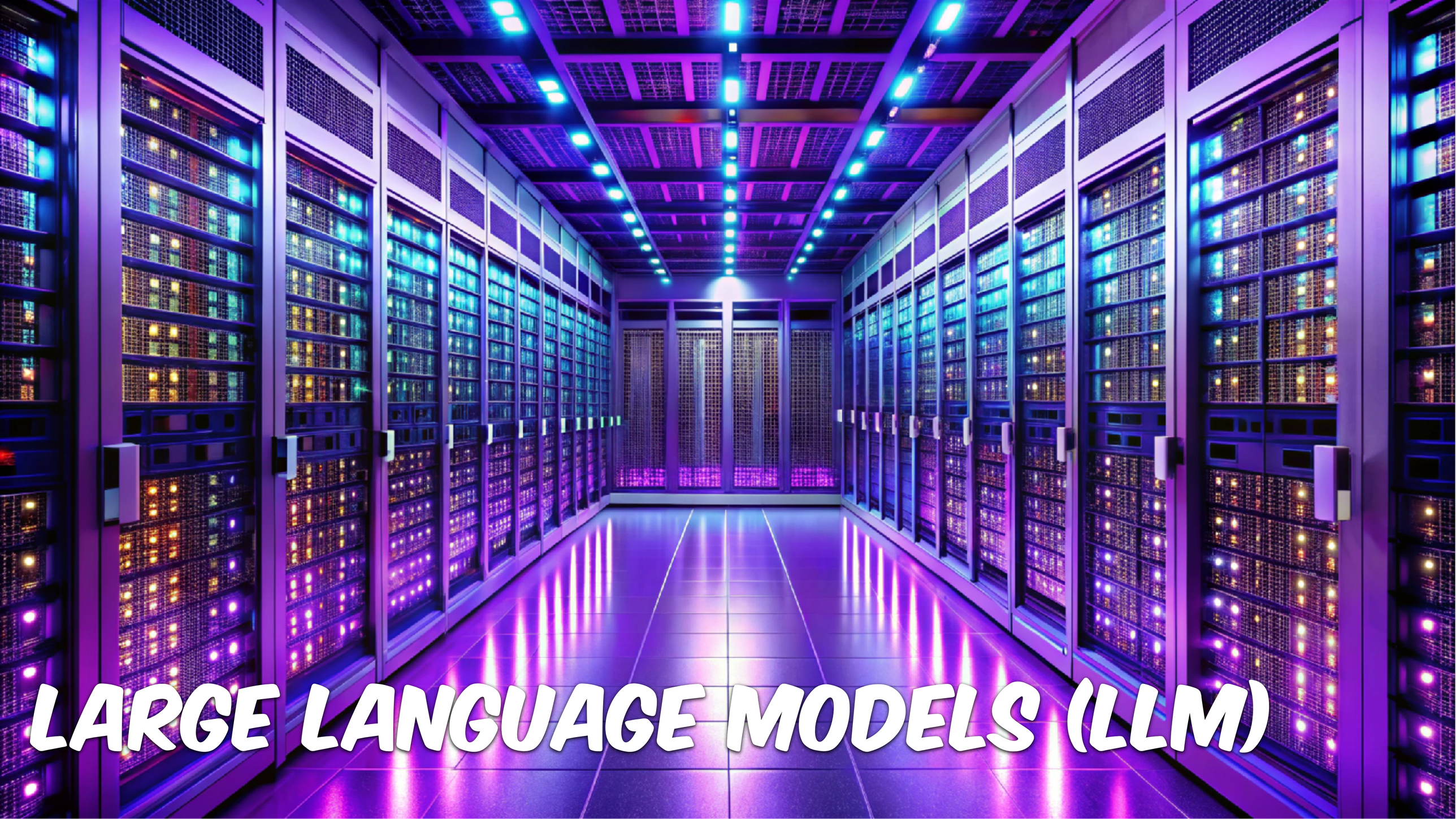
What makes deep learning powerful?

- The first layer might detect edges
- The next layer might recognize shapes
- Deeper layers could identify more complex features like eyes or wheels
- The final layer puts it all together to classify the entire image

ARTIFICIAL NEURAL NETWORK

- Scientific Advances - Deep Learning
- Availability of Big Data (You need data to configure these neural networks)
- Lots of compute power


GPU


LARGE LANGUAGE MODELS (LLM)

ATTENTION IS ALL YOU NEED

The Foundation: Attention Mechanisms

- Introduced in 2017 with the transformer, a breakthrough in AI that changed how machines process language
- The key innovation: Attention Mechanisms allow models to understand context by focusing on the relevant parts of text, similar to how humans read
- This laid the groundwork for more powerful language models

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com

Noam Shazeer[*] Google Brain noam@google.com

Niki Parmar[*] Google Research nikip@google.com

Jakob Uszkoreit[*] Google Research usz@google.com

Llion Jones[*] Google Research llion@google.com

Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com

Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

HOW TRANSFORMERS WORK

Transformer Architecture

- A specialized neural network designed for understanding language
- Uses attention mechanism to weigh the importance of words in context
- Can process entire sentences at once, understanding relationships between words

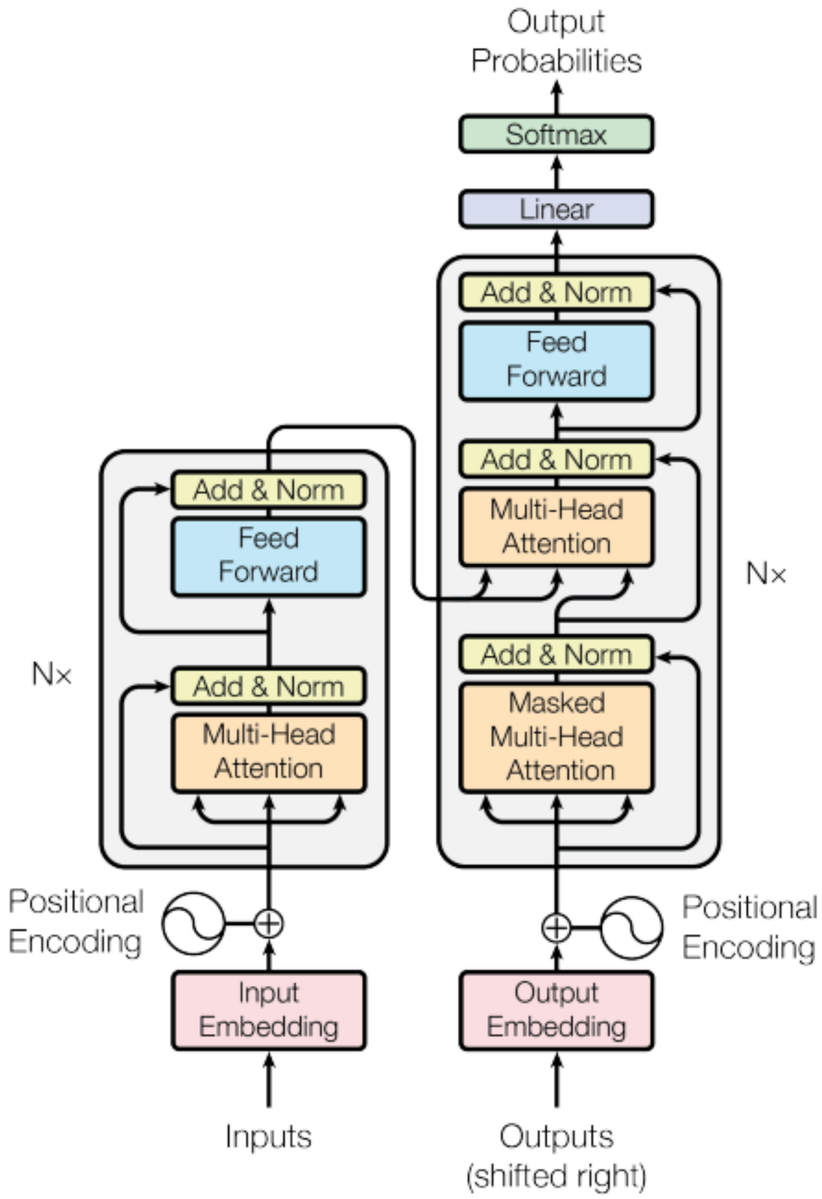
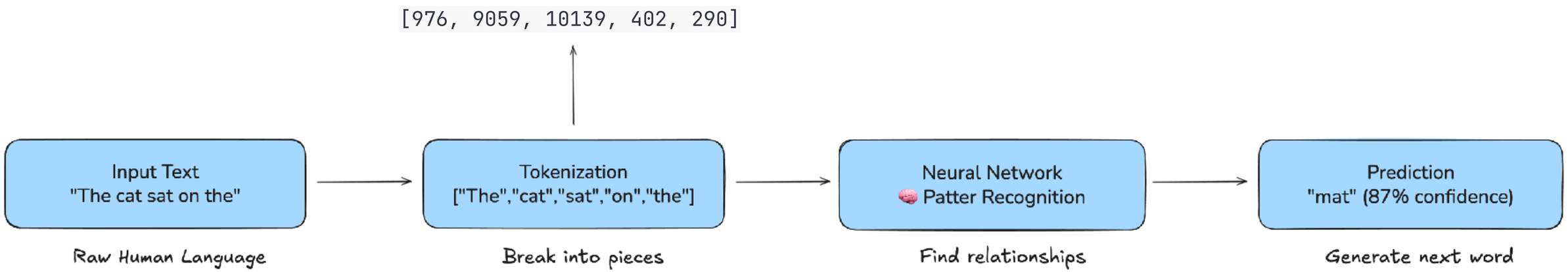


Figure 1: The Transformer - model architecture.

HOW DO LLMS ACTUALLY WORK?



Key Insight: LLMs are fundamentally *pattern matching machines* that predict the most likely next word based on patterns learned from billions of text examples.

HOW LLMS LEARN

The Training Process

1. Pre-training (Foundation)

Massive text datasets (books, web, code)
Learn general language patterns
Predict next word billions of times
Develops broad knowledge base

2. Fine-Tuning (Specialization)

Task-specific datasets
Human feedback training
Safety and alignment
Conversational abilities

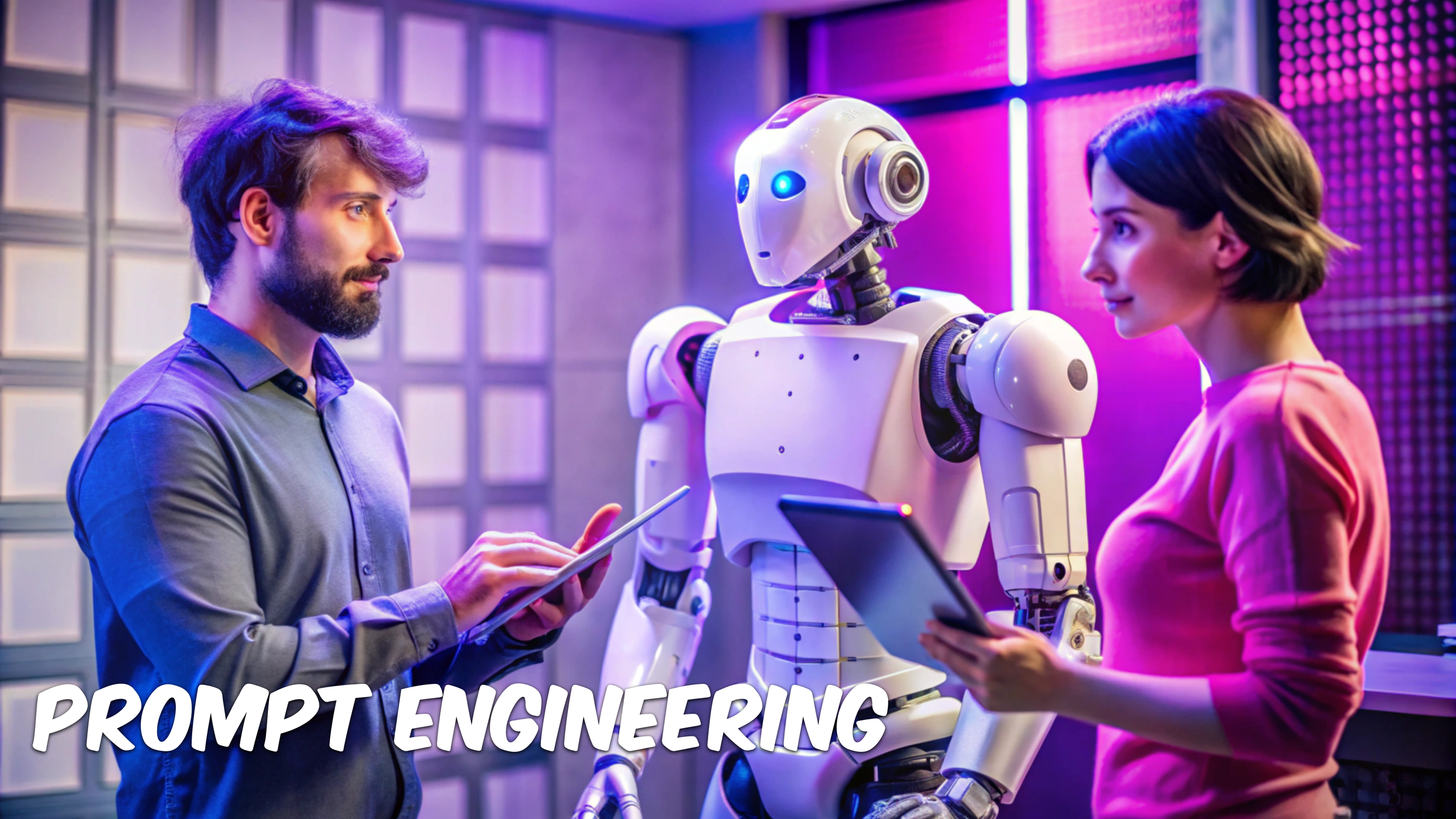
Developer Takeaway: LLMs aren't programmed with rules - they learn patterns from examples. This is why they can be creative but also unpredictable.

GENERATIVE AI

Generative Pre-trained Transformer (GPT)

- **Now that we understand:**
  - Transformers provide the architecture (how they process language)
  - Pre-training on vast amounts of data teaches language understanding
  - Generative capability allows them to create new content

- **Key Capabilities:**
  - Text generation (writing, translation, summarization)
  - Code generation and analysis
  - Complex reasoning and problem-solving


PROMPT ENGINEERING

PROMPT ENGINEERING

Learn how to effectively communicate with AI

- Clear communication is key - just like with humans
- Structure determines success - giving context, examples and specific instructions
- Think of it as teaching, not commanding
- Bad Prompt: “Write a blog post about AI”
- Good Prompt: “Write a technical blog post explaining neural networks to junior developers, focusing on practical examples.

CLEAR COMMUNICATION IS KEY

Bad Prompt:

- *Write some Java code for sorting*

Good Prompt:

Write a Java method that sorts an ArrayList of Employee objects by their salary in descending order. Include error handling for null inputs.

The 2nd prompt gives AI everything it needs

STRUCTURE DETERMINES SUCCESS

Bad Prompt:

- *Explain database indexing*

Good Prompt:

As an experienced Java developer, explain database indexing in 3 parts:

1. What indexes are and why they matter for performance

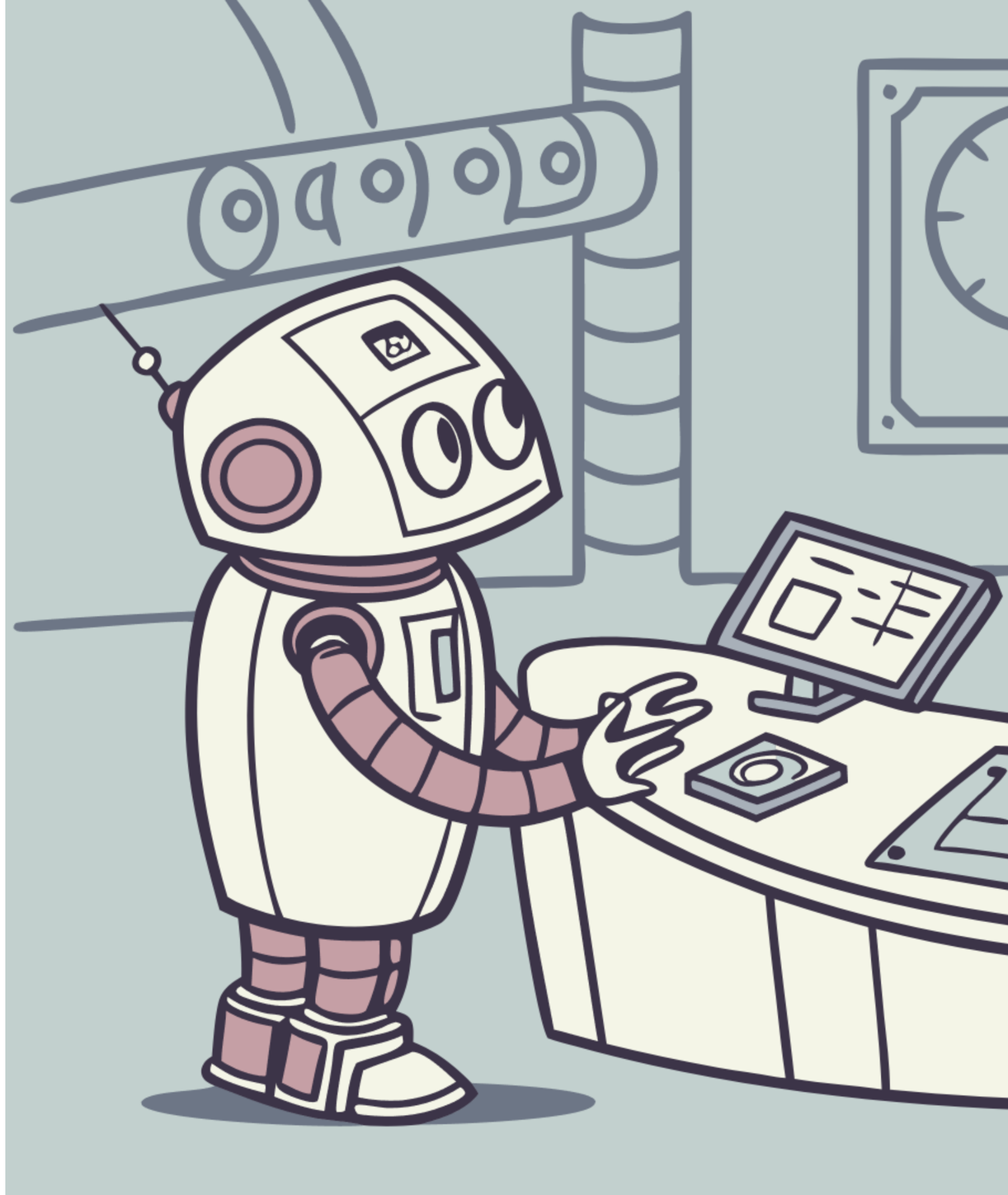
2. When to use clustered vs non-clustered indexes

3. One practical example of creating an index in PostgreSQL

Keep each section to 2-3 sentences and include one code example.

ADVANCED TECHNIQUES

- Zero-shot Prompting: Will not contain examples or demonstrations
- One-shot prompting: Providing a single example
- Few-shot prompting: Providing Examples
- Chain-of-Thought: Think step by step
- XML Tags: help structure complex requests by clearly separating different types of information
- Task Decomposition: Breaking complex problems into steps



ZERO-SHOT PROMPTING

is asking the AI to perform a task without providing any examples. There is nothing wrong with this approach and there are times when this will give you exactly what you are looking for.

> *Create a Java class that implements the Observer pattern for a stock price monitoring system.*

ONE-SHOT PROMPTING

provides a single example to establish a pattern you would like the AI to follow:

```
// Here's an example of the coding style I prefer:
public Optional<User> findUserByEmail(String email) {
    if (email == null || email.trim().isEmpty()) {
        return Optional.empty();
    }
    return userRepository.findByEmail(email.toLowerCase());
}
```

FEW-SHOT PROMPTING

provides multiple examples to guide the AI’s response.

```
Classify the sentiment of these customer reviews:
Example 1: "The software is intuitive and saves me hours of work" → Positive
Example 2: "Great documentation and excellent support team" → Positive
Example 3: "Buggy interface and crashes frequently" → Negative

Now classify: "The new features are helpful but the UI is confusing"
```

CHAIN-OF-THOUGHT PROMPTING

Bad Prompt:

- *Optimize this code. [paste your code here]*

Good Prompt:

I need to optimize this Java method that searches through a large dataset. First, help me identify the current time complexity. Then suggest specific improvements. Finally, show me the refactored code with comments explaining the performance gains.

[paste your code here]

XML TAGS

help structure complex requests by clearly separating different types of information.

```
<task>
    Create a RESTful API endpoint for user management
</task>

<requirements>
    - POST /users for creating users
    - Include validation for email and password
    - Return appropriate HTTP status codes
    - Use Spring Boot annotations
</requirements>

<constraints>
    - Java 17
    - Spring Boot 3.0
    - No external dependencies beyond Spring starter
</constraints>
```

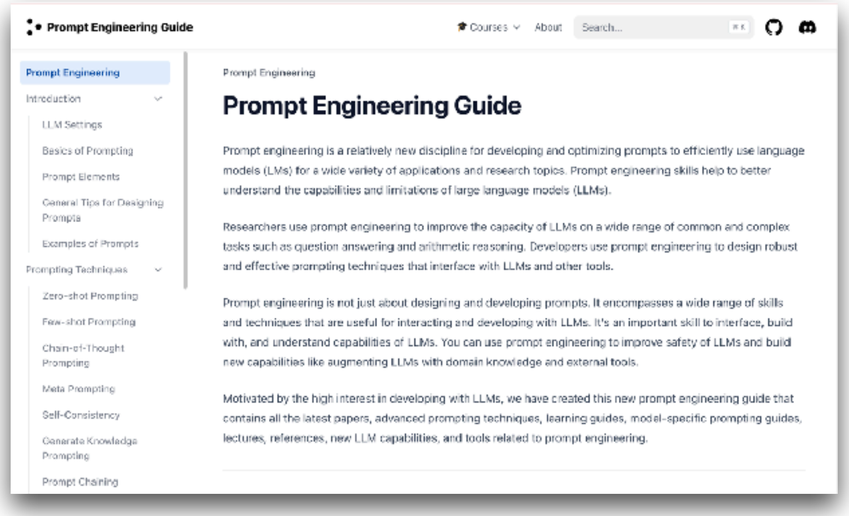
TASK DECOMPOSITION

breaks down complex problems into manageable pieces.

