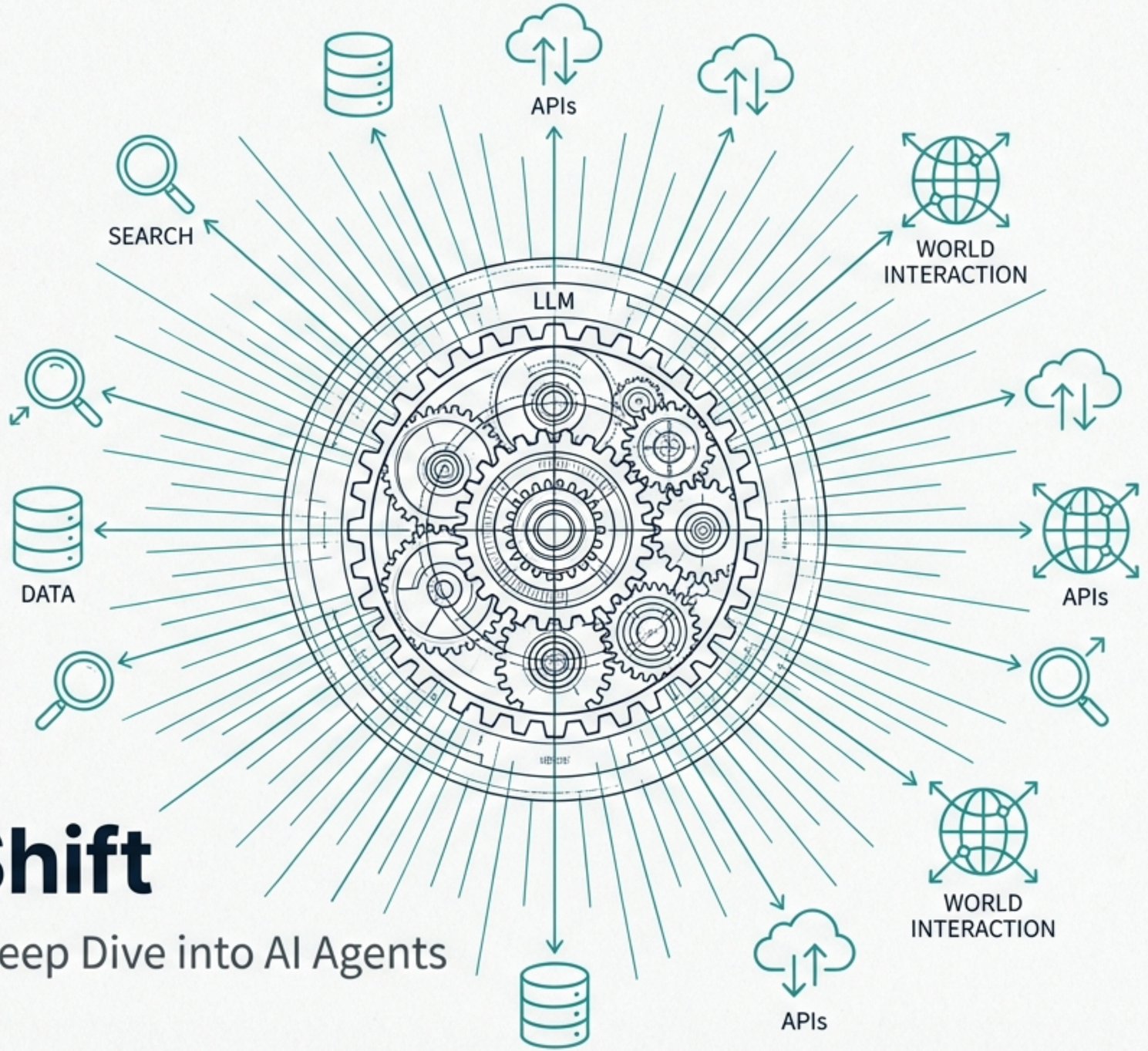
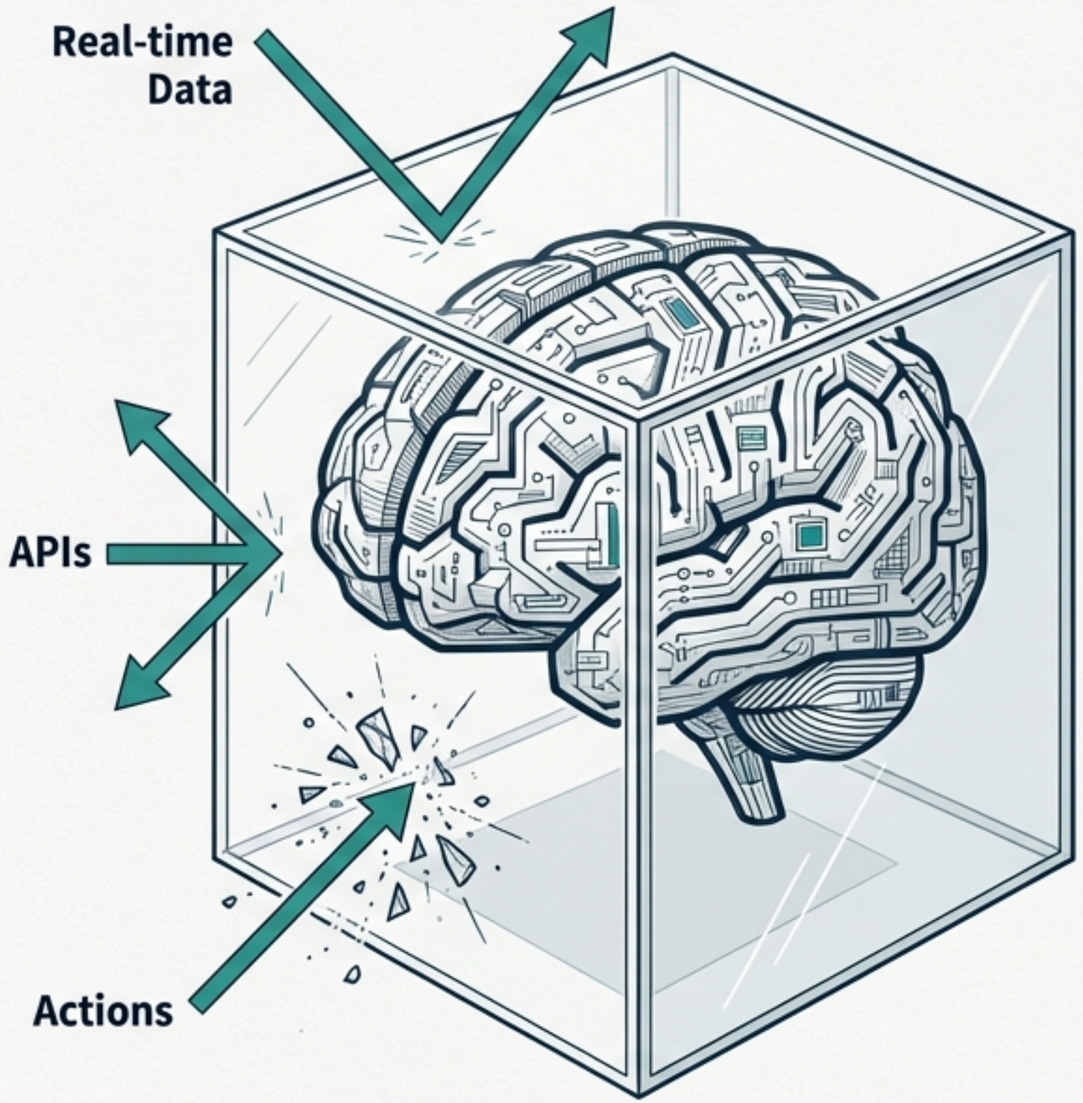


