Mastering
Agent Orchestration

Building Stateful, Cyclic, and Long-Running Agents with LangGraph

The Challenge: Linear vs. Cyclic

↓ Linear Chains (DAGs)

Traditional architectures (like LCEL) are built for **Directed Acyclic Graphs**. They excel at simple sequences: `Prompt → LLM → Parser`. Once the chain finishes, the state is lost.

Agentic Loops

Real-world agents require **reasoning loops**. The "Think, Act, Observe" pattern is inherently cyclic. Building this in linear frameworks leads to complex, brittle "spaghetti code" and manual state handling.

What is LangGraph?

Orchestration

A low-level framework designed specifically for controlling the flow of long-running, multi-step agent applications.

Stateful Graphs

Models agent behavior as a graph where nodes compute and edges control flow, similar to a Finite State Machine.

LangChain Native

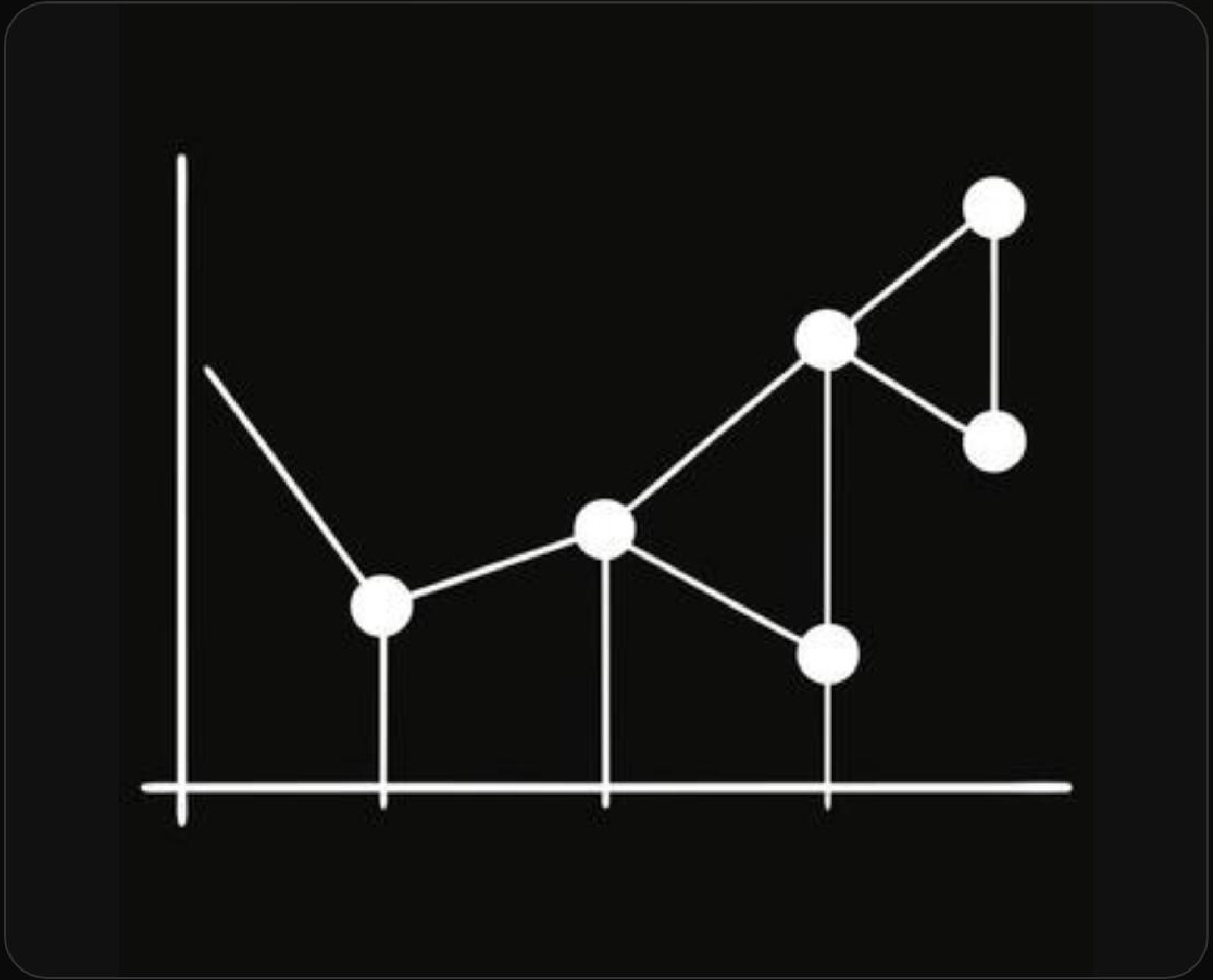
Built on top of LangChain. It acts as the "circuit board" connecting your Models, Prompts, and Tools.