> *I'm building a file processing system in Java. Help me break this down:*
>
> *Phase 1: Design the file reading strategy (stream vs batch)*
>
> *Phase 2: Create the data validation logic*
>
> *Phase 3: Implement error handling and logging*
>
> *Phase 4: Add unit tests*
>
> *Start with Phase 1 - analyze the pros and cons of each approach for processing 1GB+ files.*

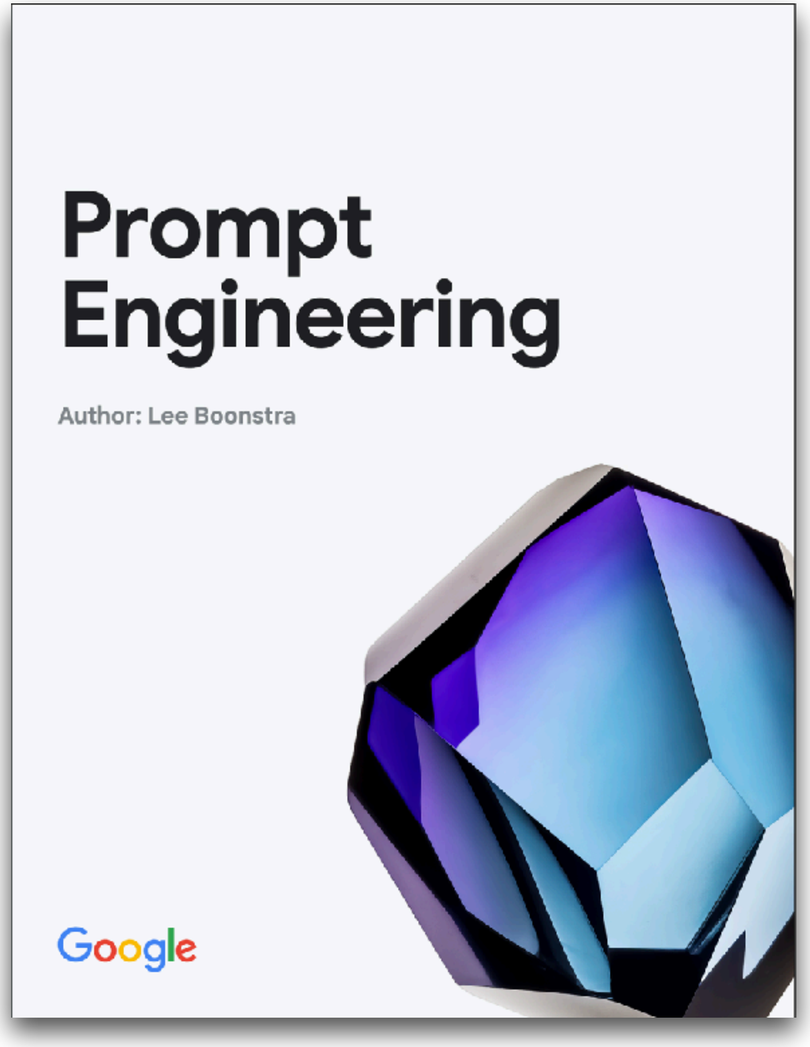
PRACTICAL TIPS FOR IMMEDIATE IMPROVEMENT

- Be Specific About Context: Always provide relevant background information. If you're working on a Spring Boot application, mention it. If you're dealing with legacy code, say so.
- Use Examples: Show the AI what good output looks like. If you want code formatted a certain way, provide an example.
- Give Context About Your Environment: Mention your Java version, frameworks you're using, or constraints you're working within.
- Specify a Role When Applicable: Give the AI a specific persona or expertise to embody. For example, "Act as a senior Java architect reviewing this code" or "As a performance optimization expert, analyze this query." This helps frame the response with the appropriate level of detail and perspective.
- Iterate and Refine: Don't expect perfection on the first try. Use the AI's response to refine your prompt and get closer to your desired output.
- Use AI to Improve Your Prompts: When you're not getting the results you want, ask the AI to help you craft a better prompt. Try something like: "I'm trying to get you to help me debug this performance issue, but your response wasn't quite what I needed. Can you suggest how I should rephrase my request to get more specific debugging steps?"
- Save Good Prompts: When you craft a prompt that works well, save it. You'll likely need similar requests in the future.



https://www.promptingguide.ai/

https://cookbook.openai.com/examples/gpt4-1_prompting_guide

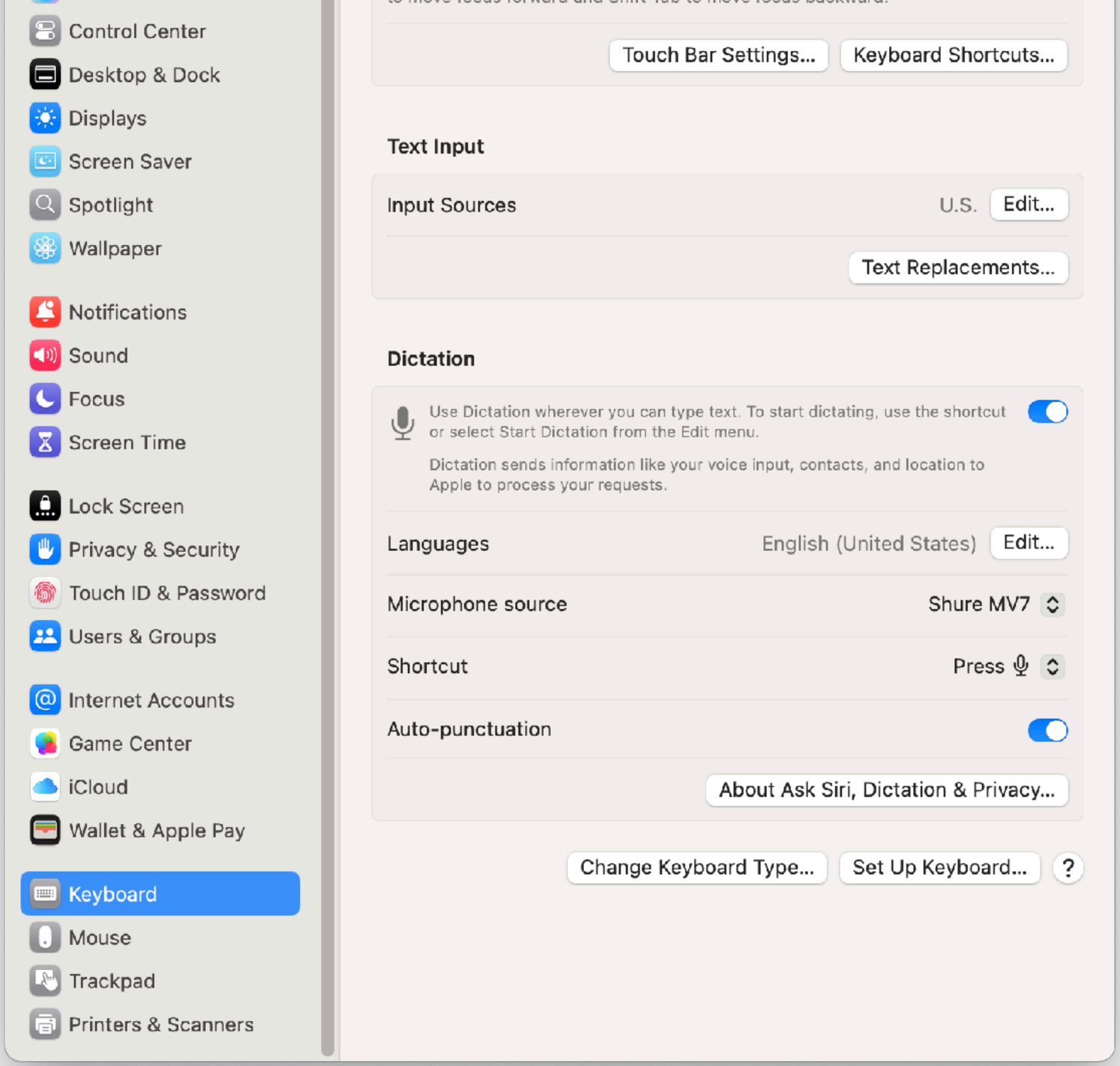


http://bit.ly/3ZVD7T5


PROMPTING WITH YOUR VOICE

macOS Dictation

Settings > Keyboard > Dictation



Flow

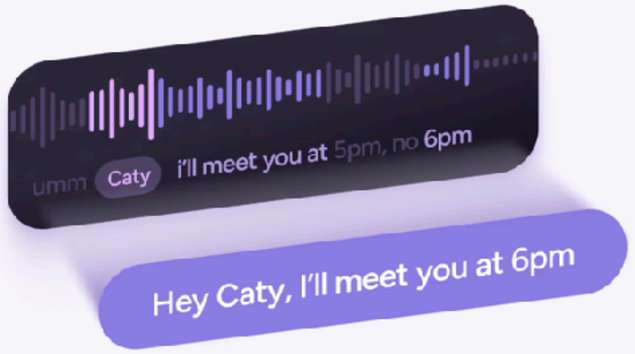
Use Cases Pricing Company Careers Download for free

Think it, speak it, send it

Effortless voice dictation in every application:
3x faster than typing, AI commands and auto-edits.

Click here to try Flow now...

Download for Mac Download for Windows



FEATURED ON Product Hunt ▲ 836

Product of the week 1st

Trusted by professionals at

substack amazon perplexity SUPERHUMAN STRAVA

there's been
nobody really knows what's going on so can you check in with

Grammar corrected

ink they're going to handle the first part of the project,

Don't type, just speak

**Effortless voice dictation in every application:
4x faster than typing, AI commands and auto-edits.**

Try Flow | Download

Available on Mac, Windows and iPhone


JAVA & AI

JAVA & AI

Leveraging Artificial Intelligence in Java Applications

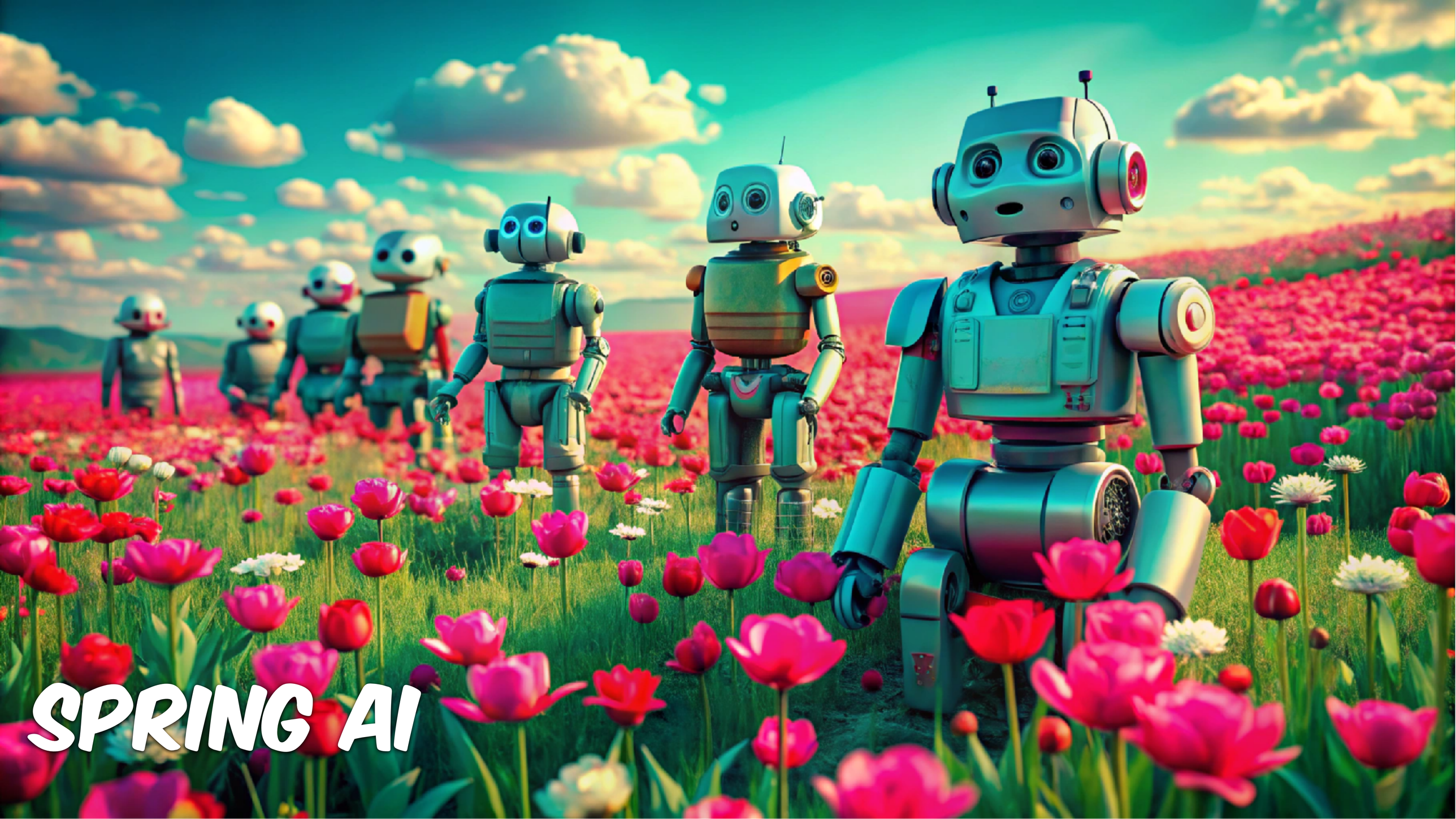
- Why AI + Java?
  - The world of software is experiencing widespread adoption of Artificial Intelligence
  - Java is the language of enterprises, creating Java + AI apps is a new requirement
- Spring AI
  - Provides the necessary API access and components for developer AI applications
  - Abstraction similar to Spring Data
- Use Cases
  - Q&A Over docs
  - Documentation Summarization
  - Text, Code, Image, Audio and Video Generation

```
#!/bin/bash
echo "Calling Open AI..."
MY_OPENAI_KEY=“YOUR_API_KEY_HERE”
PROMPT=“Tell me an interesting fact about Java"

curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $MY_OPENAI_KEY" \
  -d '{"model": "gpt-4o", "messages": [{"role":"user", "content": "'"${PROMPT}"'"}] }'
```

```
{
  "id": "chatcmpl-ABNbjZ5oRbo72OevnCX2arPufJCYK",
  "object": "chat.completion",
  "created": 1727275719,
  "model": "gpt-4o-2024-05-13",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Sure! Did you know that Java was initially designed with interactive television in mind? James Gosling and his team at Sun Microsystems started the project in 1991 under the name \"Oak.\" The name was later changed to \"Java\" after discovering there was already a programming language called Oak. Java's versatility has made it one of the most popular programming languages for a wide range of applications, far beyond its initial intended use for TV set-top boxes!",
        "refusal": null
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 14,
    "completion_tokens": 90,
    "total_tokens": 104,
    "completion_tokens_details": {
      "reasoning_tokens": 0
    }
  },
  "system_fingerprint": "fp_e375328146"
}
```

```java
public static void main(String[] args) throws IOException, InterruptedException {
    var apiKey = "YOUR_API_KEY_HERE";
    var body = """
            {
                "model": "gpt-4o",
                "messages": [
                    {
                        "role": "user",
                        "content": "Tell me an interesting fact about Java"
                    }
                ]
            }""";

    HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create("https://api.openai.com/v1/chat/completions"))
            .header("Content-Type", "application/json")
            .header("Authorization", "Bearer " + apiKey)
            .POST(HttpRequest.BodyPublishers.ofString(body))
            .build();

    var client = HttpClient.newHttpClient();
    var response = client.send(request, HttpResponse.BodyHandlers.ofString());
    System.out.println(response.body());
}
```

SPRING AI PROVIDES US SO MUCH MORE THAN A FACILITY FOR MAKING REST API CALLS


SPRING AI

SPRING AI

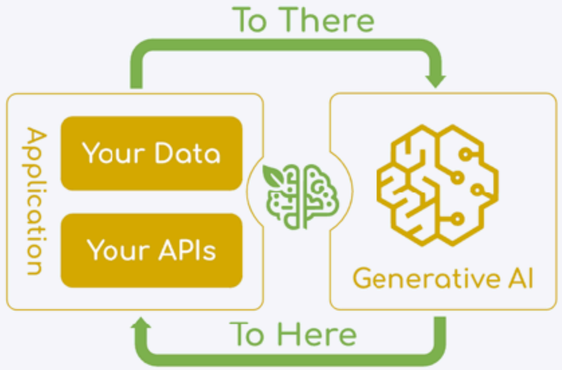
AI for Spring Developers

- https://spring.io/projects/spring-ai
  - Dr. Mark Pollack
  - Current Version 1.0.0
- Portable API support across AI providers for Chat, Image & Audio
- Synchronous & Streaming API options
- Inspired by Python Projects
  - LangChain
  - LlamaIndex

Spring AI

OVERVIEW LEARN

Spring AI is an application framework for AI engineering. Its goal is to apply to the AI domain Spring ecosystem design principles such as portability and modular design and promote using POJOs as the building blocks of an application to the AI domain.



At its core, Spring AI addresses the fundamental challenge of AI integration: Connecting your enterprise **Data** and **APIs** with the **AI Models**.

Features

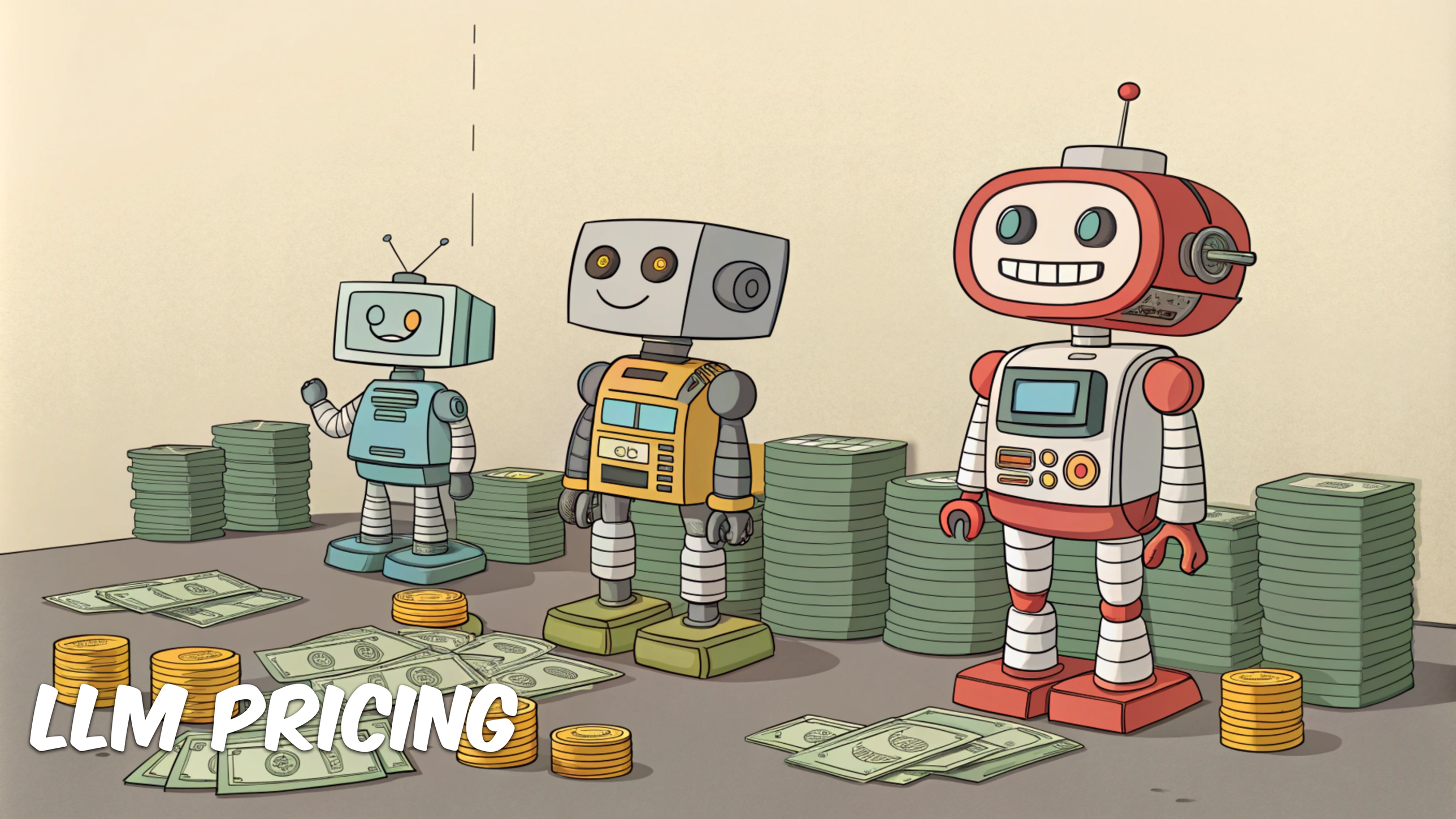
Spring AI provides the following features:

- Support for all major AI Model providers such as Anthropic, OpenAI, Microsoft, Amazon, Google, and Ollama. Supported model types include:
  - Chat Completion
  - Embedding
  - Text to Image
  - Audio Transcription
  - Text to Speech