The Agentic Shift

From Prediction to Action: A Deep Dive into AI Agents



The LLM: A Brain in a Box

Today's LLMs are masters of sophisticated pattern matching, not true reasoning. Their power is constrained by their training data and inability to interact with the real world.

Core Function

- They are essentially "super sophisticated autocomplete," performing probabilistic pattern matching to predict the next token. They can imitate thought but lack underlying conceptual understanding.

Limitation 1 - Static Knowledge

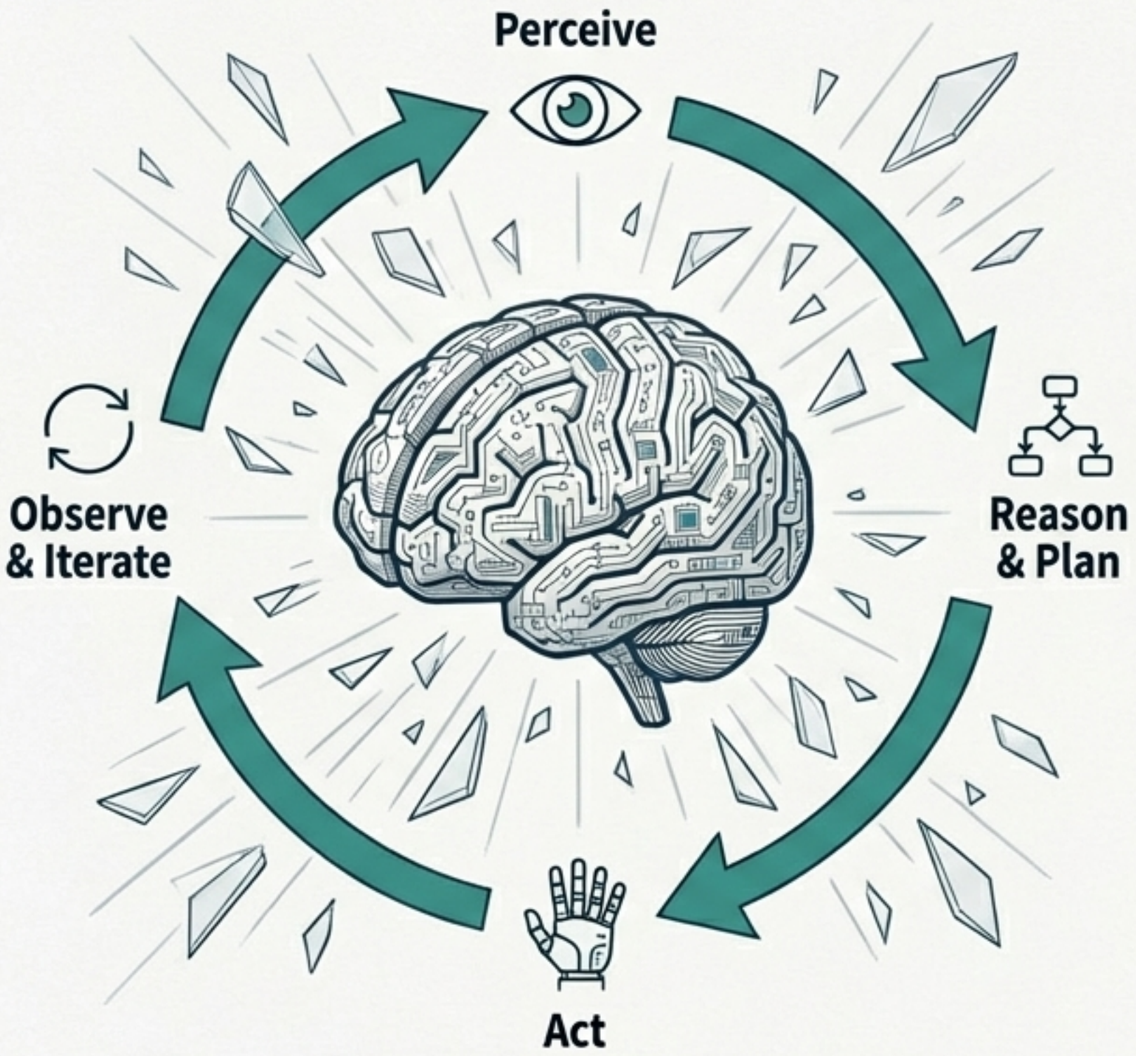
- An LLM's knowledge is frozen at the end of its training. It has no access to live, real-time information. A query like "Provide me the recent AI news" will fail because the LLM doesn't have live data access.

Limitation 2 - No Action

- A standard LLM cannot perform tasks. It cannot interact with external systems, call APIs, or affect the outside world in any way.

Limitation 3 - Finite Context

- LLMs operate with a limited short-term memory, the "context window." In long conversations, they forget earlier information, leading to inconsistencies.