Control Flow

Provides fine-grained control over loops, branching, and persistence that standard chains cannot offer.

Building Blocks: Nodes & Edges

- **Nodes (Compute):** Python functions that perform work. They receive the current `State` and return an update. Can be LLM calls or Tool invocations.
- **Edges (Flow):** define the path between nodes.
    - **Normal Edges:** Fixed transitions (A → B).
    - **Conditional Edges:** Dynamic routing based on logic (e.g., *If tool call needed, go to ToolNode*).
- **START & END:** Special nodes marking the entry and exit points of the graph execution.



The Brain: State Management

The State Object

A shared data structure (often a `TypedDict`) that serves as the **Single Source of Truth**. It persists across the entire graph execution, accessible by every node.

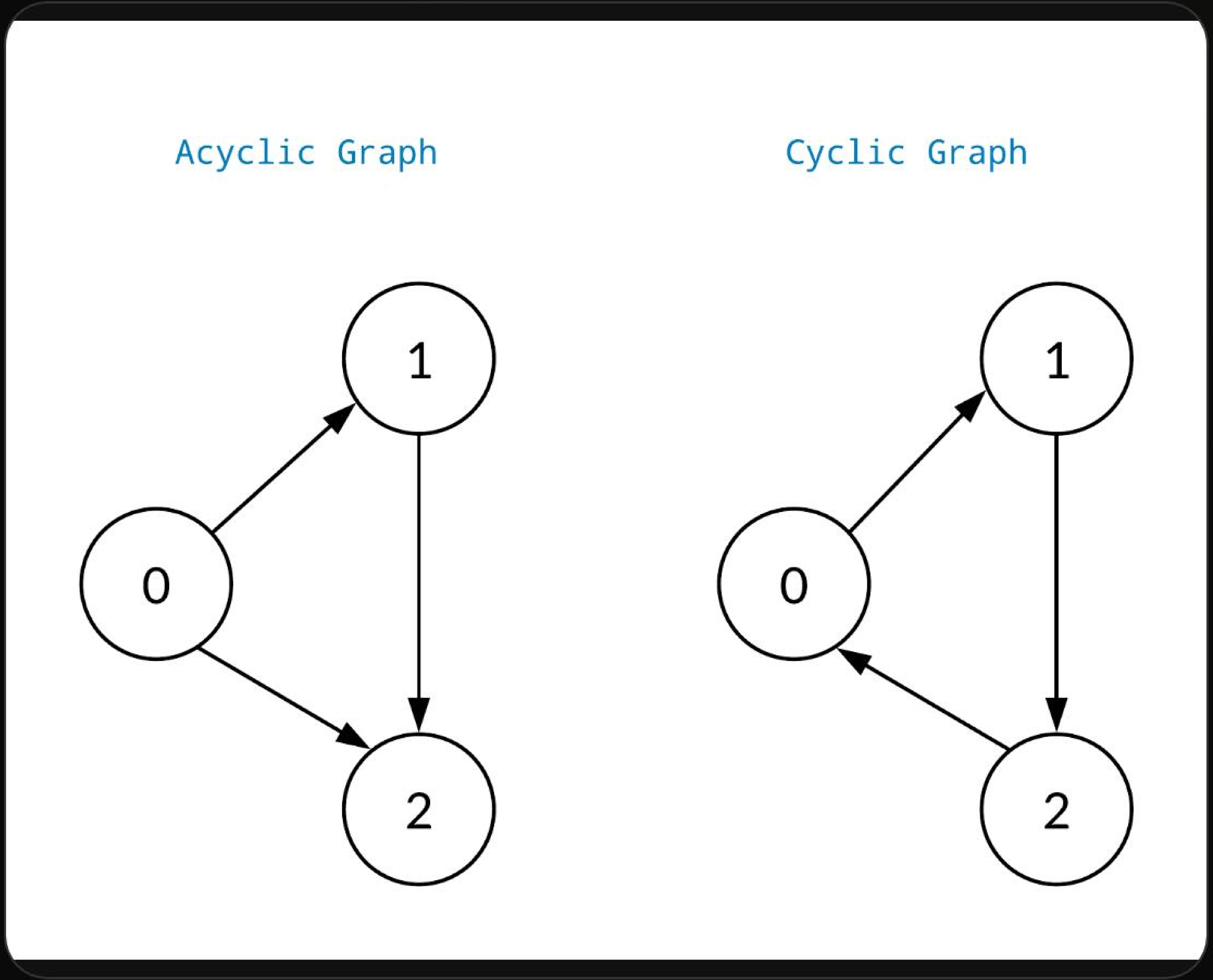
Persistence & Reducers

Reducers define how updates from nodes are merged (e.g., appending new messages). **Checkpointers** automatically save the state after every step, enabling fault tolerance and "time travel".