GETTING STARTED

GETTING STARTED

- Signing up for an API Key - OpenAI
  - Signing up for multiple keys if you want to run multiple modals
  - Setting an environment variable
- start.spring.io
- Chat Client & Chat Model
  - Blocking vs Non Blocking (Streaming Responses)
  - Response Types
  - Using Multiple Modals in the same application


LLM PRICING

Free

Explore how AI can help with everyday tasks

- ✓ Access to GPT-4.1 mini
- ✓ Real-time data from the web with search
- ✓ Limited access to GPT-4o, OpenAI o4-mini, and deep research
- ✓ Limited access to file uploads, data analysis, image generation, and voice mode
- ✓ Code edits with the ChatGPT desktop app for macOS
- ✓ Use custom GPTs

Have an existing plan? See billing help

$0 / month

Get Free ↗

Plus

Level up productivity and creativity with expanded access

- ✓ Everything in Free
- ✓ Extended limits on messaging, file uploads, data analysis, and image generation
- ✓ Standard and advanced voice mode with video and screensharing
- ✓ Access to deep research and multiple reasoning models (OpenAI o3, OpenAI o4-mini, and OpenAI o4-mini-high)
- ✓ Access to a research preview of GPT-4.5, our largest model yet, and GPT-4.1, a model optimized for coding tasks
- ✓ Create and use projects, tasks, and custom GPTs
- ✓ Opportunities to test new features

$20 / month

Get Plus ↗ Limits apply >

Pro

Get the best of OpenAI with the highest level of access

- ✓ Everything in Plus
- ✓ Unlimited access to all reasoning models and GPT-4o
- ✓ Unlimited access to advanced voice, with higher limits for video and screensharing
- ✓ Access to OpenAI o1 pro mode, which uses more compute for the best answers to the hardest questions
- ✓ Extended access to deep research
- ✓ Extended access to Sora video generation
- ✓ Access to a research preview of Operator
- ✓ Access to research preview of Codex agent

Unlimited subject to abuse guardrails. Learn more

$200 / month

Get Pro ↗

GPT-4.1

Smartest model for complex tasks

Price

Input:
$2.00 / 1M tokens

Cached input:
$0.50 / 1M tokens

Output:
$8.00 / 1M tokens

GPT-4.1 mini

Affordable model balancing speed and intelligence

Price

Input:
$0.40 / 1M tokens

Cached input:
$0.10 / 1M tokens

Output:
$1.60 / 1M tokens

GPT-4.1 nano

Fastest, most cost-effective model for low-latency tasks

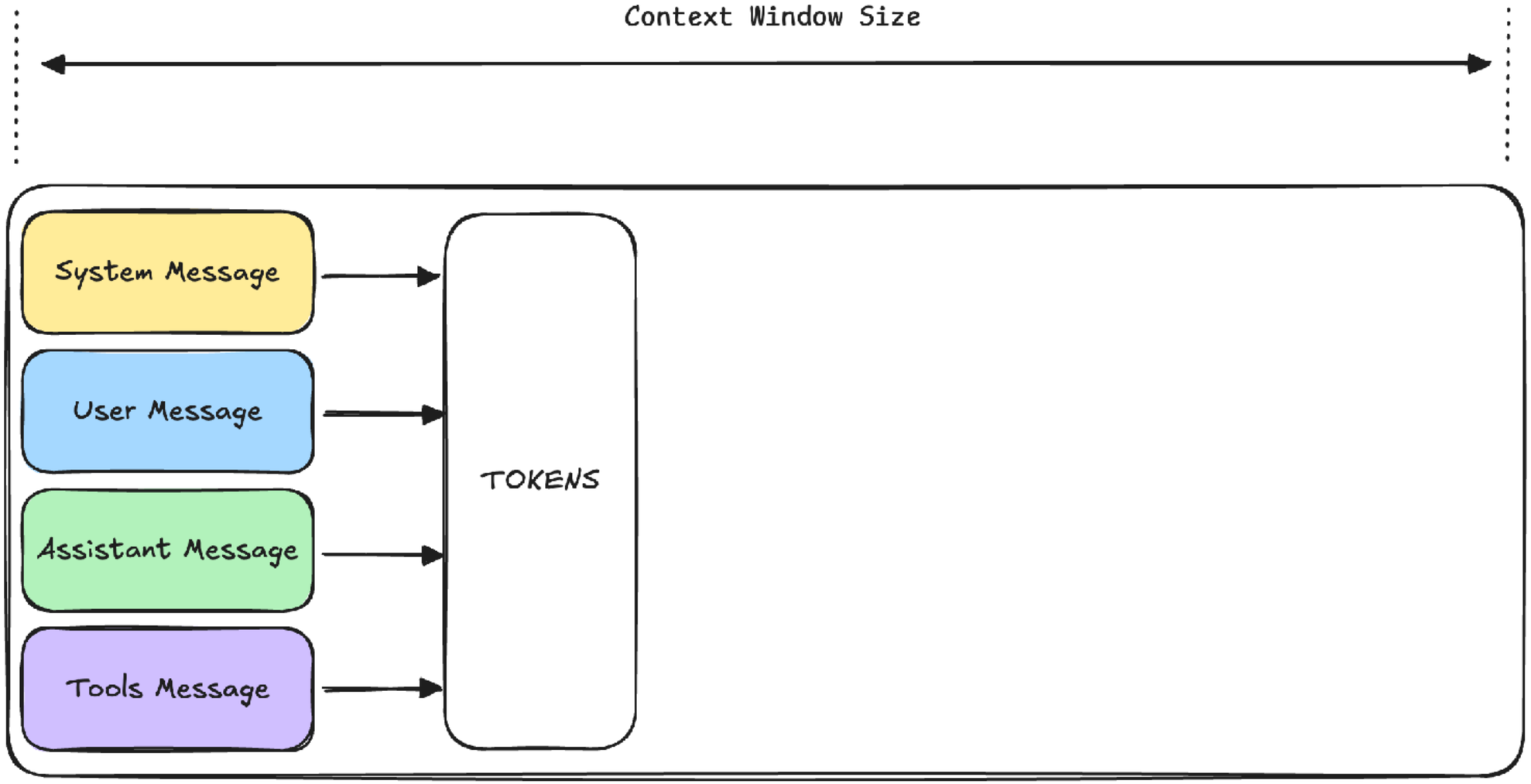
Pricing

Input:
$0.100 / 1M tokens

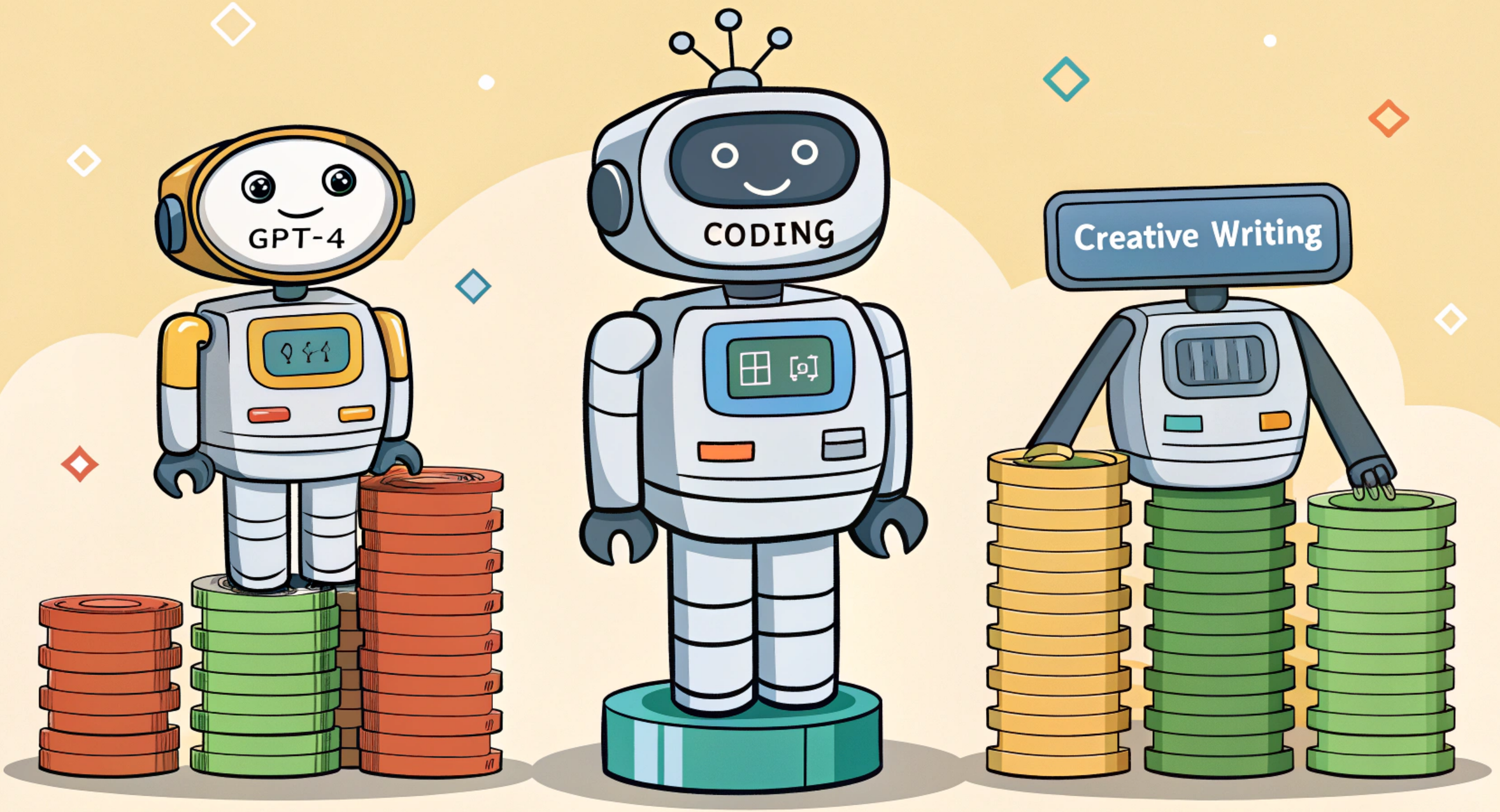
Cached input:
$0.025 / 1M tokens

Output:
$0.400 / 1M tokens

Context Window



Model	Context window	Pricing
GPT-4.1 (OpenAI)	1 M tokens	$2.00 / $8.00
GPT-4.1 Mini (OpenAI)	1 M tokens	$0.40 / $1.60
GPT-4.1 Nano (OpenAI)	1 M tokens	$0.10 / $0.40
o4-mini (OpenAI)	128 k tokens	$0.60 / $2.40
Gemini 2.5 Pro (Google)	1 M tokens (2 M rolling out)	$1.25 / $10.00 (≤ 200 k prompt)
Gemini 2.5 Flash (Google)	1 M tokens	$0.15 / $0.60 (non-thinking)
Gemini 1.5 Pro (Google)	2 M tokens	$1.25 / $5.00 (< 128 k prompt)
GPT-4.5 Preview (OpenAI)	128 k tokens	$75 / $150
Claude 3.7 Sonnet (Anthropic)	200 k tokens	$3.00 / $15.00
GPT-4o (OpenAI)	128 k tokens	$5.00 / $20.00
o3-mini (OpenAI)	200 k tokens	$1.10 / $4.40
o3-mini-high (OpenAI)	200 k tokens	$1.10 / $4.40
Gemini 2.0 Flash (Google)	1 M tokens	$0.10 / $0.40
Gemini 2.0 Flash-Lite (Google)	1 M tokens	$0.075 / $0.30
Grok 3 (xAI)	1 M tokens	$3.00 / $15.00


GPT-4
CODING
Creative Writing

Chat Model API

The Chat Model API offers developers the ability to integrate AI-powered chat completion capabilities into their applications. It leverages pre-trained language models, such as GPT (Generative Pre-trained Transformer), to generate human-like responses to user inputs in natural language.

The API typically works by sending a prompt or partial conversation to the AI model, which then generates a completion or continuation of the conversation based on its training data and understanding of natural language patterns. The completed response is then returned to the application, which can present it to the user or use it for further processing.

The `Spring AI Chat Model API` is designed to be a simple and portable interface for interacting with various AI Models, allowing developers to switch between different models with minimal code changes. This design aligns with Spring's philosophy of modularity and interchangeability.

Also with the help of companion classes like `Prompt` for input encapsulation and `ChatResponse` for output handling, the Chat Model API unifies the communication with AI Models. It manages the complexity of request preparation and response parsing, offering a direct and simplified API interaction.

You can find more about available implementations in the Available Implementations section as well as detailed comparison in the Chat Models Comparison section.

API Overview

This section provides a guide to the Spring AI Chat Model API interface and associated classes.

ChatModel

Here is the ChatModel interface definition:

```java
public interface ChatModel extends Model<Prompt, ChatResponse> {

    default String call(String message) {...}

    @Override
    ChatResponse call(Prompt prompt);
}
```

The `call()` method with a `String` parameter simplifies initial use, avoiding the complexities of the more sophisticated `Prompt` and `ChatResponse` classes. In real-world applications, it is more common to use the `call()` method that takes a `Prompt` instance and returns a `ChatResponse`.

Edit this Page
GitHub Project
Stack Overflow

Latest Models:

- **GPT-4.1** - The flagship multimodal model with major improvements in coding and instruction following, supporting up to 1 million tokens of context
- **GPT-4.1 mini** - Fast, affordable version performing almost as well as GPT-4.1 at 1/5th the price
- **GPT-4.1 nano** - The fastest and cheapest model for low-latency tasks
- **GPT-4.5** - Research preview of their strongest GPT model (being deprecated in July 2025)

Reasoning Models:

- **o3** - Most powerful reasoning model for logical, technical, and scientific tasks
- **o4-mini** - Most powerful small reasoning model, almost as powerful as o3 at 1/10th the price
- **o3-mini** (2025-01-31) - Latest reasoning model with enhanced reasoning abilities

Specialized Models:

- **GPT-image-1** (2025-04-15) - Latest image generation model with major improvements over DALL-E
- **gpt-4o-audio-preview** - Model for audio generation and voice-based interactions
- **Sora** - Video generation model for creating videos from text prompts

Latest Models:

- **Gemini 2.5 Pro** - Most intelligent AI model with "thinking" capabilities and enhanced reasoning
- **Gemini 2.5 Flash** - Efficient workhorse model designed for speed and low cost, improved across reasoning, multimodality, and code
- **Gemini 2.5 Pro Deep Think** - Experimental enhanced reasoning mode for highly complex math and coding

Current Generation:

- **Gemini 2.0 Flash** - Multimodal model with native image generation and text-to-speech capabilities
- **Gemini 2.0 Pro** - Released February 5, 2025
- **Gemini 2.0 Flash Thinking Experimental** - Shows the model's thinking process when responding

Legacy Models:

- **Gemini 1.5 Pro** and **Gemini 1.5 Flash** (note: starting April 29, 2025, these are not available in new projects)

Latest Models (Claude 4 Family):

- **Claude Opus 4** - Most powerful model, described as "the best coding model in the world" capable of working autonomously for up to 7 hours
- **Claude Sonnet 4** - Smart, efficient model for everyday use, designed as a drop-in replacement for Sonnet 3.7 with improved coding and instruction following

Previous Generation:

- **Claude Sonnet 3.7** - Previous flagship model
- **Claude Haiku 3** - Fastest, most compact model

Benchmark		Gemini 2.5 Flash Preview (05-20) Thinking	Gemini 2.0 Flash	OpenAI o4-mini	Claude Sonnet 3.7 64k Ext. Thinking	Grok 3 Beta Extended thinking	DeepSeek R1
Input price	$/M tokens	$0.15	$0.10	$1.10	$3.00	$3.00	$0.55
Output price	$/M tokens	$0.60 No reasoning $3.50 Reasoning	$0.40	$4.40	$15.00	$15.00	$2.19
Reasoning & knowledge Humanity's Last Exam (no tools)		11.0%	5.1%	14.3%	8.9%	—	8.6%*
Science GPQA diamond	single attempt (pass@1)	82.8%	60.1%	81.4%	78.2%	80.2%	71.5%
	multiple attempts	—	—	—	84.8%	84.6%	—
Mathematics AIME 2025	single attempt (pass@1)	72.0%	27.5%	92.7%	49.5%	77.3%	70.0%
	multiple attempts	—	—	—	—	93.3%	—
Code generation LiveCodeBench v5	single attempt (pass@1)	63.9%	34.5%	—	—	70.6%	64.3%
	multiple attempts	—	—	—	—	79.4%	—
Code editing Aider Polyglot		61.9% / 56.7% whole / diff-fenced	22.2% whole	68.9% / 58.2% whole / diff	64.9% diff	53.3% diff	56.9% diff
Agentic coding SWE-bench Verified		60.4%	—	68.1%	70.3%	—	49.2%
Factuality SimpleQA		26.9%	29.9%	—	—	43.6%	30.1%
Factuality FACTS Grounding		85.3%	84.6%	62.1%	78.8%	74.8%	56.8%
Visual reasoning MMMU	single attempt (pass@1)	79.7%	71.7%	81.6%	75.0%	76.0%	no MM support
	multiple attempts	—	—	—	—	78.0%	no MM support
Image understanding Vibe-Eval (Reka)		65.4%	56.4%	—	—	—	no MM support
Long context MRCR v2	128k (average)	74.0%	36.0%	49.0%	—	54.0%	45.0%
	1M (pointwise)	32.0%	6.0%	—	—	—	—
Multilingual performance Global MMLU (Lite)		88.4%	83.4%	—	—	—	—

Methodology

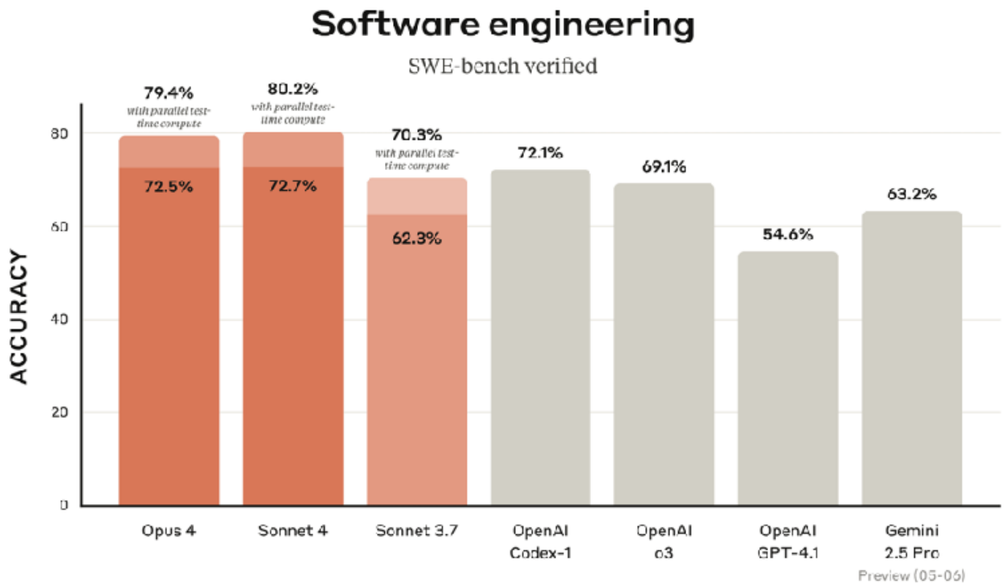
Gemini results: All Gemini scores are pass @1 (no majority voting or parallel test time compute unless indicated otherwise). They are all run with the AI Studio API for the model-id gemini-2.5-flash-preview-05-20 and gemini-2.0-flash with default sampling settings. To reduce variance, we average over multiple trials for smaller benchmarks. Vibe-Eval results are reported using Gemini as a judge.

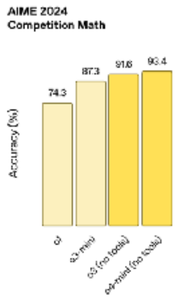
Non-Gemini results: All the results for non-Gemini models are sourced from providers' self reported numbers unless mentioned otherwise below. All SWE-bench Verified numbers follow official provider reports, using different scaffoldings and infrastructure. Google's scaffolding includes drawing multiple trajectories and re-scoring them using model's own judgement.

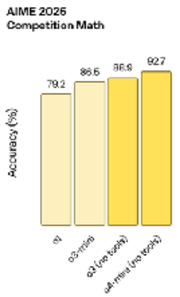
Thinking vs not-thinking: For Claude 3.7 Sonnet GPQA, AIME 2024, MMMU, come with 64k extended thinking, Aider with 32k, and HLE with 16k. Remaining results come from the non thinking model due to result availability. For Grok-3 all results come with extended reasoning except for SimpleQA (based on xAI reports) and Aider.

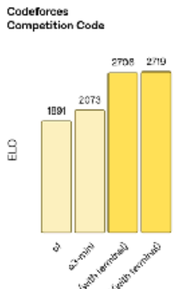
Single attempt vs multiple attempts: When two numbers are reported for the same eval higher number uses majority voting with n=64 for Grok models and internal scoring with parallel test time compute for Anthropic models.