Unleashing the Brain: The AI Agent

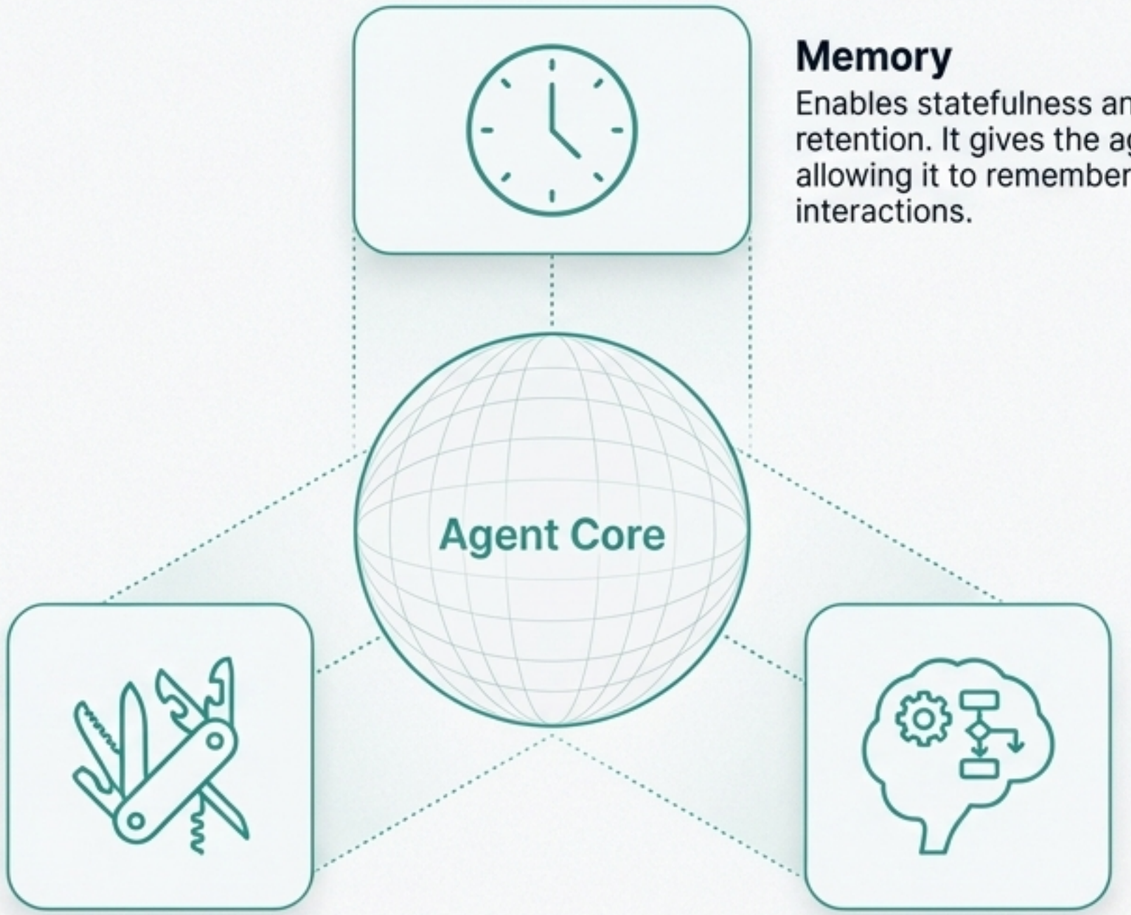
Key Message: An AI agent is a system that leverages an LLM as its core reasoning engine to autonomously achieve goals by observing its environment, planning, and executing actions using a set of tools.

Core Definition
An LLM-powered system that operates in a cycle:

- **Perceive:** Understands its environment and the user's goal.
- **Reason & Plan:** Breaks down the goal into a sequence of executable steps. The LLM is the "brain behind taking this decision."
- **Act:** Executes these steps by making a `tool_call` to an available function or API.
- **Observe & Iterate:** Analyzes the results of its actions and dynamically adjusts its plan until the goal is achieved.

Anatomy of an Agent

Every agent is constructed from a foundational triad of components: Memory, Tools, and a Planning & Reasoning Loop. These components work in concert to enable autonomous action.