The Power of Cycles

Cycles are essential for autonomous agents. They allow the system to:

- ✓ **Reason & Act:** Perform a task, observe the result, and decide the next step iteratively.
- ✓ **Self-Correct:** If an output is invalid, loop back to the LLM with an error message to retry.
- ✓ **Memory:** Maintain context over long interactions via the persistent State object.



Advanced Capabilities

Human-in-the-Loop

Programmatically pause execution at specific nodes to request human approval, feedback, or edits before proceeding.

Multi-Agent Systems

Coordinate multiple specialized agents (as nodes) within a single graph, allowing for complex team-based architectures.

Streaming Support

First-class support for streaming tokens and graph updates, ensuring responsive UIs even for complex workflows.

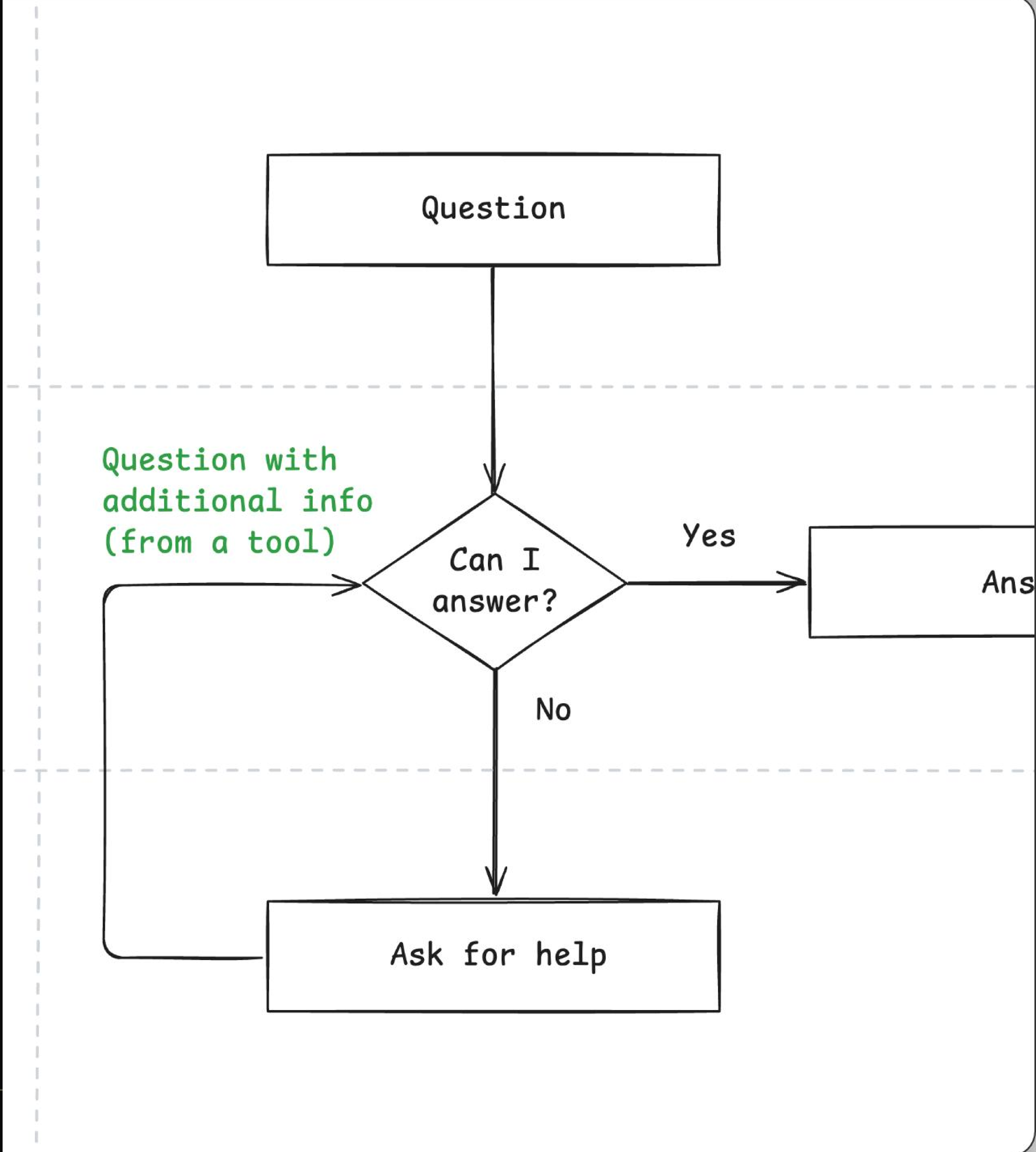
Production Ready

Designed for scale with built-in persistence layers (Postgres, Redis) to handle long-running sessions reliably.