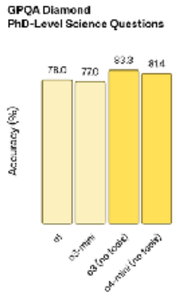
Result sources: Where provider numbers are not available we report numbers from leaderboards reporting results on these benchmarks: Humanity's Last Exam results are sourced from https://agi.safe.ai and https://scale.com/leaderboard/humanitys_last_exam, AIME 2025 numbers are sourced from https://matharena.ai/. LiveCodeBench results are from https://livecodebench.github.io/leaderboard.html (01/01/2024 - 3/1/2025 in the UI). Aider Polyglot numbers come from https://aider.chat/docs/leaderboards/. FACTS come from https://www.kaggle.com/benchmarks/google/facts-grounding. For MRCR v2 which is not publically available yet we include 128k results as a cumulative score to ensure they can be comparable with previous results and a pointwise value for 1M context window to show the

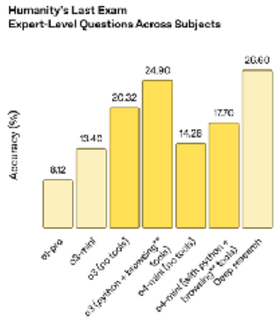












#	Criterion	Questions to Ask	Typical Signals / How to Test
1	**Task Fit**	• Does the model already solve your task (chat, code-gen, doc QA, …)? • Is there evidence on similar domains?	- Quick manual prompts - does it “sound” right?- Search papers/blogs for domain evals.- Try small-scale eval harness with 10–20 gold answers.
2	**Quality & Benchmarks**	• Are there recent third-party numbers on tasks that resemble yours? • Has the vendor cherry-picked?	Check open leaderboards (Hugging Face, LM); run LM-Eval-Harness / OpenAI Evals yourself. Benchmarks age fast, so prefer ones < 3 mo old.
3	**Context Window**	• Minimum tokens your prompt + docs require? • Do you anticipate growth (RAG, code)?	Hard cut-off → disqualify below the needed window.
4	**Latency & Throughput**	• User-facing? batch background? real-time streaming?	Measure end-to-end. Compare latency & tokens/sec.
5	**Cost & Licensing**	• Budget per request / month? • Can you cache or fine-tune cheaper model? • OSS licensce compatible with your product?	API $/1K tokens vs. GPU TCO (open-source on-prem). Total Cost of Ownership often outweighs raw $/token.
6	**Privacy / Data Residency**	• Can data leave VPC? • PII, trade secrets?	If “no cloud” → shortlist local OSS (e.g., Llama 3) or vendor’s private endpoints.
7	**Integration & Tooling**	• Supports function/tool calling, JSON mode?	Verify in docs / quick code snippet.
8	**Governance & Support**	• SLAs? rate-limit ceiling? • Security posture & audits?	Vendor contracts, community activity, GitHub issues velocity.

1

Define the Job, Not the Model

Write a one-line "job description" for the LLM. Anything that doesn't satisfy that job is out.

> e.g., "Summarise 5-page PDFs into 3 bullet points"

2

Short-list 3–5 Candidates

Filter with hard constraints

> Context ≥ need • License compatible • GPU/CPU available • Spring AI adapter exists

3

Run a 30-minute Smoke Test

Use Spring AI's **ChatClient** with 10–20 representative prompts

> Drop models that hallucinate or need extreme prompt-engineering. Saves hours on duds.

4

Automate a Narrow Eval

LM-Eval-Harness or OpenAI Evals with golden answers

> Track: Accuracy • Reasoning • Latency
> **cookbook.openai.com**

5

Weight the Metrics

Score each model 1-5, multiply by weights. Keep #2/#3 as fallback.

30% Quality **20% Cost** **20% Privacy** **30% Other**

6

Prototype & Monitor

Wire winner into Spring AI, keep alternatives registered

> Runtime switches **@Bean(name="chatModelX")**
> Log: Latency • Cost • Error rate • Ready to swap

CHECK OUT MY DEMO

SPRING AI FEATURES

SPRING AI FEATURES

- Spring AI Reference Documentation
- Prompts & Prompt Templates
- Structured Output
- Multimodal (Text, Images & Audio)
- Chat Memory

CHECK OUT MY DEMO

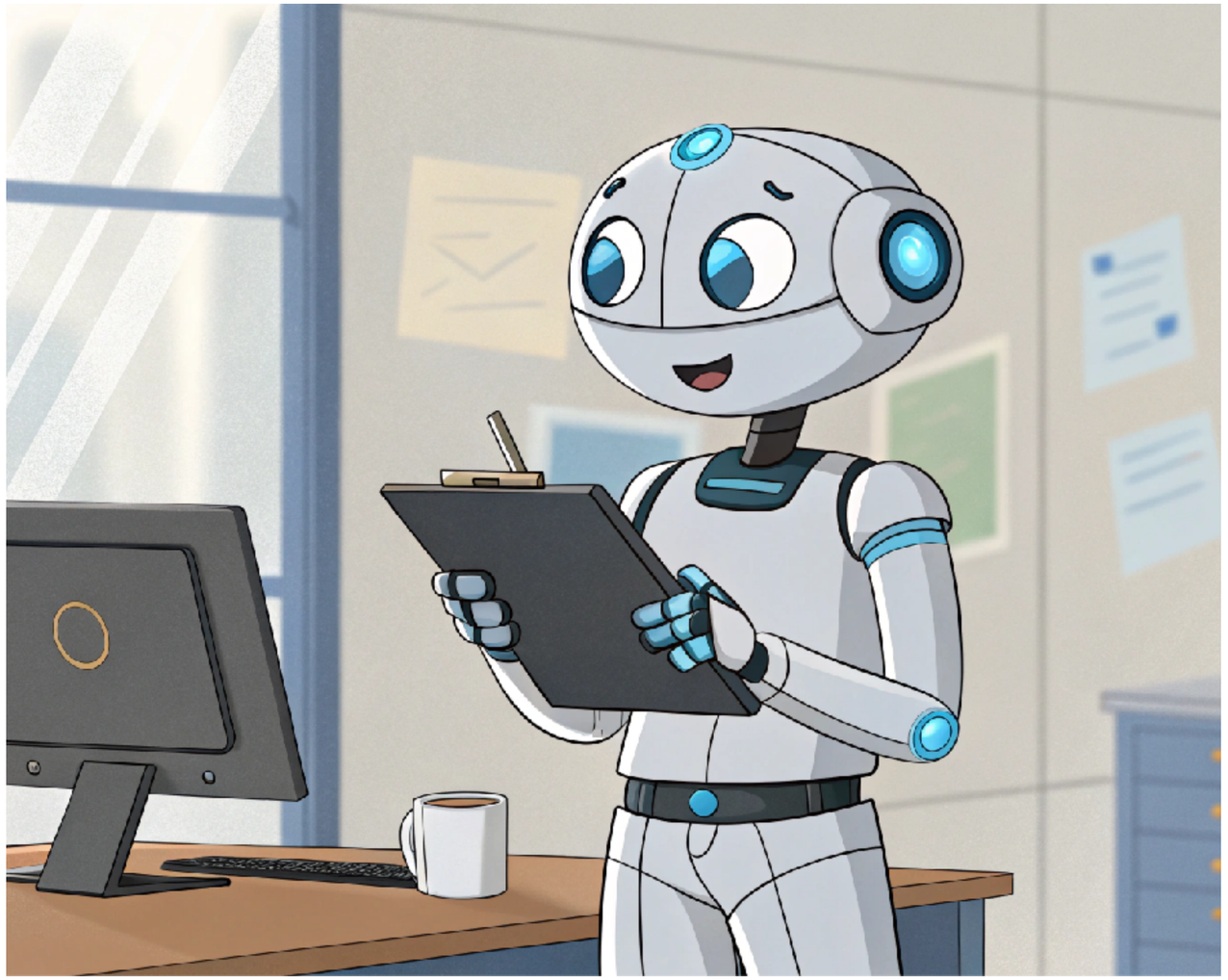
LLM LIMITATIONS

WHY DO WE CARE?

Risk of Wrong Answers

Risk of Lost Trust

Run-away Costs

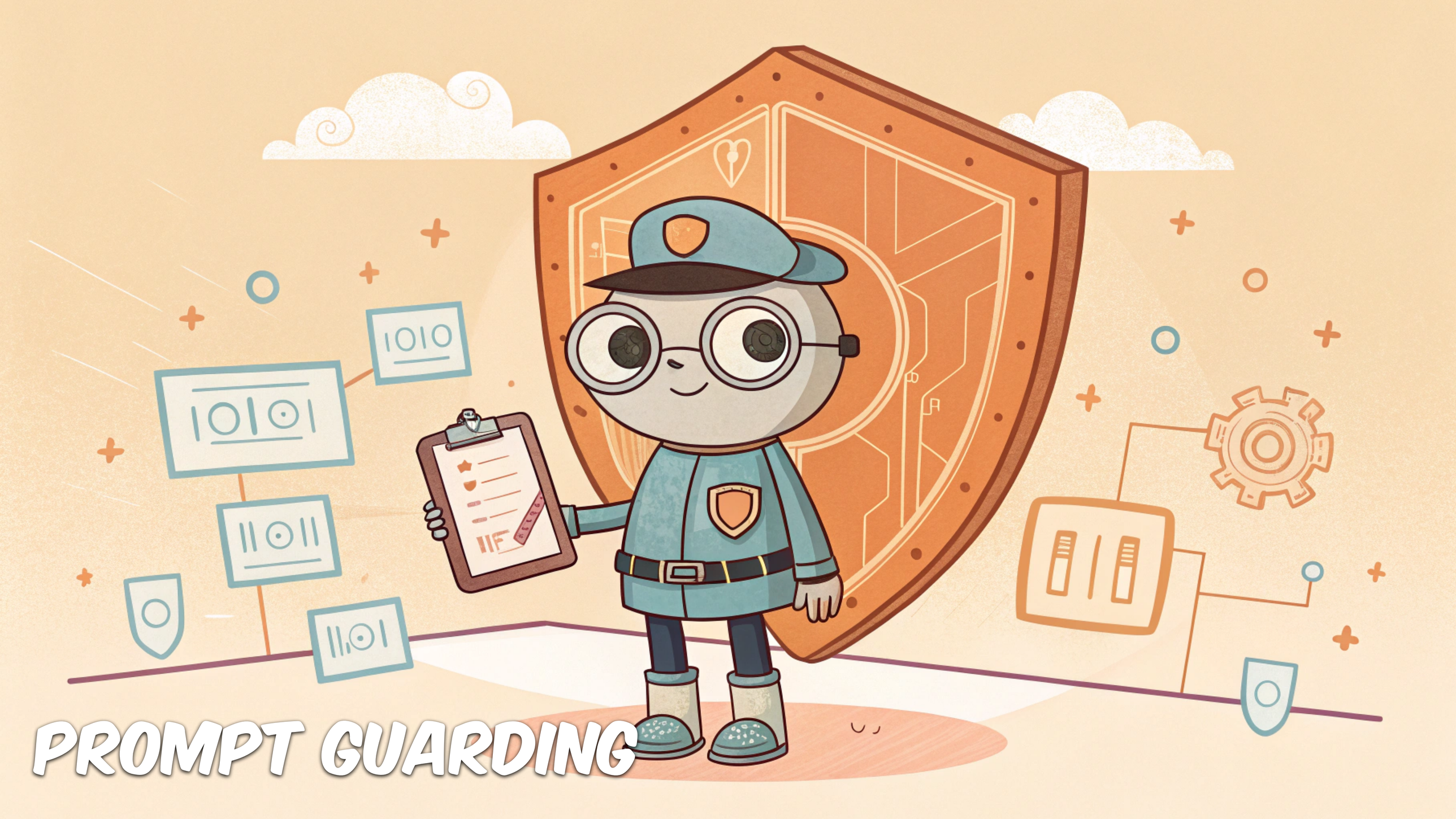


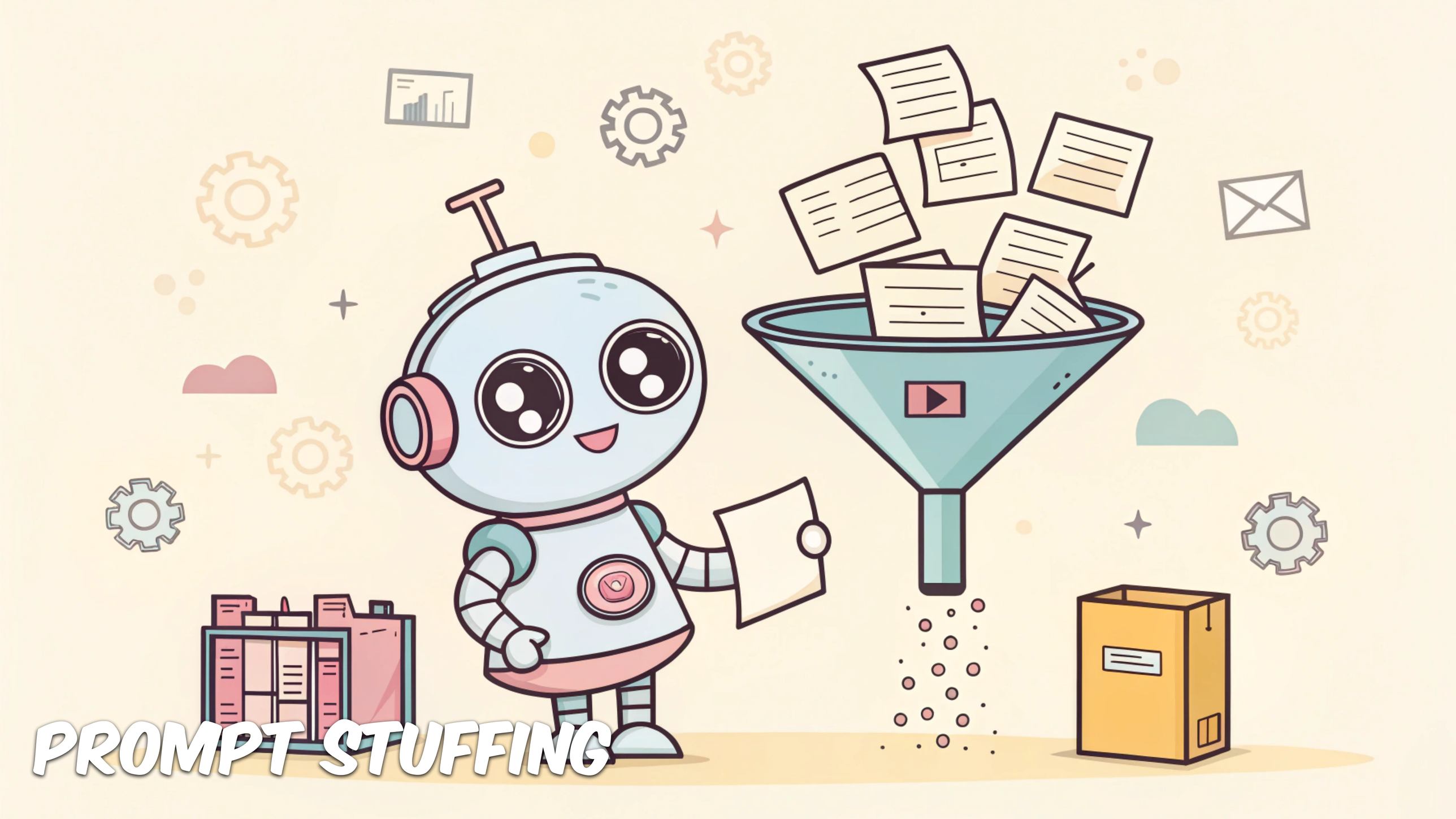
Limitation	Quick Note — “what this looks like”
🧠✨ **Hallucinate**	Invents facts or API names with total confidence
🕒 **Stale Data**	Knowledge cutoff means “today’s stock price?” → ¯_(ツ)_/¯
⚖️ **Bias & Safety**	Outputs stereotypes, toxic language, or policy violations
🏗️ **Domain Gaps**	Uses generic wording where niche jargon is required
📏 **Context Window**	Long threads get truncated; model “forgets” earlier details
🎲 **Non-Determinism**	Same prompt, different answer → flaky tests, review churn
🔒 **Privacy Leak**	Proprietary/PII text could leave your trusted boundary
💰⏱️ **Cost & Latency**	High-token chains drain budget and slow UX
🧩 **Weak Reasoning**	Multi-step calculations or logical deductions fail
🔍 **Low Explainability**	Hard to audit: “Why did you choose that answer?”

LLM LIMITATIONS

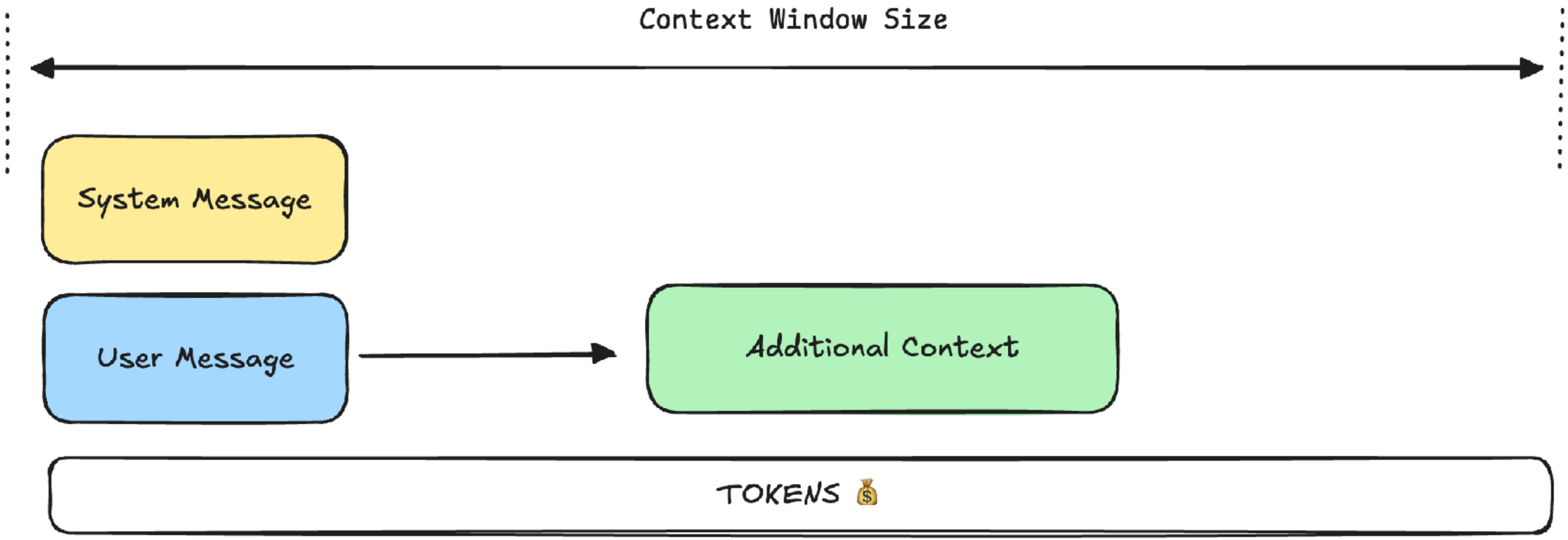
Here's our Swiss-army lineup for taming any limitation

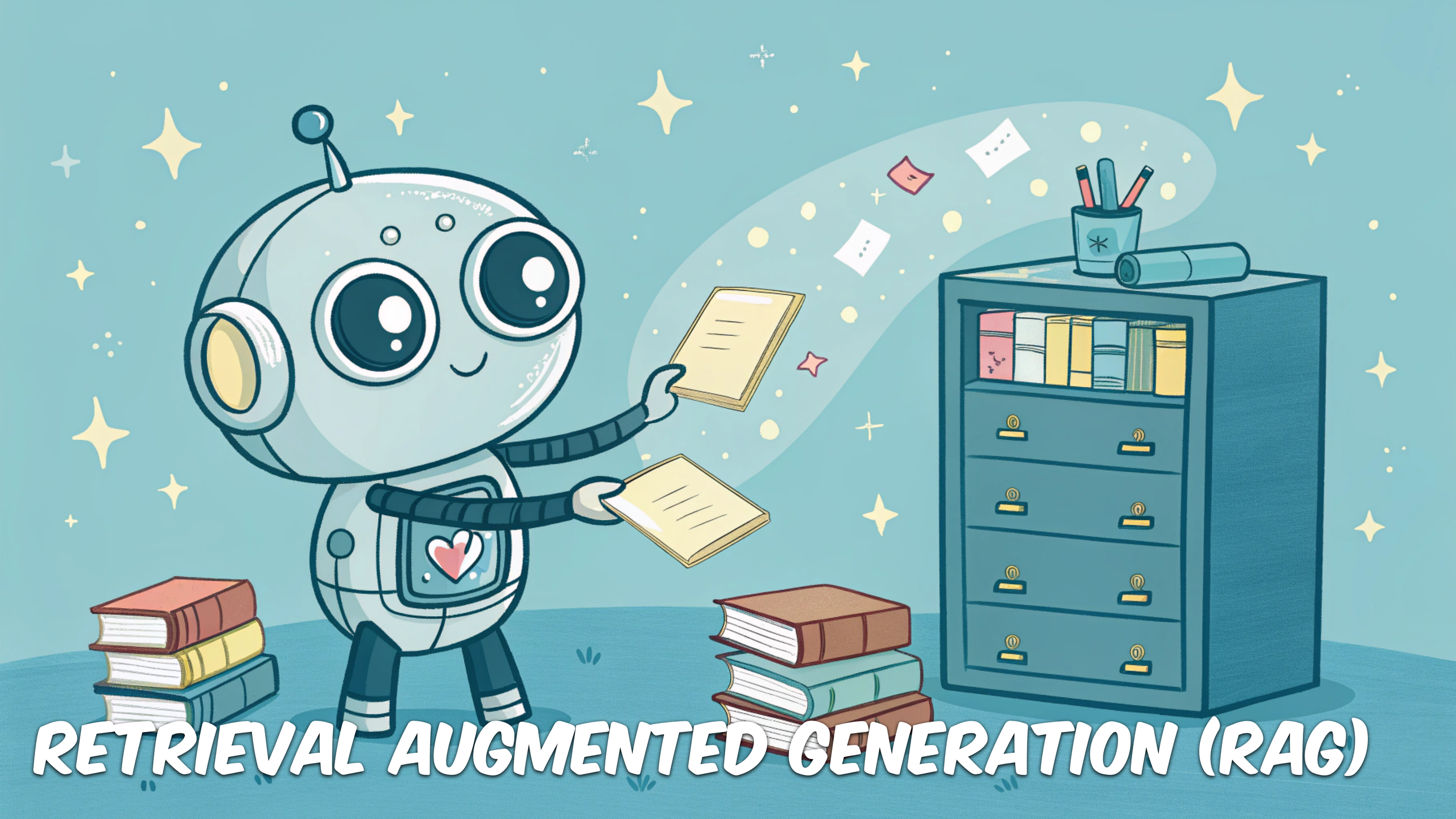
#	Lever	Purpose
1	🛡️ **Prompt Guarding**	Encode rules that constrain the model's behavior (tone, honesty, refusal policy).
2	📄 **Prompt Stuffing / RAG**	Inject fresh, task-specific context so the model quotes facts instead of guessing.
3	🔧 **Tools / Function Calling**	Let the model invoke code or APIs for real-time data, calculations, or business logic.
4	🌐 **MCP (Resources + Tools)**	Package those tools as reusable, versioned endpoints every client can share.