Memory

Enables statefulness and context retention. It gives the agent a past, allowing it to remember previous interactions.

Tools & APIs

These are the agent's "hands." They are functions the agent can call to interact with the world, such as web search, database queries, or custom code execution.

Planning & Reasoning

The core cognitive engine, driven by the LLM. It's where the agent decides what to do next using frameworks like ReAct.

Giving Agents a Past: The Role of Memory

Key Message: Memory transforms an agent from a stateless tool into a coherent assistant capable of handling multi-step tasks and recalling previous interactions.

The Problem of Statelessness:

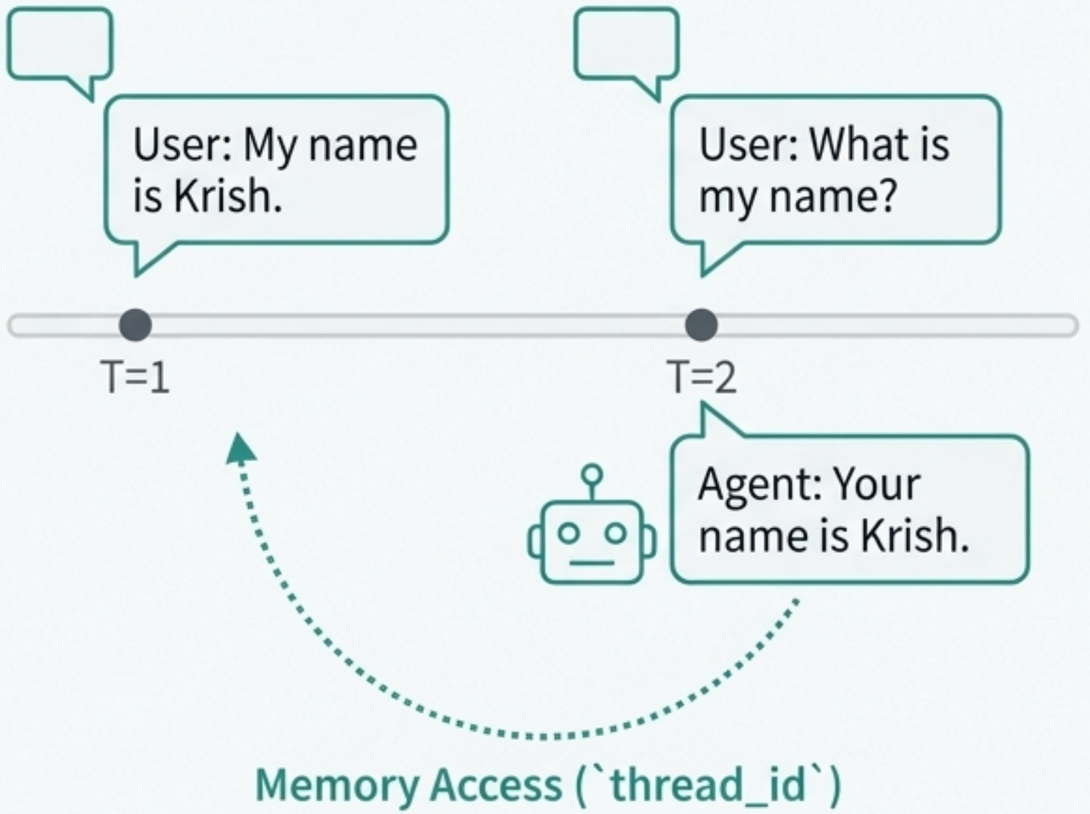
Without memory, an agent has no recollection of past events. As Krish Naik explains: “I just now told my name and... now when I’m asking the same question what is my name, it does not know.”