PROMPT GUARDING

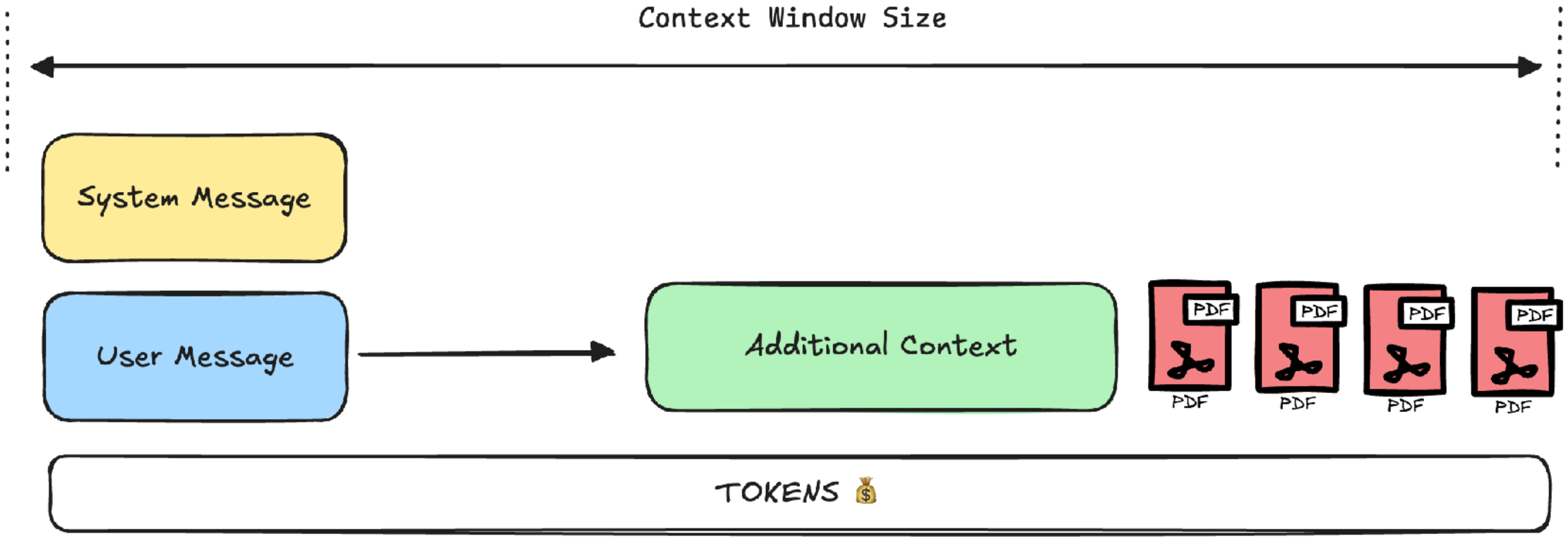

PROMPT STUFFING

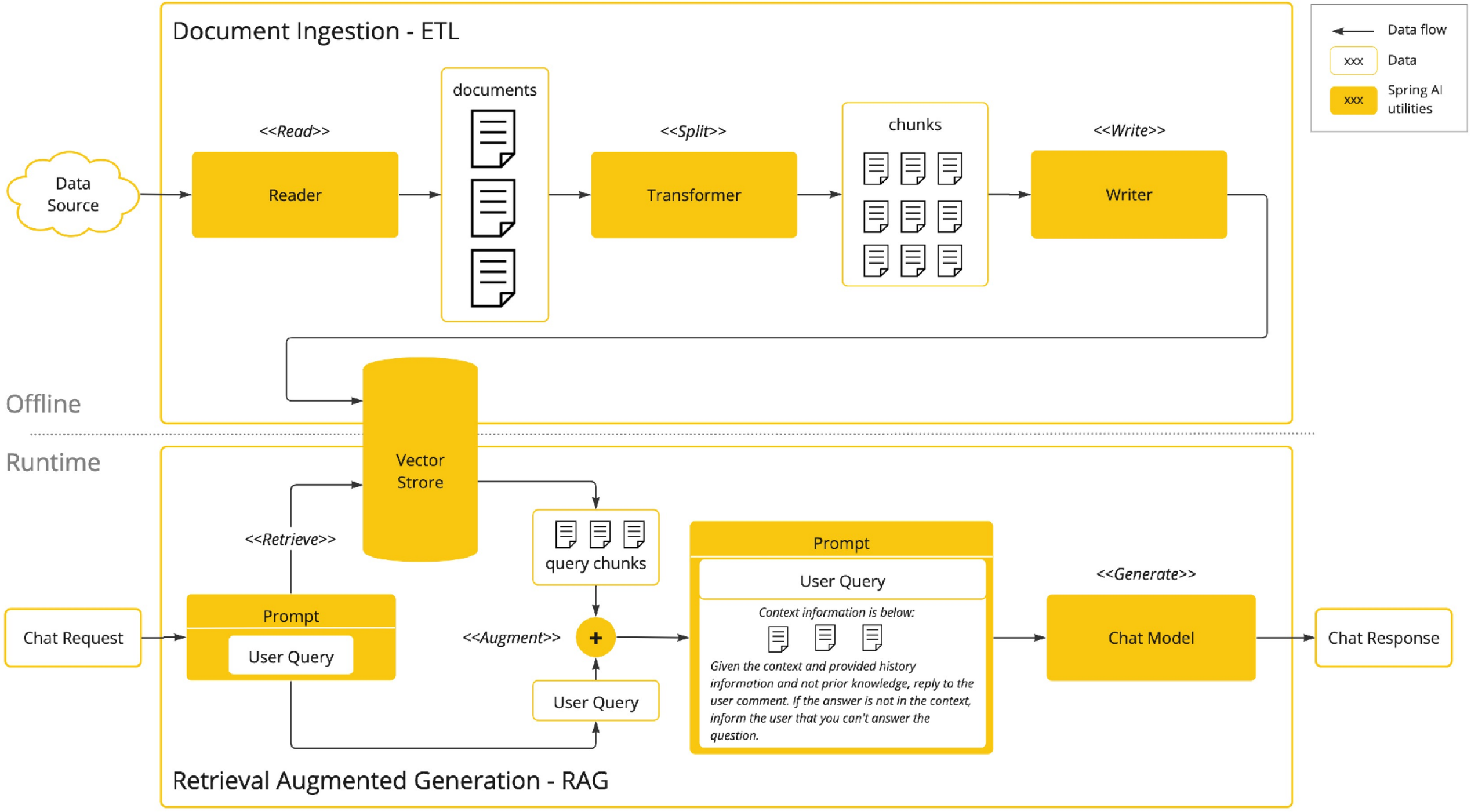
Context Window

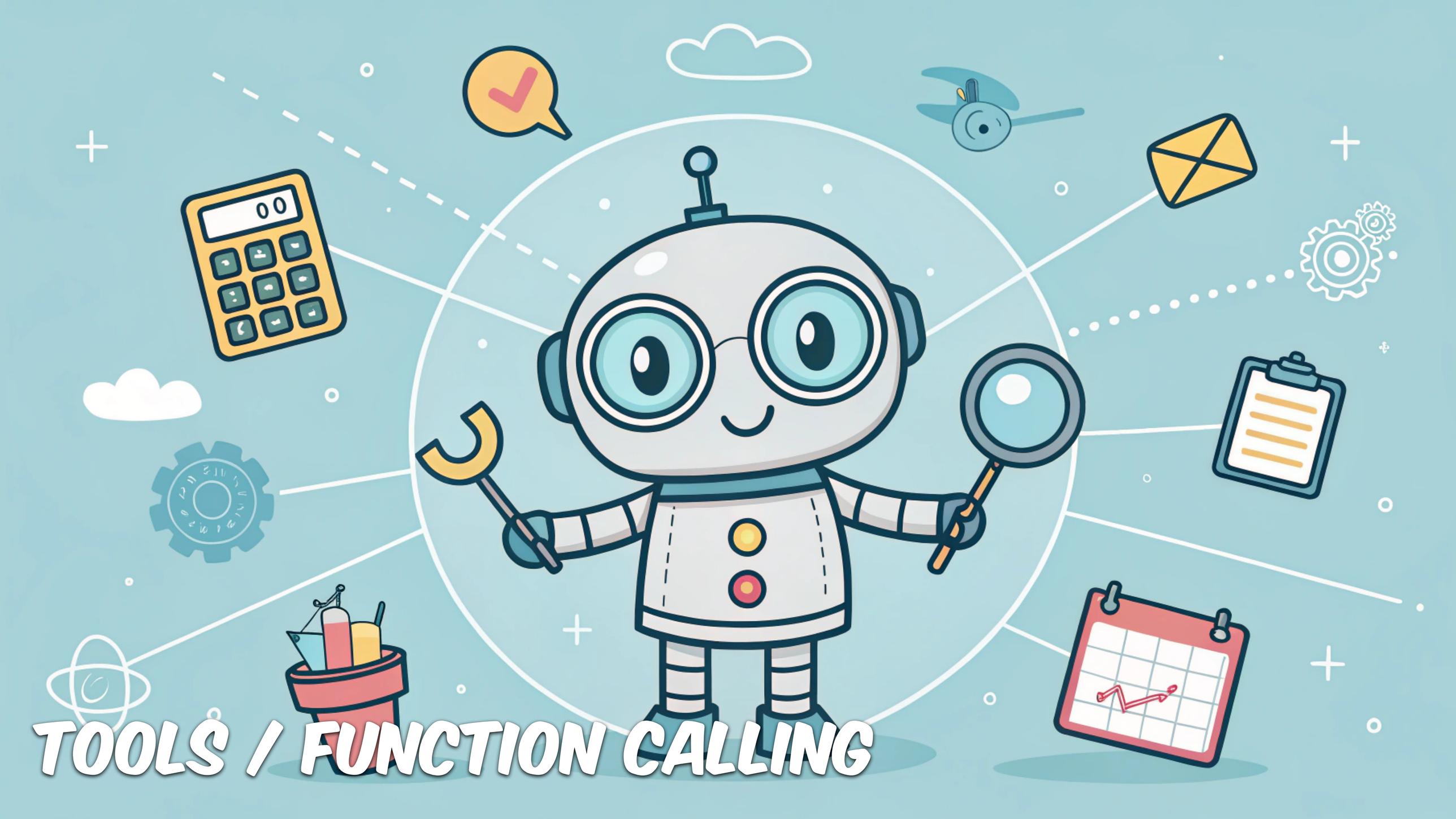



RETRIEVAL AUGMENTED GENERATION (RAG)

Context Window




Document Ingestion - ETL
Data flow
xxx
Data
xxx
Spring AI utilities
Data Source
<<Read>>
Reader
documents
<<Split>>
Transformer
chunks
<<Write>>
Writer
Offline
Runtime
Vector Strore
<<Retrieve>>
query chunks
Prompt
User Query
Chat Request
<<Augment>>
+
User Query
Prompt
User Query
Context information is below:
Given the context and provided history information and not prior knowledge, reply to the user comment. If the answer is not in the context, inform the user that you can't answer the question.
<<Generate>>
Chat Model
Chat Response
Retrieval Augmented Generation - RAG


TOOLS / FUNCTION CALLING

TOOL CALLING / FUNCTION CALLING

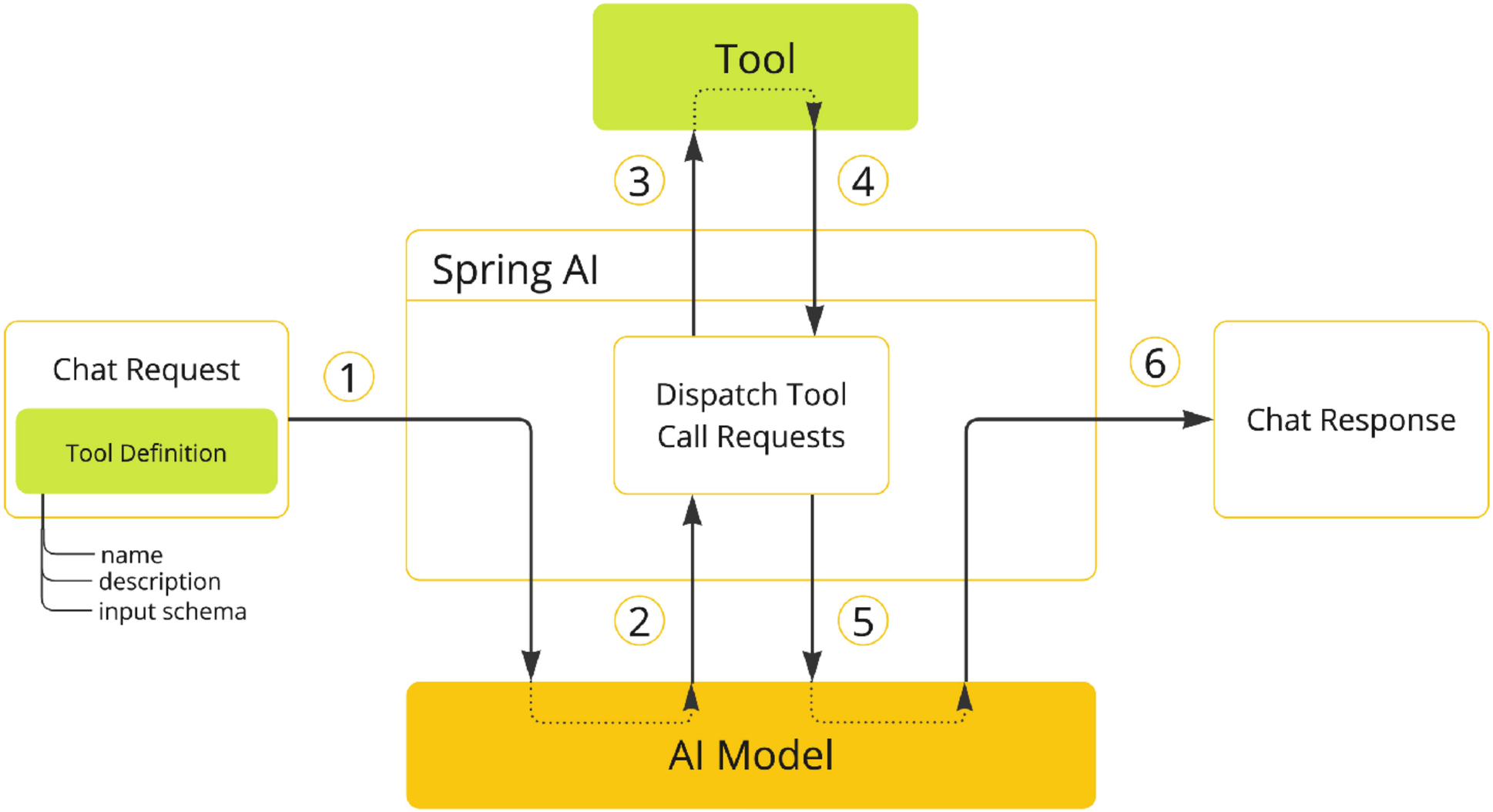
Information Retrieval

Tools in this category can be used to retrieve information from external sources, such as a database, a web service, a file system, or a web search engine. The goal is to augment the knowledge of the model, allowing it to answer questions that it would not be able to answer otherwise. As such, they can be used in Retrieval Augmented Generation (RAG) scenarios. For example, a tool can be used to retrieve the current weather for a given location, to retrieve the latest news articles, or to query a database for a specific record.

TOOL CALLING / FUNCTION CALLING

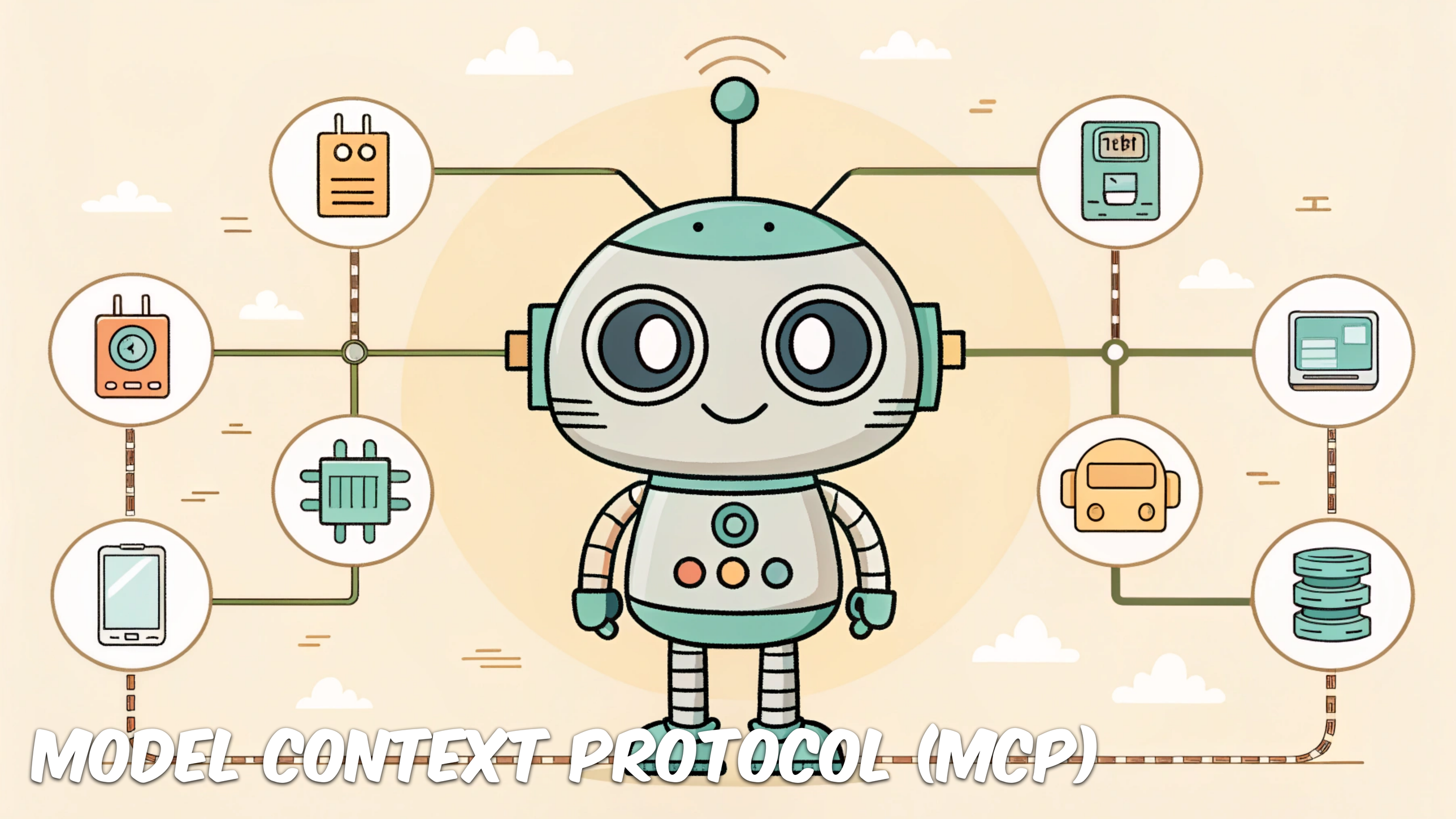
Taking Action

Tools in this category can be used to take action in a software system, such as sending an email, creating a new record in a database, submitting a form, or triggering a workflow. The goal is to automate tasks that would otherwise require human intervention or explicit programming. For example, a tool can be used to book a flight for a customer interacting with a chatbot, to fill out a form on a web page, or to implement a Java class based on an automated test (TDD) in a code generation scenario.