The Solution:

Checkpointers: Frameworks like LangGraph solve this using “memory savers” or checkpointers.

Mechanism: For each unique session, identified by a `thread_id`, the agent can persist the state of the conversation.

Quote: “LangGraph has a very special property in order to overcome this... which is called as memory.” - Krish Naik



Giving Agents Hands: The Power of Tools

Tools are what transform an agent from a thinker into a doer, allowing it to access information beyond its training data and interact with external systems.

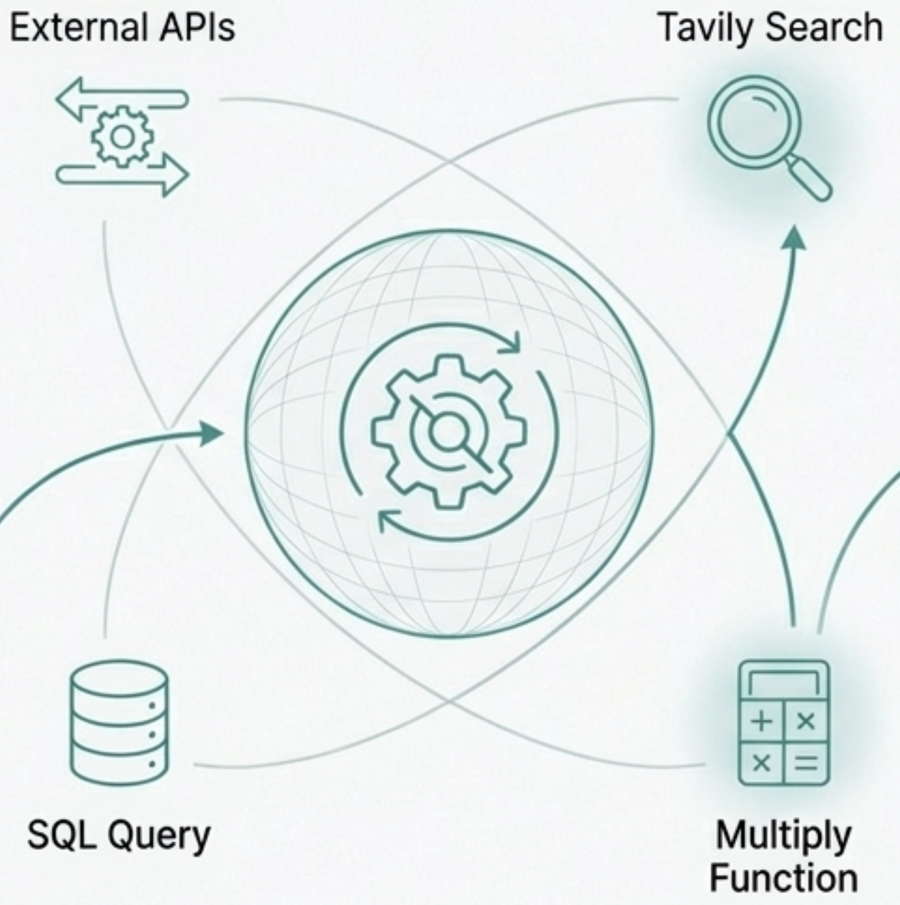
How it Works: An LLM is 'bound' with a set of predefined tools. The LLM uses the function's description or 'docstring' to understand what the tool does, its inputs, and when it should be called.