Tool
3
4
Spring AI
Chat Request
Tool Definition
name
description
input schema
1
Dispatch Tool
Call Requests
6
Chat Response
2
5
AI Model

```java
public class DatTimeTools {

    @Tool(description = "Get the current date and time in the user's timezone")
    String getCurrentDateTime() {
        return LocalDateTime.now().atZone(LocaleContextHolder.getTimeZone().toZoneId()).toString();
    }

}
```

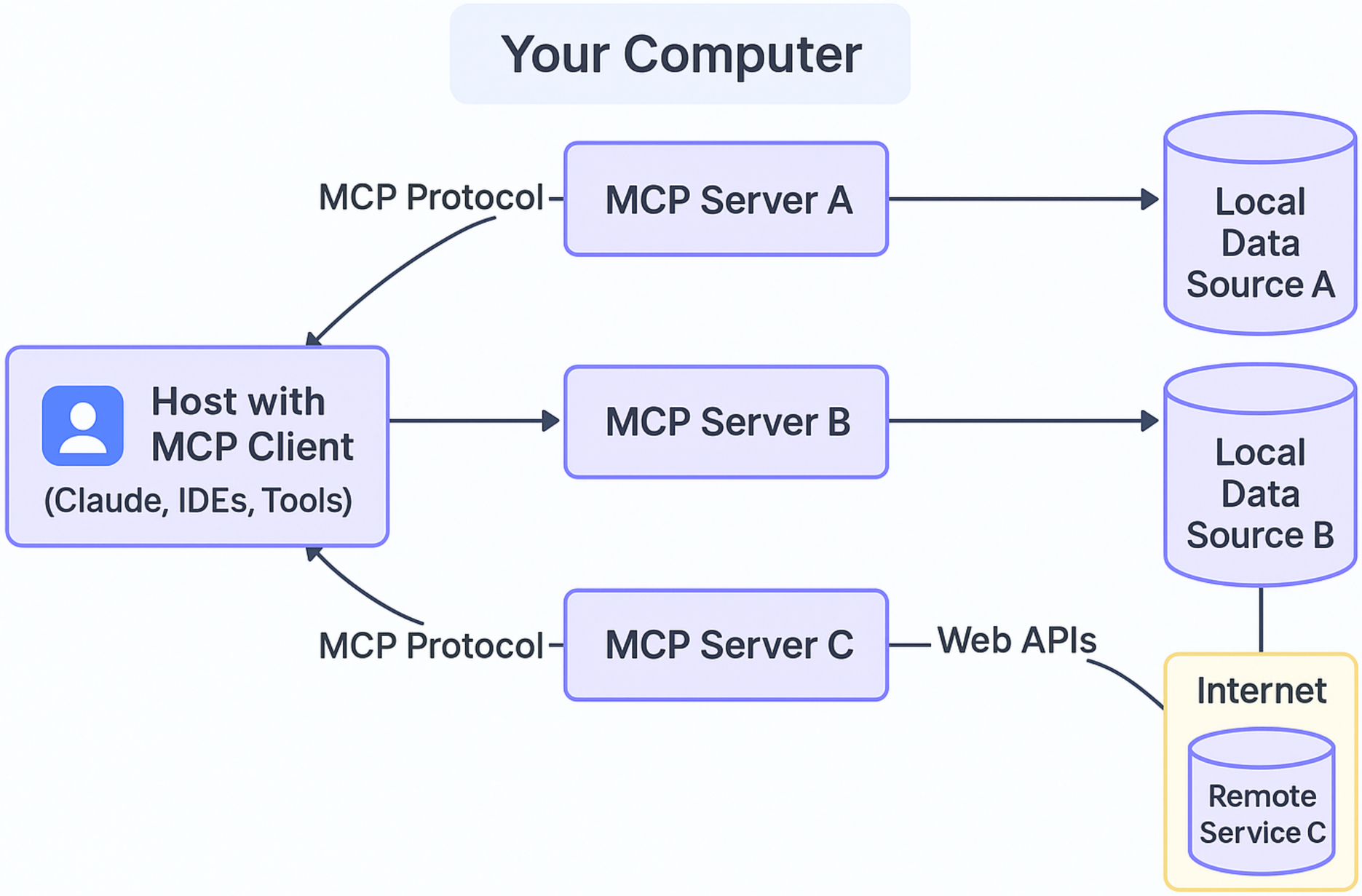
```java
@RestController
public class DatTimeChatController {

    private final ChatClient chatClient;

    public DatTimeChatController(ChatClient.Builder builder) {
        this.chatClient = builder
                .build();
    }

    @GetMapping("/tools")
    public String tools() {
        return chatClient.prompt("What day is tomorrow?")
                .tools(new DatTimeTools())
                .call()
                .content();
    }

}
```


MODEL CONTEXT PROTOCOL (MCP)

WHY MCP?

MCP helps you build agents and complex workflows on top of LLMs. LLMs frequently need to integrate with data and tools, and MCP provides:

- A growing list of pre-built integrations that your LLM can directly plug into
- The flexibility to switch between LLM providers and vendors
- Best practices for securing your data within your infrastructure


Your Computer
MCP Protocol
MCP Server A
Local Data Source A
Host with MCP Client
(Claude, IDEs, Tools)
MCP Server B
Local Data Source B
MCP Protocol
MCP Server C
Web APIs
Internet
Remote Service C

TRANSPORTS

Standard Input/Output (stdio)

- Use stdio when:
  - Building command-line tools
  - Implementing local integrations
  - Needing simple process communication
  - Working with shell scripts

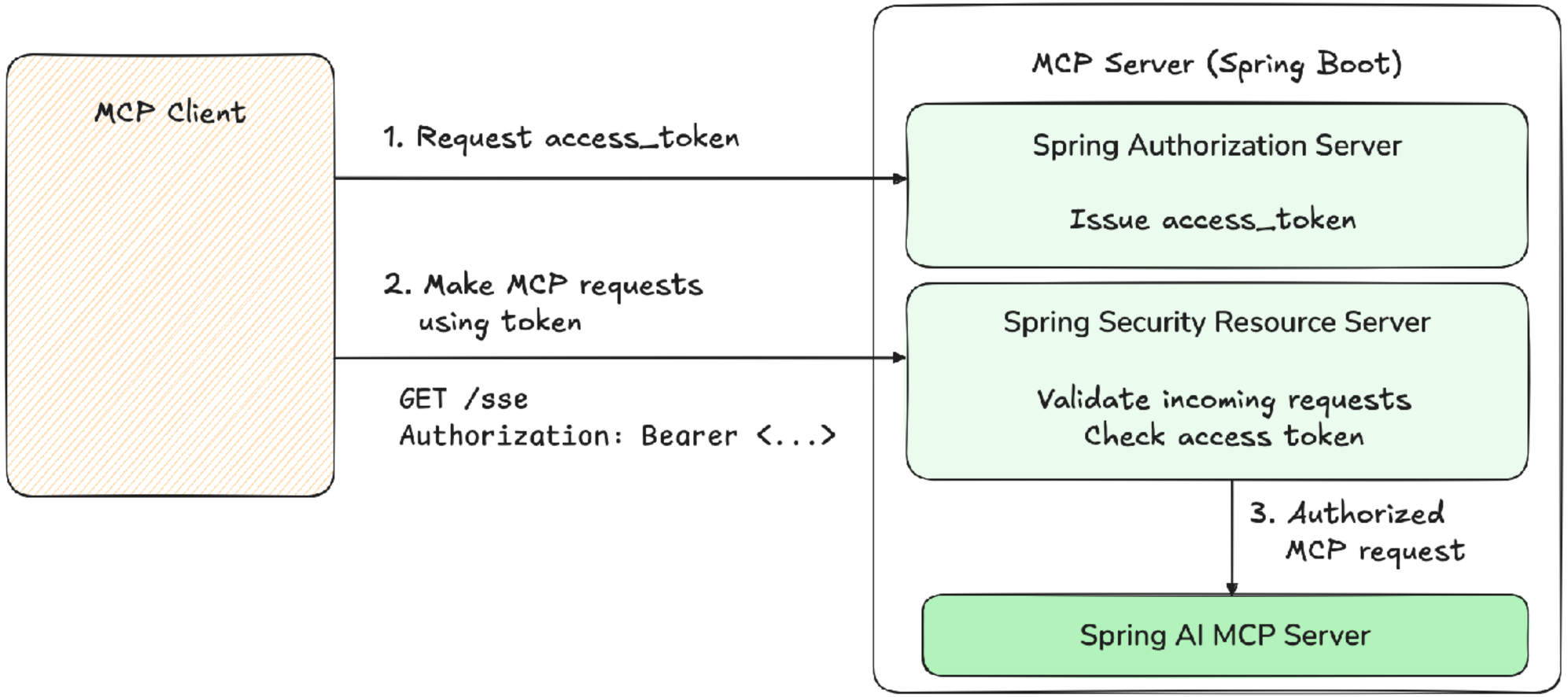
Server-Sent Events (SSE)

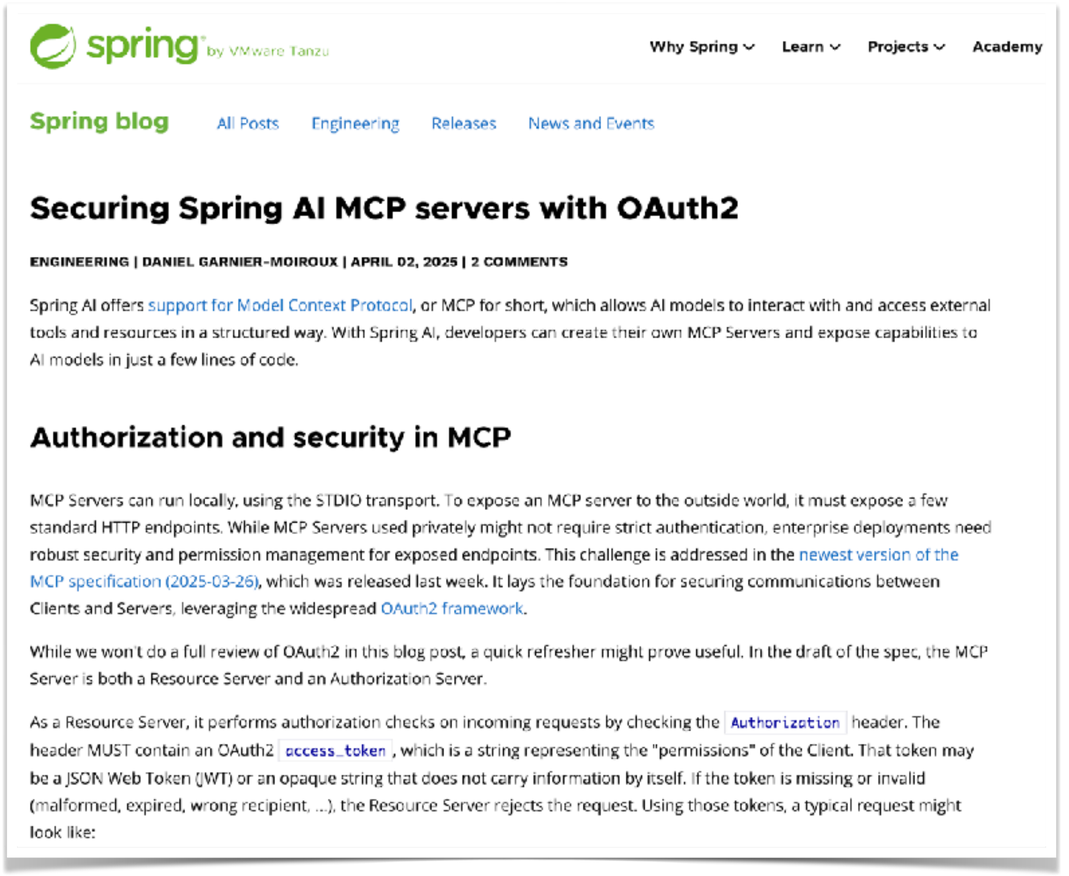
- Use SSE when:
  - Only server-to-client streaming is needed
  - Working with restricted networks
  - Implementing simple updates

SECURING MCP SERVERS

- **Security Challenge:** While local MCP servers (STDIO transport) may not need authentication, enterprise HTTP deployments require robust security and permission management
- **OAuth2 Integration:** New MCP spec (2025-03-26) leverages OAuth2 framework - MCP server acts as both Resource Server (validates tokens) and Authorization Server (issues tokens)
- **Implementation Requirements:**
  - Add Spring Security & Spring Authorization Server Dependencies
  - Configure OAuth2 client credentials in application.properties
  - Create SecurityFilterChain to handle authentication and token validation

SECURING MCP SERVERS





spring by VMware Tanzu

Why Spring Learn Projects Academy

Spring blog All Posts Engineering Releases News and Events

Securing Spring AI MCP servers with OAuth2

ENGINEERING | DANIEL GARNIER-MOIROUX | APRIL 02, 2025 | 2 COMMENTS

Spring AI offers support for Model Context Protocol, or MCP for short, which allows AI models to interact with and access external tools and resources in a structured way. With Spring AI, developers can create their own MCP Servers and expose capabilities to AI models in just a few lines of code.

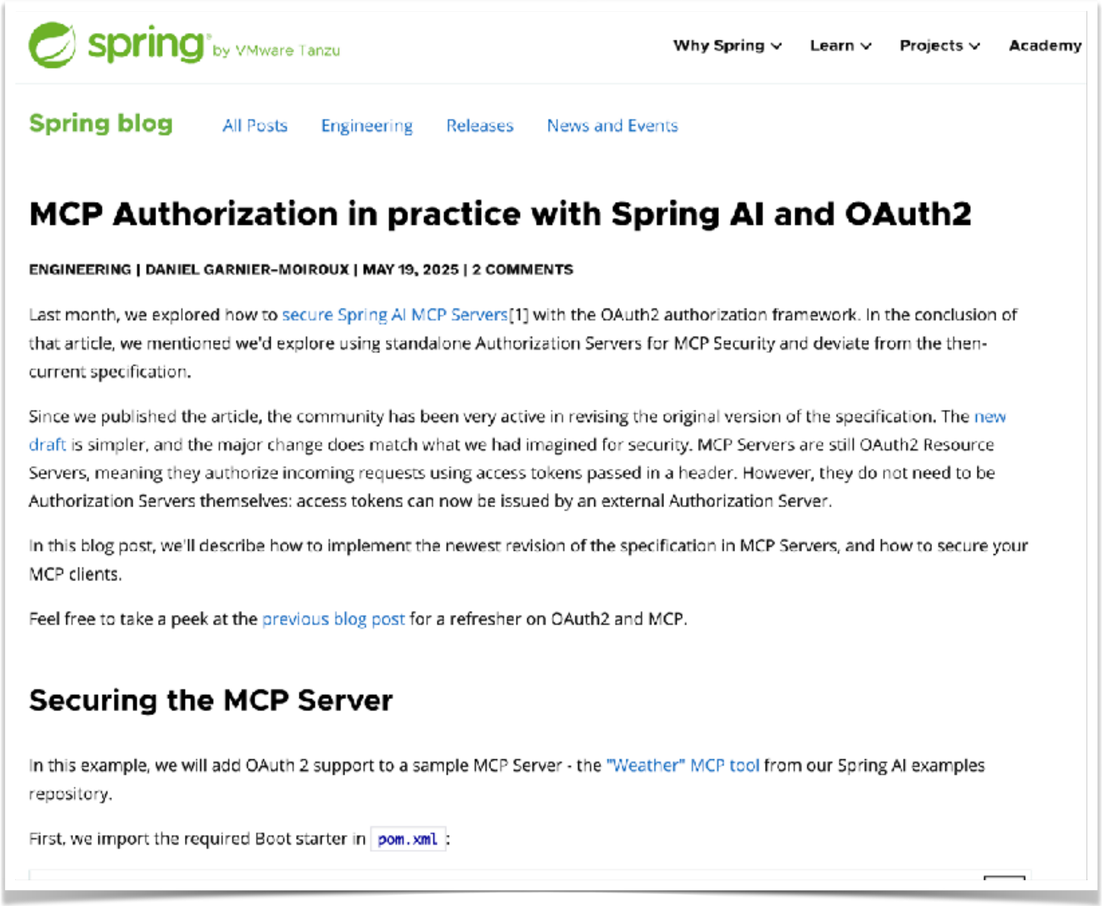
Authorization and security in MCP

MCP Servers can run locally, using the STDIO transport. To expose an MCP server to the outside world, it must expose a few standard HTTP endpoints. While MCP Servers used privately might not require strict authentication, enterprise deployments need robust security and permission management for exposed endpoints. This challenge is addressed in the newest version of the MCP specification (2025-03-26), which was released last week. It lays the foundation for securing communications between Clients and Servers, leveraging the widespread OAuth2 framework.

While we won't do a full review of OAuth2 in this blog post, a quick refresher might prove useful. In the draft of the spec, the MCP Server is both a Resource Server and an Authorization Server.

As a Resource Server, it performs authorization checks on incoming requests by checking the `Authorization` header. The header MUST contain an OAuth2 `access_token`, which is a string representing the "permissions" of the Client. That token may be a JSON Web Token (JWT) or an opaque string that does not carry information by itself. If the token is missing or invalid (malformed, expired, wrong recipient, ...), the Resource Server rejects the request. Using those tokens, a typical request might look like:

https://spring.io/blog/2025/04/02/mcp-server-oauth2



spring by VMware Tanzu

Why Spring Learn Projects Academy

Spring blog All Posts Engineering Releases News and Events

MCP Authorization in practice with Spring AI and OAuth2

ENGINEERING | DANIEL GARNIER-MOIROUX | MAY 19, 2025 | 2 COMMENTS

Last month, we explored how to secure Spring AI MCP Servers[1] with the OAuth2 authorization framework. In the conclusion of that article, we mentioned we'd explore using standalone Authorization Servers for MCP Security and deviate from the then-current specification.

Since we published the article, the community has been very active in revising the original version of the specification. The new draft is simpler, and the major change does match what we had imagined for security. MCP Servers are still OAuth2 Resource Servers, meaning they authorize incoming requests using access tokens passed in a header. However, they do not need to be Authorization Servers themselves: access tokens can now be issued by an external Authorization Server.

In this blog post, we'll describe how to implement the newest revision of the specification in MCP Servers, and how to secure your MCP clients.

Feel free to take a peek at the previous blog post for a refresher on OAuth2 and MCP.

Securing the MCP Server

In this example, we will add OAuth 2 support to a sample MCP Server - the "Weather" MCP tool from our Spring AI examples repository.

First, we import the required Boot starter in `pom.xml`:

https://spring.io/blog/2025/05/19/spring-ai-mcp-client-oauth2

TESTING YOUR MCP SERVERS

- Create an executable JAR
- Test using an MCP Client
  - Spring MCP Client
  - Claude Desktop
  - Cursor / Windsurf / Junie
  - Any MCP Client of your choice

MCP Inspector v0.14.2

Transport Type

STDIO

Command

/Users/vega/.sdkman/candidates/

Arguments

-jar /Users/vega/Downloads/spri

Environment Variables

Server Entry | Servers File

Configuration

Restart | Disconnect

Connected

Logging Level

debug

System

Resources | Prompts | Tools | Ping | Sampling | Roots | Auth

Tools

List Tools

Clear

spring-io-sessions

Returns all sessions for Spring I/O 2025 Conference

spring-io-sessions

Returns all sessions for Spring I/O 2025 Conference

Run Tool

Tool Result: Success


```
[
  0: {
    day: "2025-05-22"
    time: "08:00"
    title: "Registration"
    type: "logistics"
    speakers: []
    room: "General"
  }
  1: {
    day: "2025-05-22"
    time: "09:00"
    title: "Welcome"
    type: "opening"
    speakers: [
```


History

6. ping ▶

5. initialize ▶

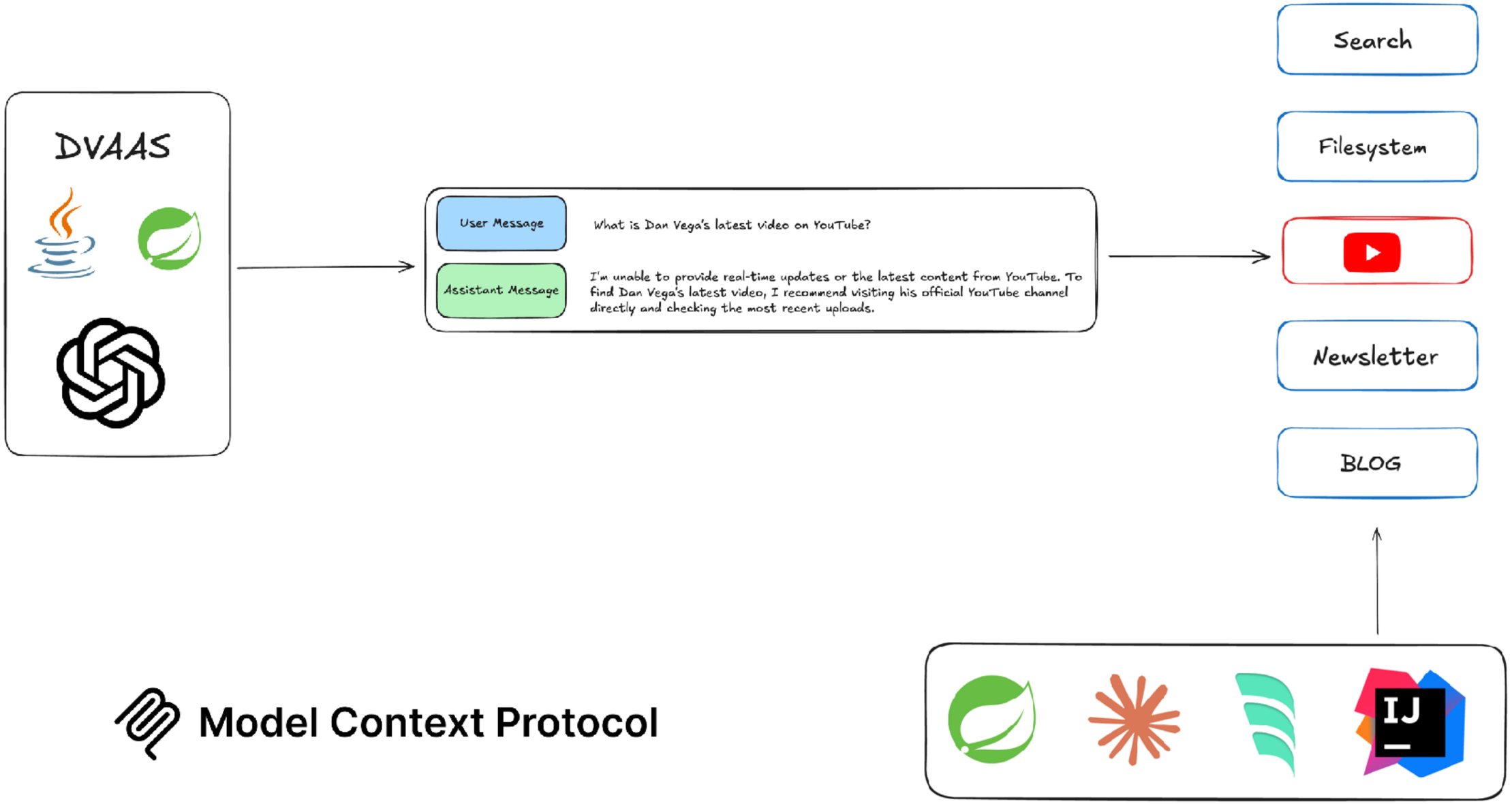
4. tools/call ▶

3. tools/list ▶

2. resources/list ▶

Server Notifications

No notifications yet


DVAAS
User Message
What is Dan Vega's latest video on YouTube?
Assistant Message
I'm unable to provide real-time updates or the latest content from YouTube. To find Dan Vega's latest video, I recommend visiting his official YouTube channel directly and checking the most recent uploads.
Search
Filesystem
Newsletter
BLOG
IJ
Model Context Protocol

THE SPRING
DEVELOPER CONFERENCE

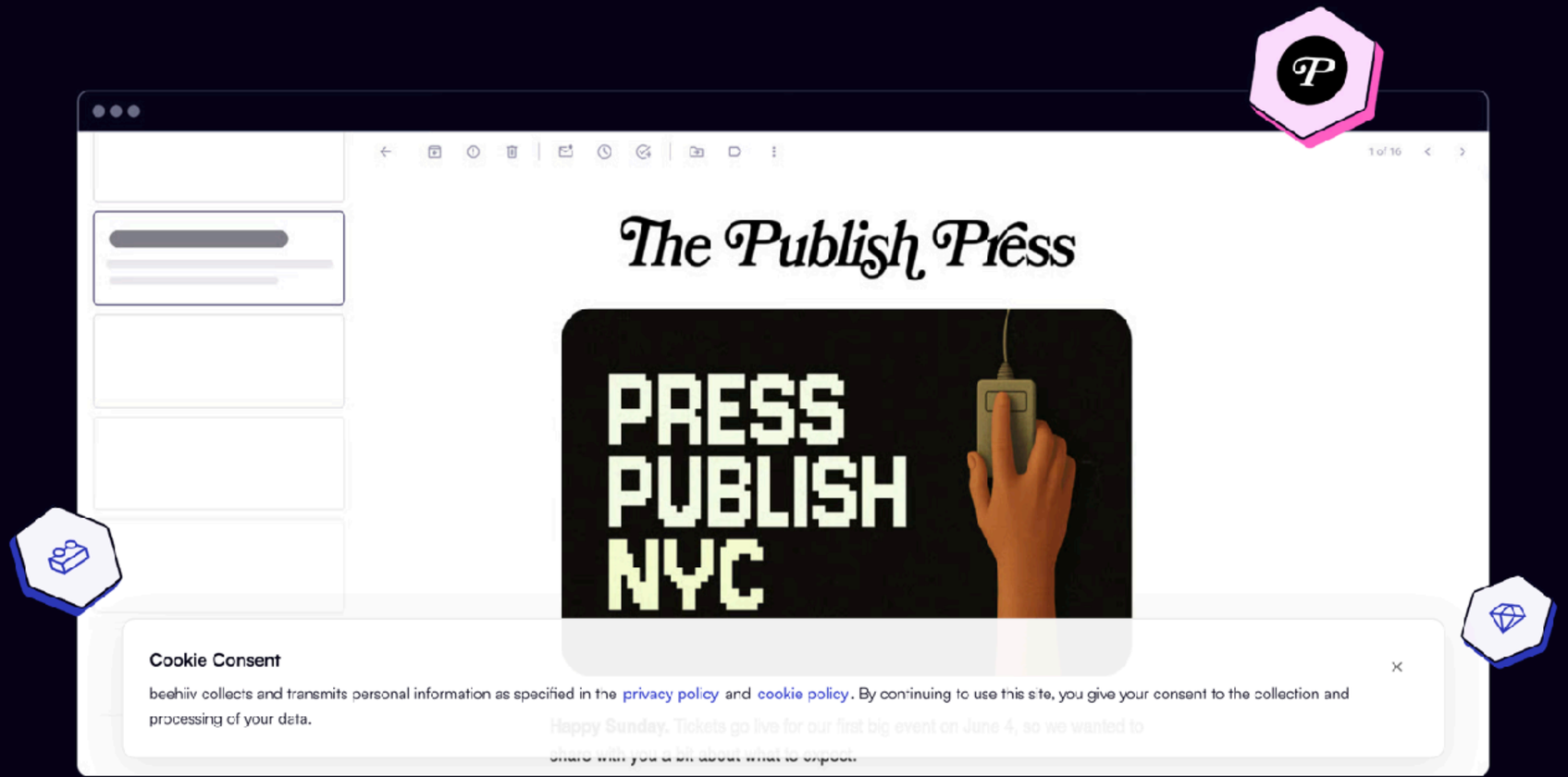
Be part of the future and the next generation of Spring applications

THE ONE PLACE TO BUILD NEWSLETTERS

Launch, engage, and earn with the all-in-one growth system for digital content publishers.

★★★★★ 5/5 from 19,386 customers

Start your free trial → View Plans



OPEN SOURCE VS PROPRIETARY MODELS

Open Source AI Model Checklist

Model Weights

OSI

Source Code

Training Data

OPEN SOURCE MODELS - KEY CHALLENGES

- Performance Gap vs Frontier Models
- Up-Front Hardware & Ops Overhead
- No Vendor-Grade Support or SLA
- Safety & Quality Risks (Hallucinations/Bias)
- Security & Supply-Chain Exposure
- License & IP Ambiguity
- Benchmark / Eval Burden Rests on You
- Rapid-Fire Releases - Maintenance Churn

WHY OPEN SOURCE MODELS

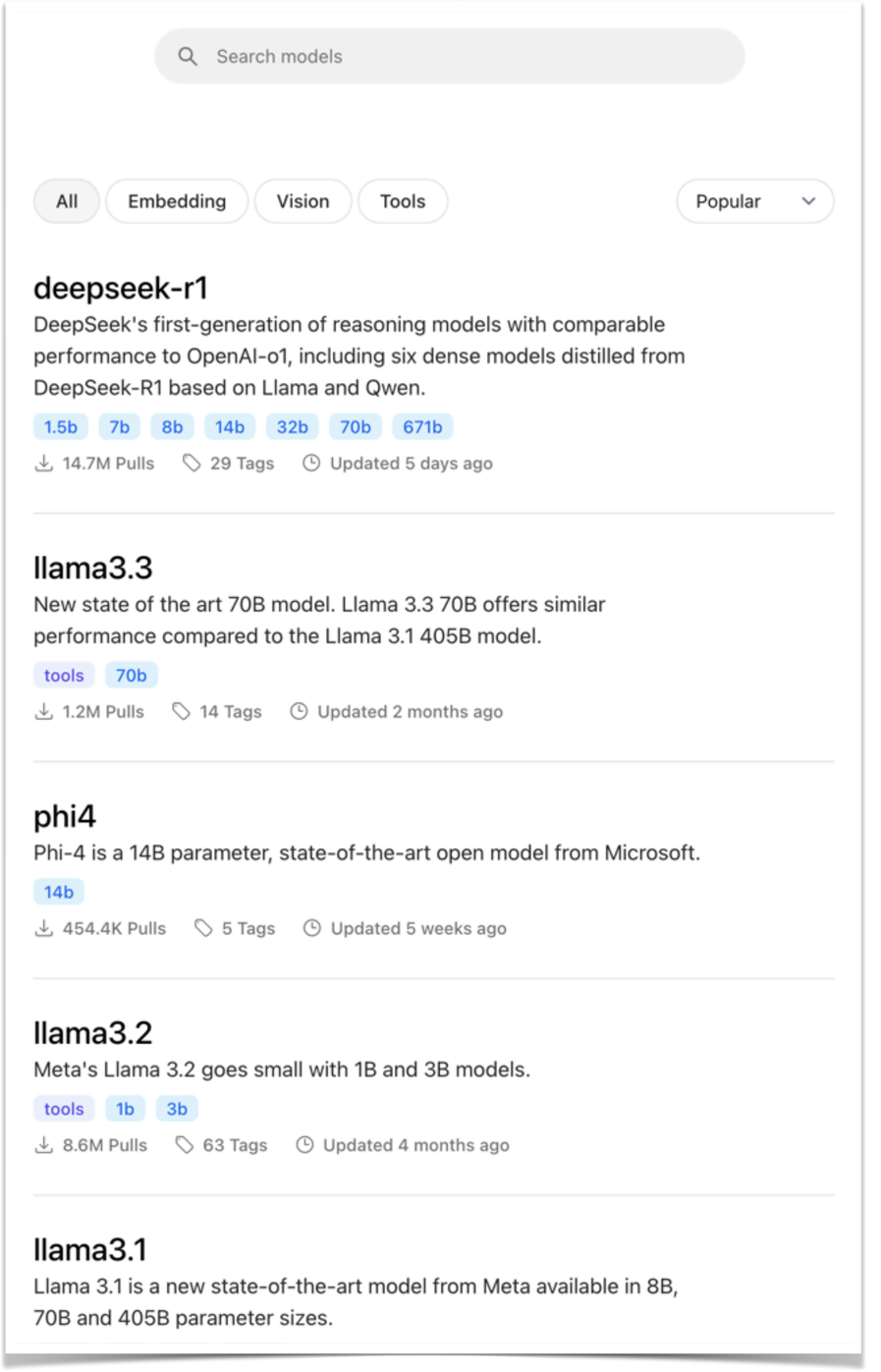
- Transparency & Auditability
- Self-Hosting for Data Privacy & Compliance
- Lower Total Cost (No API metering)
- Low-Latency Local Development
- Full Customization & Fine-Tuning Rights
- Community-Driven Innovation & Fast Patch Cycles
- Deployment Flexibility (cloud, on-prem, edge)
- No Vendor Lock-In

HOW TO EVALUATE OPEN SOURCE MODELS

- Licence & “Openness” – OSI-approved? any commercial-use limits?
- Capability Benchmarks – MMLU, HellaSwag, GSM-8K scores; check the Open LLM Leaderboard
- Tool / Function-Calling Support – built-in schemas for agents & Spring AI’s callable() interface
- Size vs Hardware Budget – parameter count, VRAM need, quantised variants
- Context Window – 8 K, 128 K… even 1 M tokens for long-doc RAG
- Latency & Throughput – tokens/sec locally vs on-prem GPU or HF Endpoint
- Community & Release Cadence – active issues, model-card updates, patch frequency
- Security & Supply Chain – reproducible weights, SBOM, no suspicious commits

Ollama

https://ollama.com



Docker Model Runner

Page options

Availability: Beta

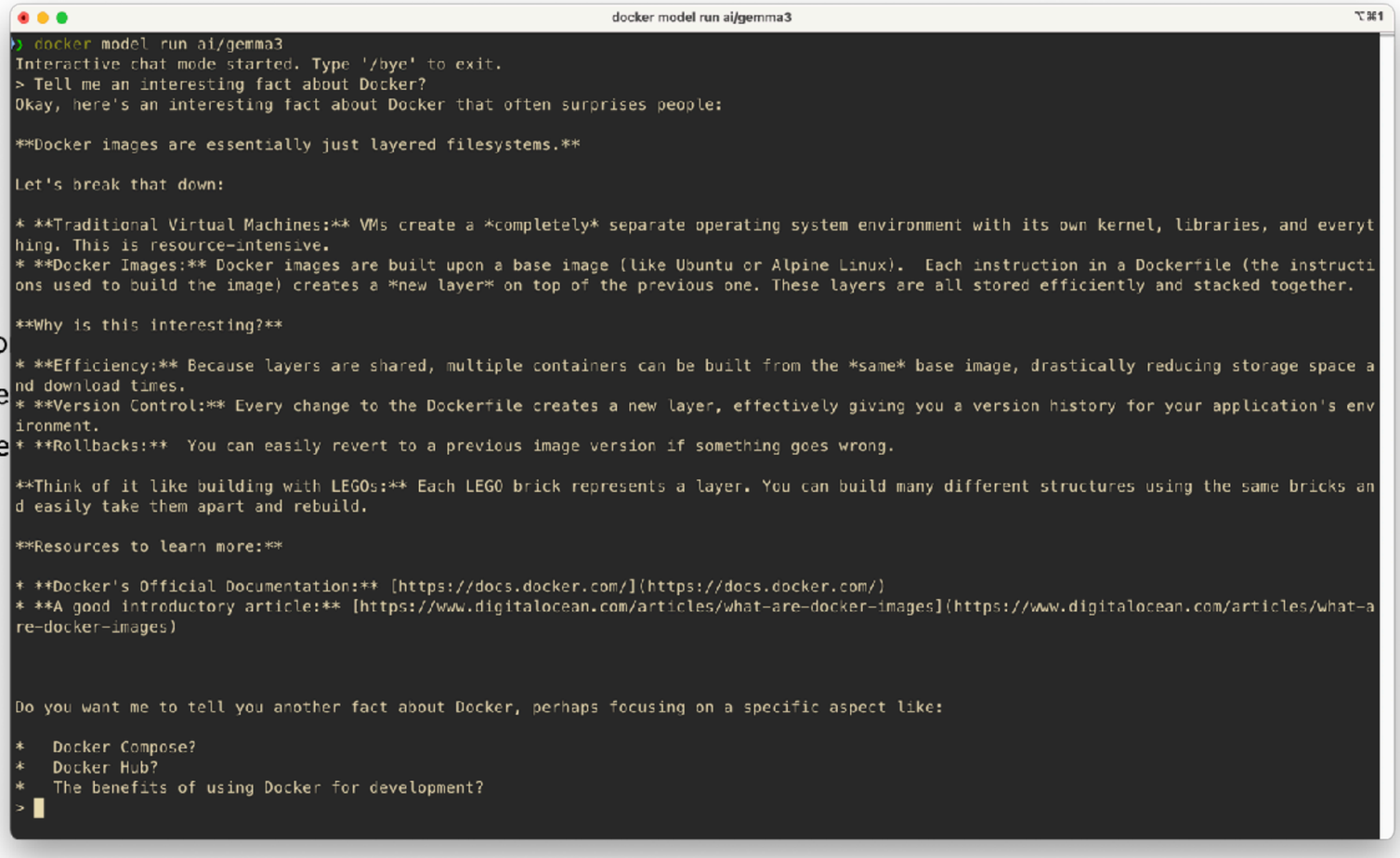
Requires: Docker Desktop 4.40 and later

For: Docker Desktop for Mac with Apple Silicon

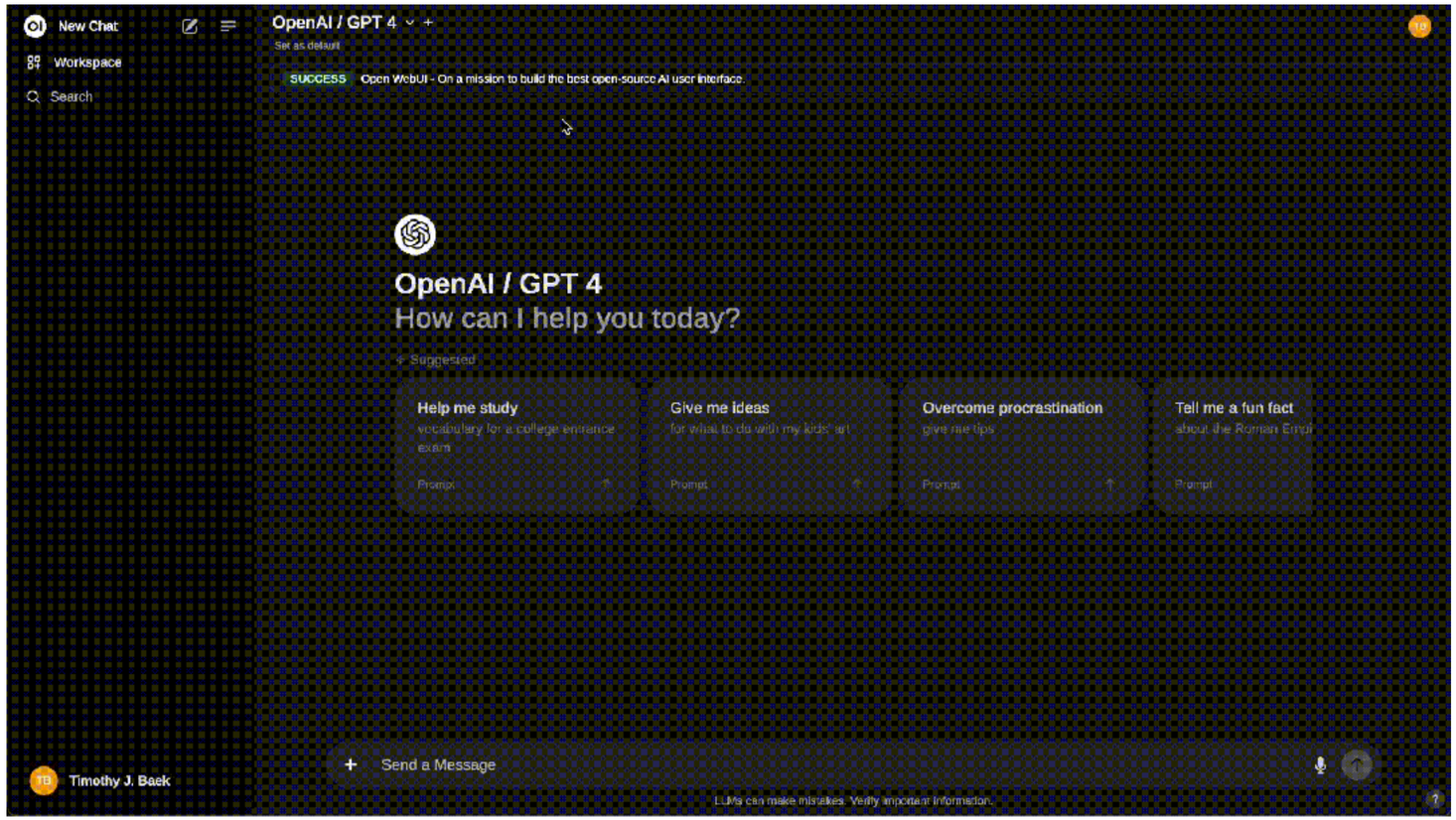
The Docker Model Runner plugin lets you:

- Pull models from Docker Hub
- Run AI models directly from the command line
- Manage local models (add, list, remove)
- Interact with models using a submitted prompt or in chat mode

Models are pulled from Docker Hub the first time they're used and sto
when a request is made, and unloaded when not in use to optimize re
take some time — but after that, they're cached locally for faster acce
compatible APIs.



Open WebUI



https://openwebui.com

Chats

Secret project

C++ Filesyste... (1600 tokens)

log about version of ... (56 tokens)

Financial analysis (106 tokens)

lmstudio-community/Meta-Llama-3.1-8B-Instruct-GGUF

Eject

C++ Filesystem Implementation

Assistant Meta-Llama-3.1-8B-Instruct-GGUF

Filesystem Implementation in C++

Before we begin, let's outline our approach:

1. We'll create a basic directory structure with methods for creating directories, deleting directories, listing files and directories, and navigating through the directory tree.
2. We'll implement file operations such as creating, reading, writing, and deleting files.
3. We'll handle errors and exceptions to ensure robustness.

Filesystem Class

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <dirent.h>

class Filesystem {
public:
    // Constructor
    Filesystem(const std::string& rootDir) : rootDir_(rootDir) {}