Example 1 (External Data)

A user asks: "What is the recent AI news?"

The LLM recognizes its static knowledge limit.

`tool_call` to a web search API.



Example 2 (Custom Functions):

A user asks: "What is 5 * 2?"

The LLM identifies the need for a custom math tool.

`tool_call` to `multiply(a=5, b=2)`.

Giving Agents a Mind: Planning & Reasoning

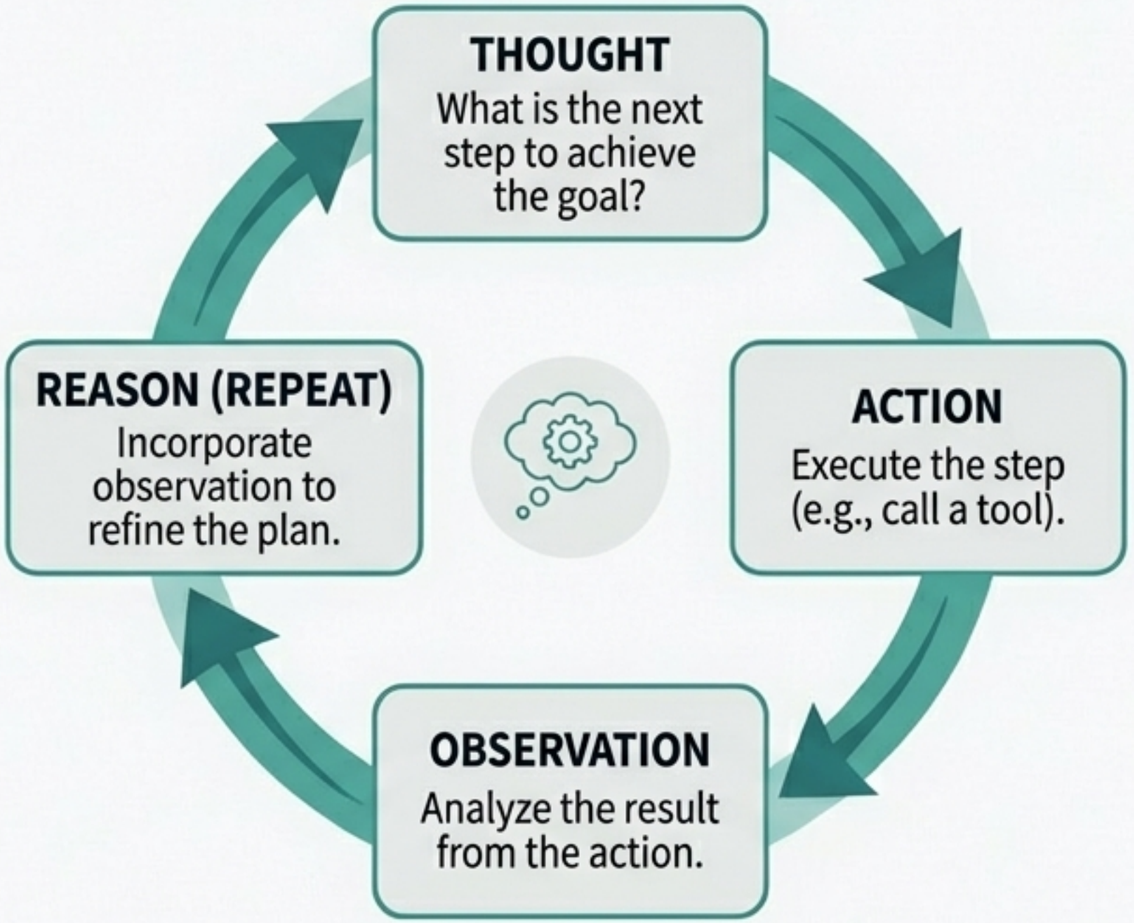
Key Message: Agents utilize sophisticated reasoning frameworks like ReAct to deliberate, act, and learn from feedback in a continuous loop until a goal is achieved. This is a shift towards ‘inference time compute.’

Inference Time Compute:

This approach “effectively tells the model to spend some time thinking before giving you an answer,” improving the quality of its reasoning steps.

The ReAct Framework: A popular and effective agent loop consisting of a repeating cycle:

1. **Reason**: The LLM thinks about the overall goal and formulates the next immediate step.
2. **Act**: The agent executes that step, typically by calling a tool with specific inputs.
3. **Observe**: The agent analyzes the output from the tool.
4. **Repeat**: The observation is fed back into the next reasoning step, refining the plan until the task is complete.



From Solo Agent to Collaborative Crew

Complex problems are often best solved by a team of specialized agents collaborating, mirroring real-world teamwork. This is a core concept behind frameworks like CrewAI.

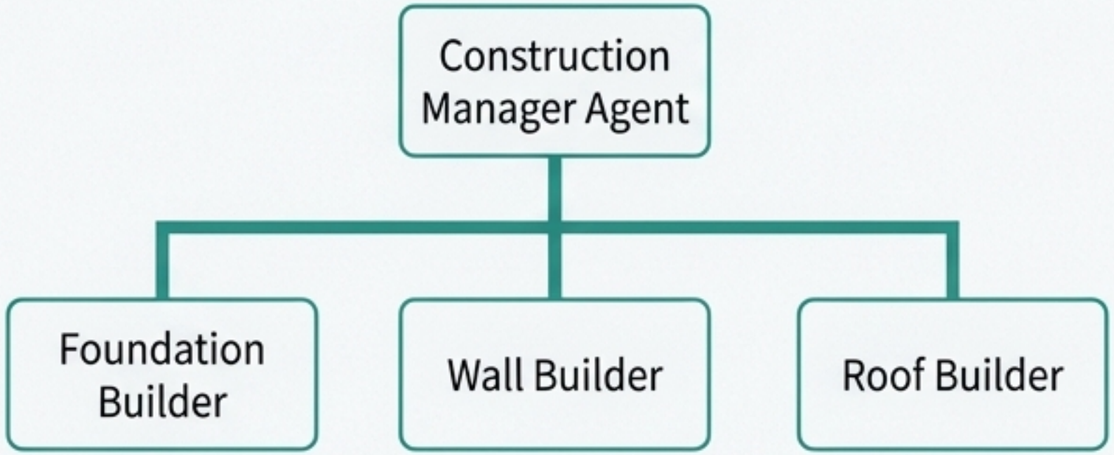
Sequential Collaboration

An “assembly line” process where tasks are executed in a fixed order. One agent's output becomes the next agent’s input.

Example: A Research agent finds data, a Writer agent drafts a blog post from that data, and an Editor agent refines the draft.

Hierarchical Collaboration

A “manager-worker” model. A manager agent oversees the goal, delegates sub-tasks to specialized agents, and synthesizes their results.



Example: A Construction Manager agent delegates tasks to a Foundation Builder, a Wall Builder, and a Roof Builder.

The Architect’s Toolkit: Frameworks for Building Agents

Open-source frameworks provide the essential abstractions, components, and observability tools required to build, debug, and deploy sophisticated AI agents efficiently.

LangChain

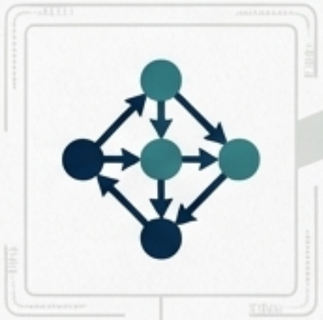
The foundational library providing the core components for building applications with LLMs, including Chains, Tools, and Memory modules.

LangSmith

The essential observability and MLOps platform. Allows developers to "debug, test, evaluate, and monitor" agent runs, critical for building production-grade applications.

LangGraph

A powerful extension for building stateful, cyclical, and multi-agent applications. It represents workflows as graphs, crucial for complex agentic systems.



CrewAI

A high-level framework specifically designed for orchestrating role-playing, autonomous multi-agent systems, simplifying the management of tasks and processes.