    // Create a new directory
    void createDirectory(const std::string& path);
```

Type a message and press Enter to send ...

User (⌘↓)

Insert (⌘/)

Advanced Configuration

Preset

Discard Unsaved

Coding Helper (C++)

Commit Changes

Save As New...

System Prompt

You are an incredibly good C++ engineer. Think about the problems you're about to solve step-by-step. First make a plan, and then ask the user to confirm. Only then act on it.

Token count: 41

Settings

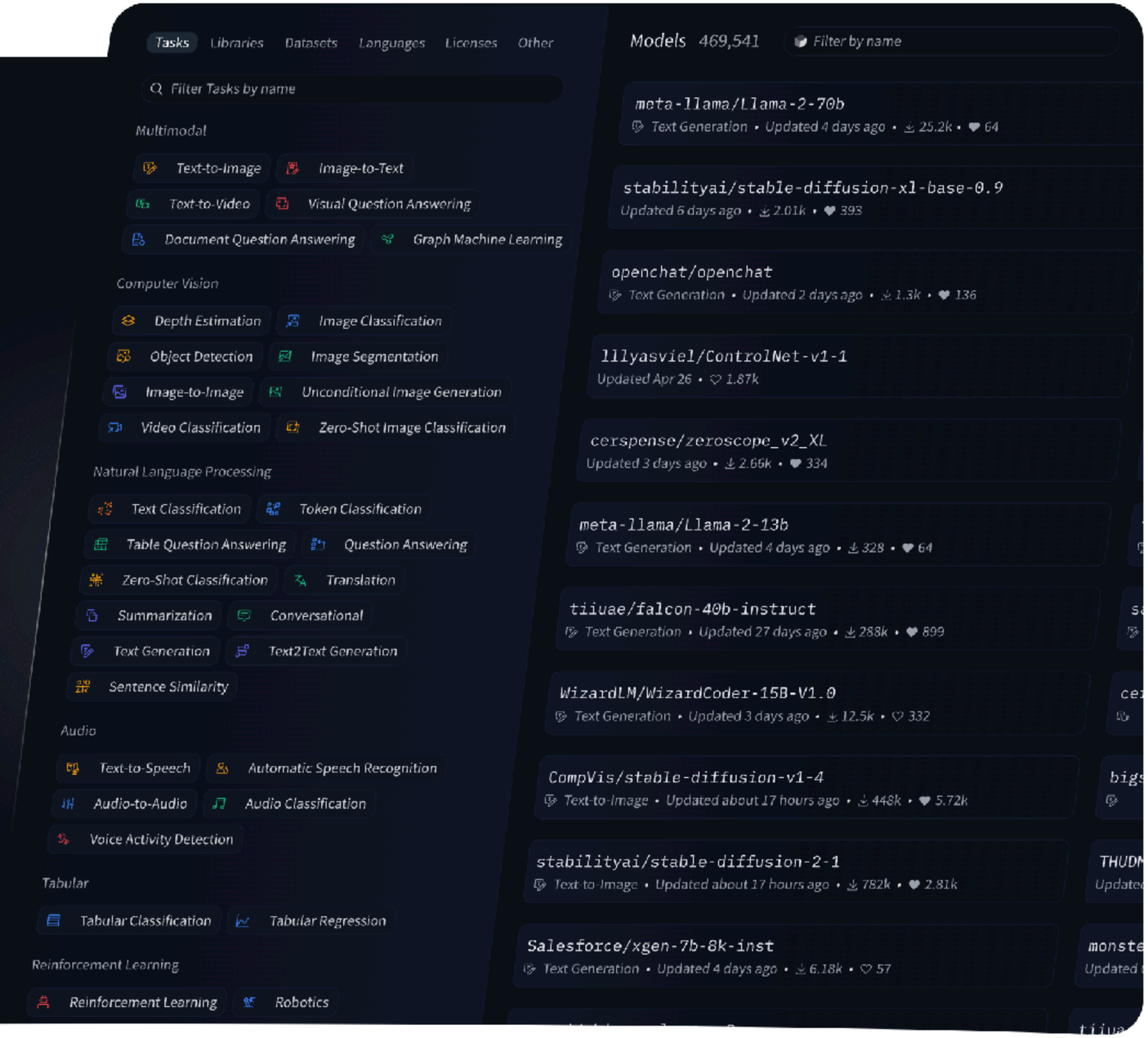
Sampling

Structured Output

The AI community building the future.

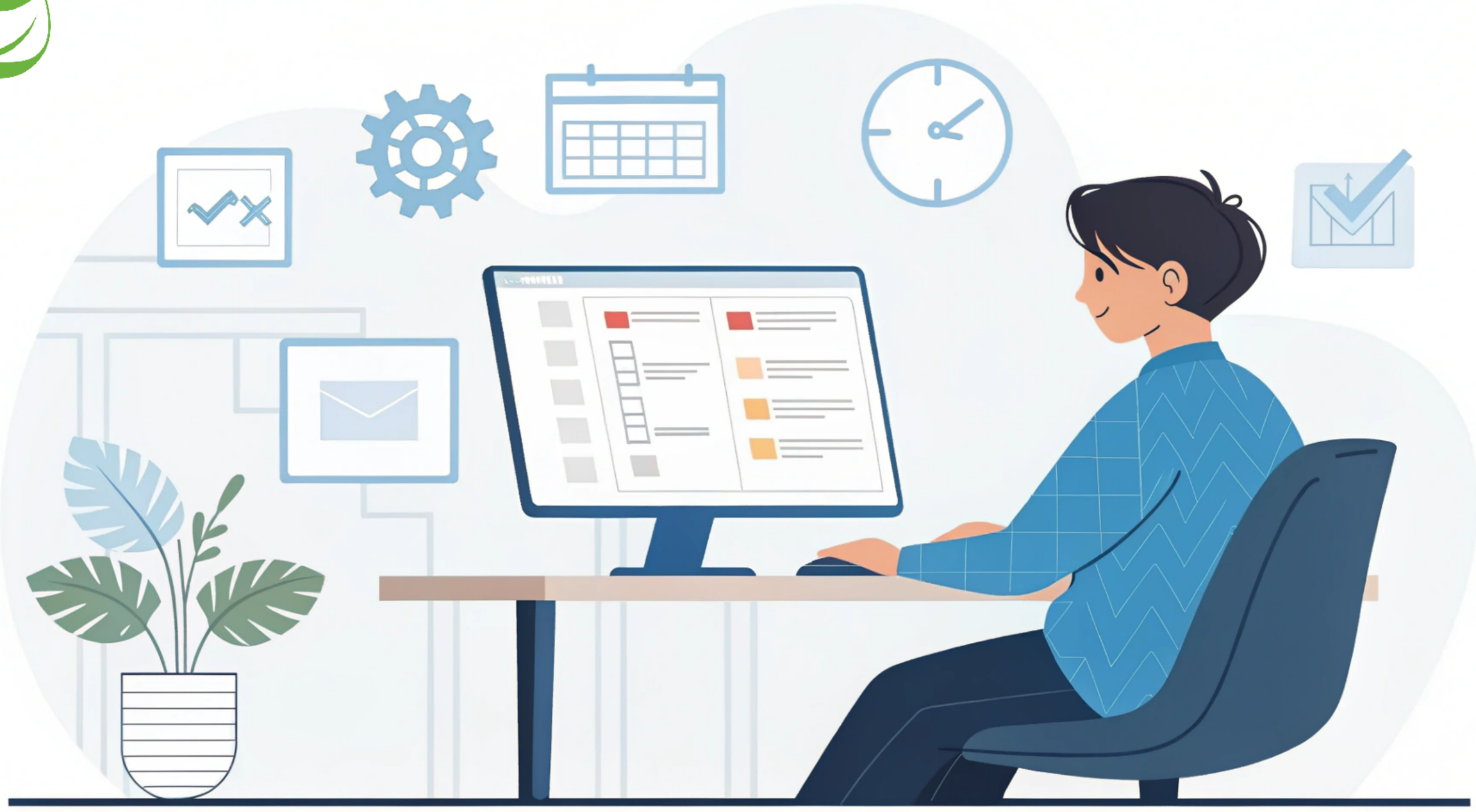
The platform where the machine learning community collaborates on models, datasets, and applications.

Explore AI Apps or Browse 1M+ models



CHECK OUT MY DEMO

OBSERVABILITY



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

WHY OBSERVABILITY

- Cost can spike ⟶ need token & $ metrics
- • LLMs are nondeterministic ⟶ traces & logs for debug
- • Safety / legal ⟶ need evidence when things go wrong

THE 3+1 PILLARS OF OBSERVABILITY

- Metrics – latency, token counts, cost
- Logs – structured prompt / response (PII-redacted)
- Traces – end-to-end view of RAG pipeline
- Evaluations – automated quality checks (FactCheckingEvaluator)

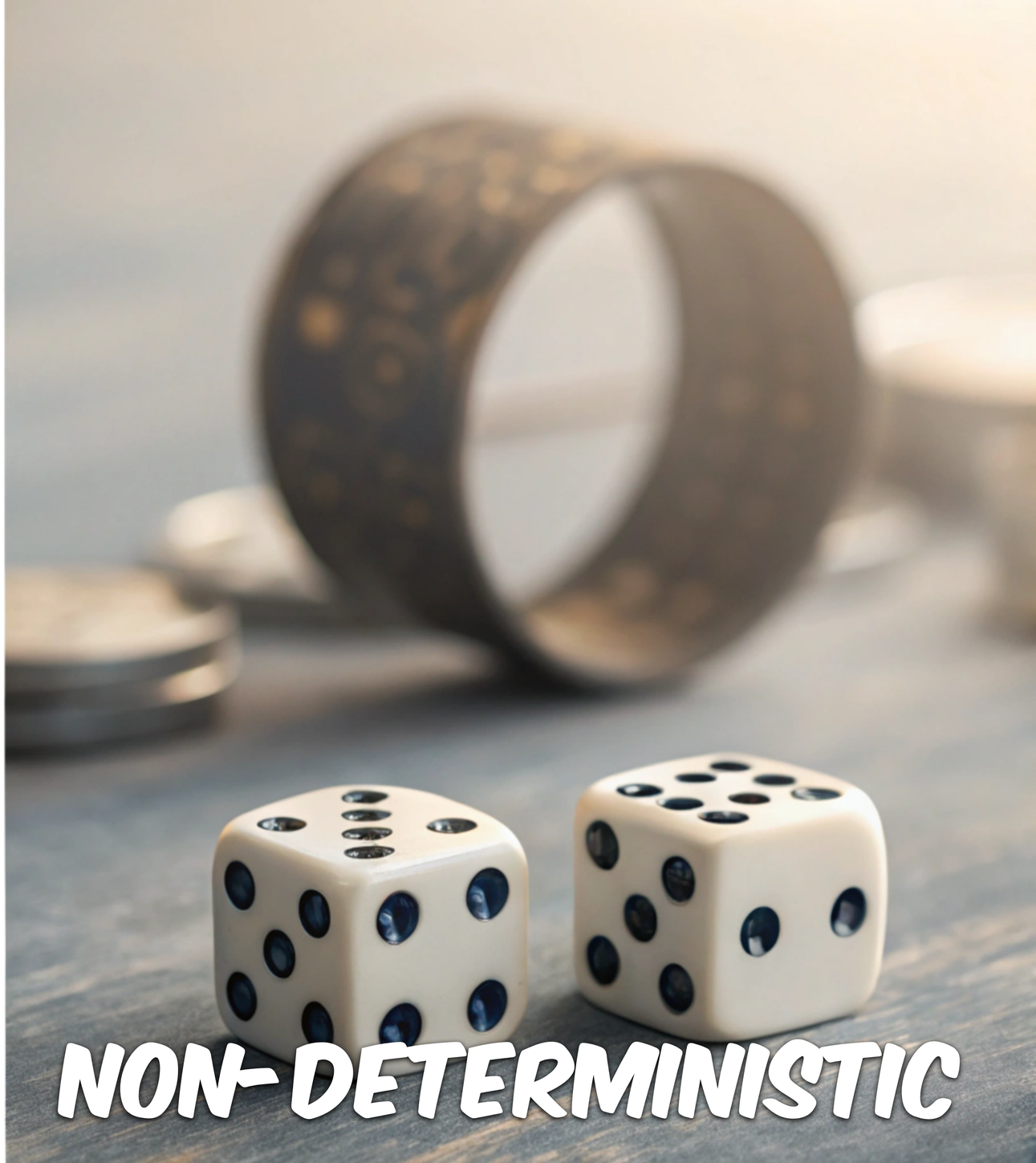
WHAT SPRING GIVES YOU FOR FREE

- Micrometer metric names (gen_ai.usage.*, gen_ai.client.latency) (docs.spring.io)
- • Observation events → Micrometer Tracing / OpenTelemetry
- • Pluggable log masking (PII filter)
- • GenerationListener hook for custom metrics

CHECK OUT MY DEMO

EVALUATIONS


DETERMINISTIC


NON-DETERMINISTIC

WHY EVALUATE AI RESPONSES

The Challenge

- AI Models can generate plausible but incorrect responses (hallucinations)
- Traditional Unit tests don’t work for AI-generated content
- Need systematic ways to validate AI output quality

The Solution

- Use AI to evaluate AI responses
- Structured evaluation frameworks
- Automated quality checks in your pipeline

SPRING AI EVALUATION FRAMEWORK

Core Interface

```
@FunctionalInterface
public interface Evaluator {
    EvaluationResponse evaluate(EvaluationRequest evaluationRequest);

}
```

Evaluation Request Components

- **userText** - Original user input
- **dataList** - Context data (e.g., from RAG)
- **responseContent** - AI model's response

TWO KEY EVALUATORS

Relevancy Evaluator

- Purpose: Is there AI response relevant to the user's question?
- Best for:
  - RAG (Retrieval Augmented Generation)
  - Ensuring responses stay on topic
  - Quality Control for chatbots

Fact Checking Evaluator

- Purpose: Is the AI response factually accurate?
- Best for:
  - Detecting hallucinations
  - Verifying claims against source material
  - Content validation

RELEVANCY EVALUATOR DEEP DIVE

What it does:

- Compares AI response against retrieved context
- Asks: “Does this response answer the users’s question?”
- Returns YES/NO evaluation

Default Prompt Template

```
Your task is to evaluate if the response for the query is in line with
the context information provide
```

Answer YES if relevant, otherwise NO.

FACT CHECKING EVALUATOR DEEP DIVE

What it does:

- Verifies claims against provided documents
- Detects factual inaccuracies and hallucinations
- Can use specialized models like *Bespoke-Minicheck*
    - Accurate, Small, Fast & Cost-effective

Evaluation Format:

```
Document: {context}
Claim: {ai_response}
```

TESTING DETERMINISTIC AI TASKS

Classification Tasks

More predictable outcomes = Traditional testing approaches

Examples:

- Sentiment analysis (positive/negative/neutral)
- Content Moderation (safe/unsafe)
- Intent detection (question/request/complaint)

TESTING DETERMINISTIC AI TASKS

```
@Test
void testSentimentClassification() {
    String positiveText = "I love this product!";
    String result = classifySentiment(positiveText);

    assertThat(result).isEqualTo("POSITIVE");
}
```

Key Difference: Expected outcomes are known and consistent

CHECK OUT MY DEMO

BEST PRACTICES FOR AI EVALUATION

Choose the Right Model

- ✅ Use your best model for evaluation (may differ from generation model)
- ✅ Consider cost vs. accuracy tradeoffs
- ✅ Specialized models (like Minicheck) for specific tasks

Test Systematically

- ✅ Include evaluation in your CI/CD pipeline
- ✅ Test both positive and negative cases
- ✅ Evaluate against real user scenarios

Custom Templates

- ✅ Customize prompts for your specific use case
- ✅ Include domain-specific evaluation criteria
- ✅ Maintain consistency across evaluations

WHEN TO USE EACH EVALUATOR

Relevancy Evaluator

- ✅ RAG applications
- ✅ Q&A systems
- ✅ Topic adherence
- ✅ Context-aware responses

Fact Checking Evaluator

- ✅ Knowledge-based applications
- ✅ Fact verification systems
- ✅ Content validation
- ✅ Hallucination detection

Traditional Unit Tests

- ✅ Classification tasks
- ✅ Structured outputs (JSON)
- ✅ Deterministic functions
- ✅ Consistent expected results

CONCLUSION

WHAT DID WE LEARN?

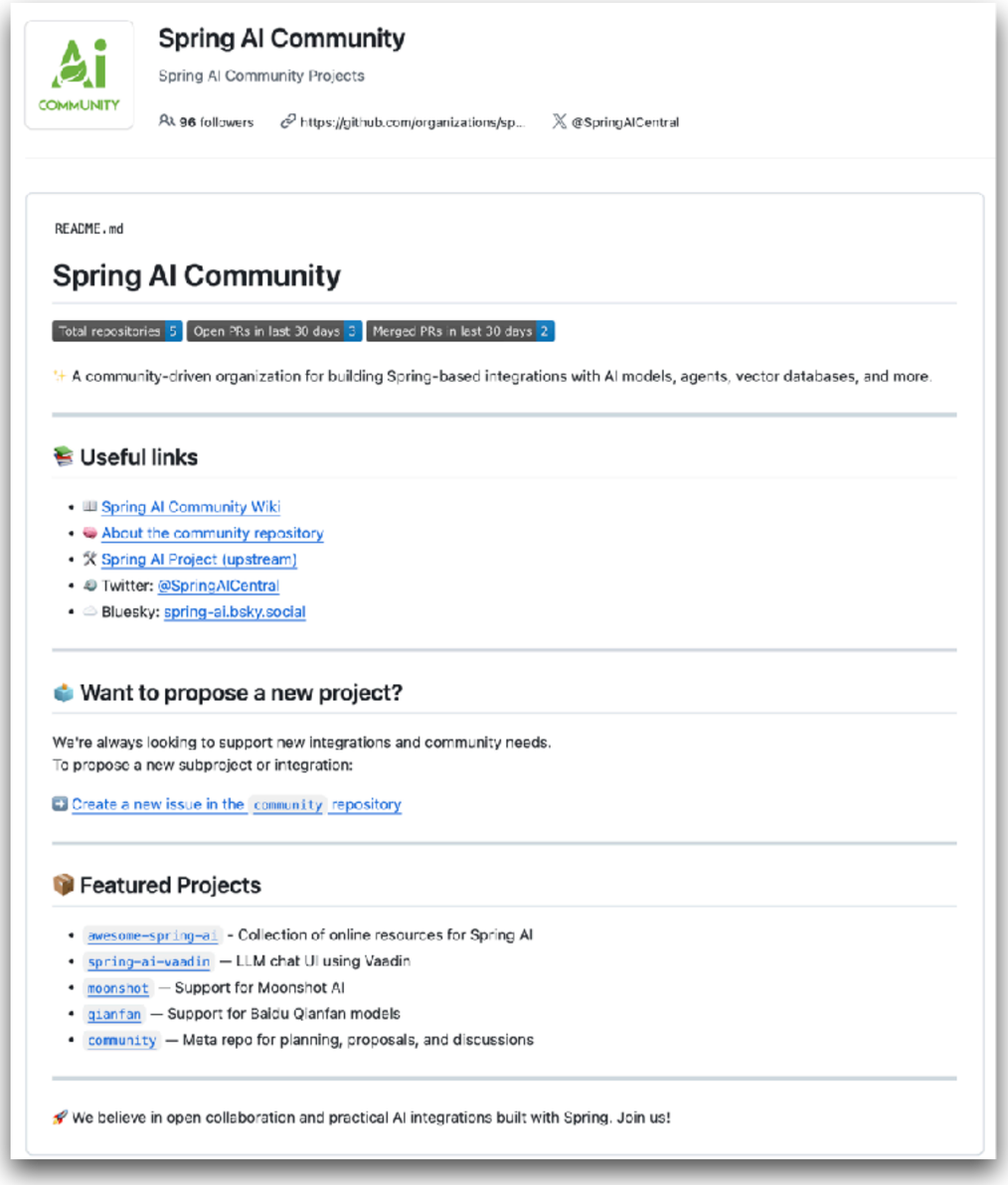
- **Fundamentals of AI:** We explored the key concepts of Artificial Intelligence, including Machine Learning, Deep Learning, and the transformer architecture that powers Large Language Models (LLMs).
- **Prompt Engineering is Key:** We learned that the quality of our input (prompts) directly impacts the quality of the AI's output. We covered several techniques, from basic-to-advanced, for crafting effective prompts, including zero-shot, one-shot, and few-shot prompting, as well as providing clear context and structure.
- **Spring AI for Java Developers:** We saw how Spring AI simplifies integrating artificial intelligence capabilities into Java applications. It provides abstractions for interacting with various AI models for chat, image, audio, and more.
- **Overcoming LLM Limitations:** We addressed the inherent limitations of LLMs, such as hallucinations and stale data, and discussed strategies like Retrieval Augmented Generation (RAG) and tool calling to make them more reliable and capable.
- **Evaluating AI Responses:** We learned that traditional testing methods are insufficient for non-deterministic AI outputs and explored how to use evaluator tools within Spring AI to check for relevance and factual accuracy.

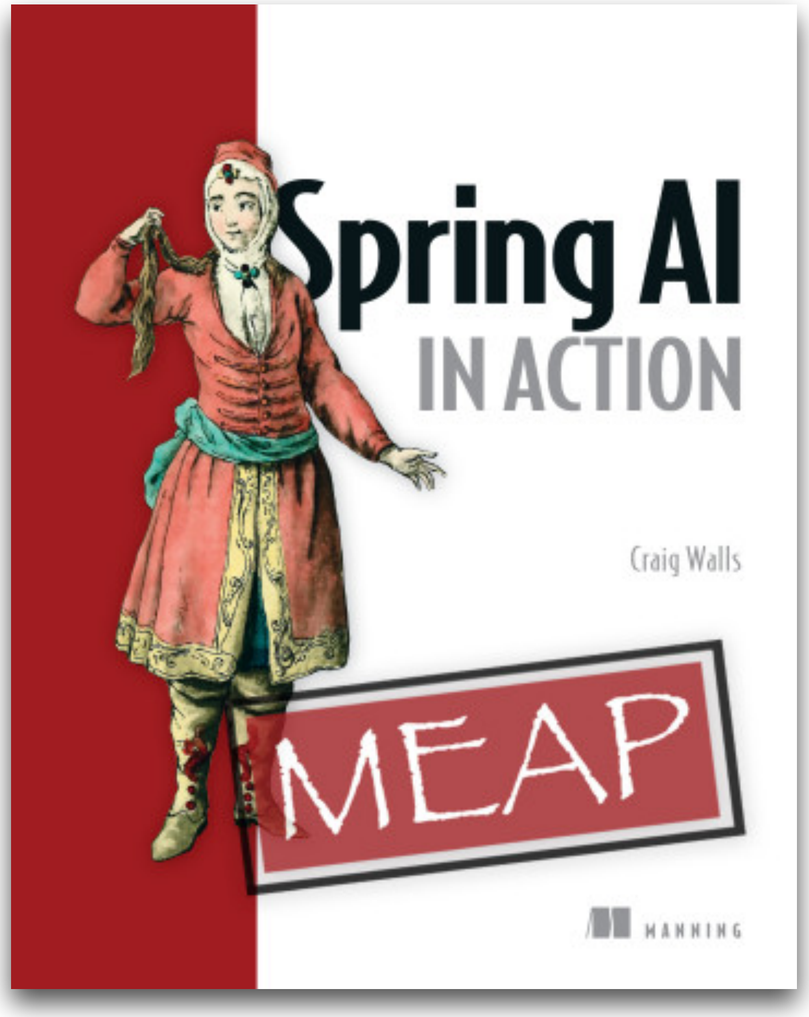
KEY TAKEAWAYS

- **AI is a Tool to be Mastered:** Large Language Models are powerful but not magical. They are pattern-matching machines that predict the next word. Your success in using them depends on your ability to communicate effectively through well-crafted prompts.
- **Prompt Engineering is a Developer Skill:** Don't just command the AI; teach it what you want. Providing context, examples, and a clear structure to your prompts will dramatically improve the quality of the response. Start saving your effective prompts as you create them.
- **Spring AI is Your Gateway:** Spring AI simplifies the process of integrating AI into your Java applications by providing a portable API that can work across different AI providers. This means you can build with AI without being locked into a single vendor.
- **Don't Trust, Verify:** LLMs can "hallucinate" and provide incorrect information with confidence. You must build guardrails. Use techniques like Retrieval Augmented Generation (RAG) to ground the model with your specific data , and implement evaluation checks to validate the accuracy and relevance of AI responses.
- **Start Building Now:** The best way to learn is by doing. Begin with a simple use case, get your API keys, and start experimenting. Use the provided resources like the Spring AI workshop code and documentation to actively code along and build a project of your own.

RESOURCES

- https://github.com/danvega/spring-ai-workshop
- Spring AI Reference Documentation
  - Spring AI Source Code
  - Spring AI Examples Repo
- Spring AI Community
- Spring Office Hours
- Dan Vega
  - http://www.danvega.dev
  - https://www.youtube.com/@danvega





https://www.manning.com/books/spring-ai-in-action


THANKFUL
FOR
BEING YOUR
TEACHER