LangGraph: Orchestrating Complex Agentic Workflows

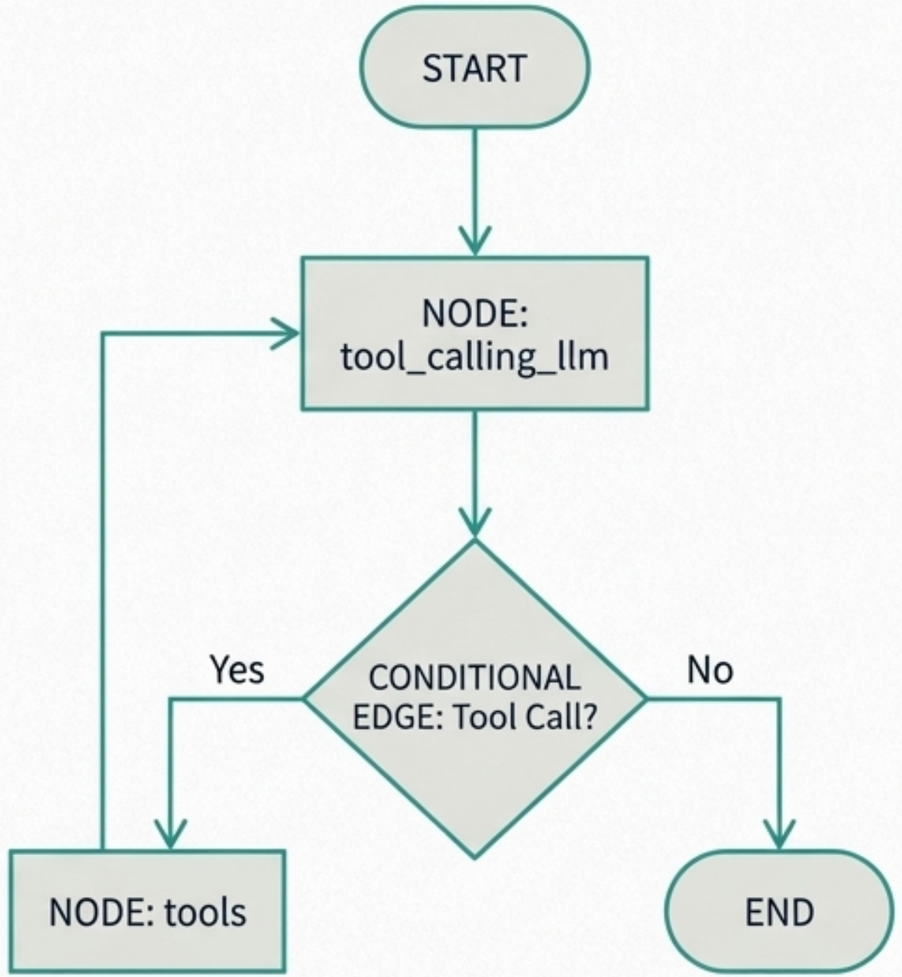
LangGraph models agent systems as state graphs, where nodes represent functions (agents or tools) and edges represent the logic that directs the flow of information, enabling complex, cyclical, and stateful behavior.

How it Works: A Three-Step Blueprint

1. **Define State**: Create a shared data structure (e.g., a `TypedDict`) that persists throughout the graph's execution. It holds variables like `messages` that any node can access and modify.

2. **Define Nodes**: Write Python functions that operate on the state. One node could be the 'tool_calling_llm," another the 'tool_node.'

3. **Define Edges**: Implement the logic that determines the next node to visit. This includes *conditional edges* that can route the workflow down different paths (e.g., 'If the last message was a tool call, go to the tool node. Otherwise, go to the end.').



Agents in the Wild: Transforming Industries

Agentic AI is moving from theoretical prototypes to practical applications that are creating real value in customer support, research, software development, and enterprise automation.

Agentic RAG

A conversational chatbot that doesn't just retrieve information but can actively search external knowledge bases (e.g., company PDFs, websites) to find and synthesize answers to user questions.

Automated Data Analysis

An agent given access to a database understands a request like “Show last quarter’s sales,” then writes and executes the necessary SQL query to generate a report.

Scientific Research

A multi-agent system where a ‘Researcher’ agent scours new papers, a ‘Summarizer’ condenses them, and a ‘Synthesizer’ agent identifies cross-paper trends.

Enterprise Automation

Agents that manage calendars, book travel, and file expense reports by interacting directly with internal company APIs, automating complex administrative tasks.

The Path Forward: Opportunities and Obstacles

As agents gain more autonomy, the industry must address critical challenges in reliability, safety, and ethics to fully realize their transformative potential.

Challenges

Hallucination & Reliability

Hallucination & Reliability: Agents can generate plausible but incorrect information or get stuck in loops. As one source notes, the model can answer "with so much confidence that if you are not aware about the truth you will be misled." Rigorous testing and observability platforms like LangSmith are essential.

Ethical Implications

Ethical Implications: Bias in training data can lead to agents performing discriminatory actions. The ability to act on information creates risks of spreading misinformation. Human oversight and ethical design are non-negotiable.

Opportunities

True Agency

True Agency: The agentic shift is pushing AI from simply reasoning about concepts to having true agency and complex decision-making capabiiities in the digital world.

Future Vision

Future Vision: The horizon includes multi-modal agents that can see, hear, and act; swarms of collaborative agents tackling complex scientific problems; and personalized agents assisting every individual.

Welcome to the Agentic Era

AI Agents represent a fundamental shift from AI as a tool for information retrieval to AI as an active partner in accomplishing complex goals.

Recapping the Ascent

1. We've moved beyond the "brain in a box," empowering LLMs with **Memory**, **Tools**, and sophisticated **Reasoning loops**.
2. We've scaled from individual agents to collaborative **Multi-Agent Systems** that can tackle complex, multi-faceted tasks.
3. Powerful frameworks like **LangGraph** and **CrewAI**, supported by observability tools like **LangSmith**, provide the architecture to build these systems today.

The future isn't about asking an AI what it knows; it's about telling it what to achieve.

Q&A

Thank You