Example: Tool Calling Agent

A standard ReAct agent loop implemented in LangGraph:

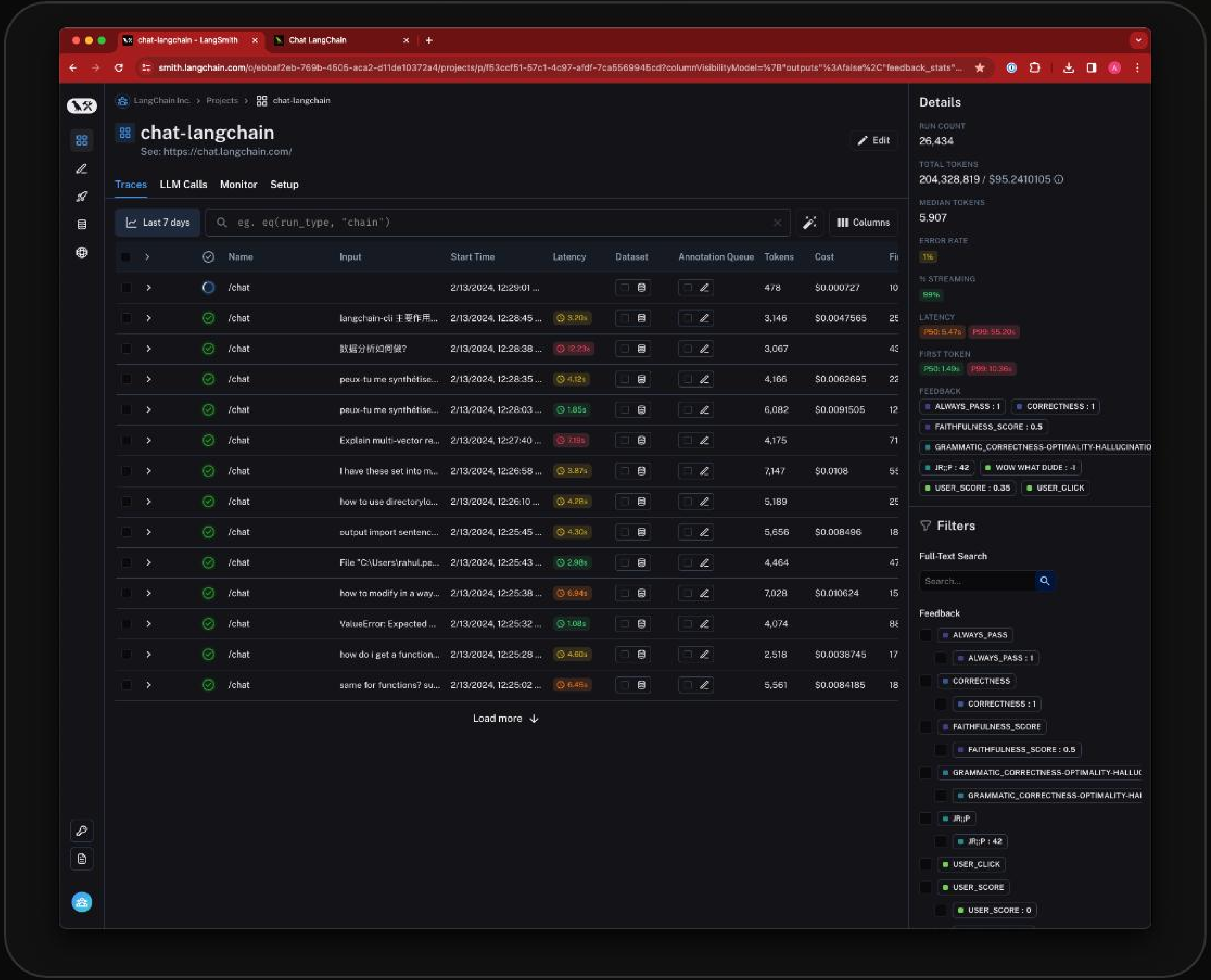
- ✓ **1. Start:** Input user query.
- ✓ **2. LLM Node:** Decides to call a tool.
- ✓ **3. Router:** Detects tool call, routes to Tool Node.
- ✓ **4. Tool Node:** Executes function, returns output.
- ✓ **5. Cycle:** Graph loops back to LLM Node to read tool output and formulate a final answer.



Observability with LangSmith

Debugging cyclic graphs can be challenging. LangGraph integrates seamlessly with LangSmith to provide:

- **Visual Traces:** See the exact path taken through the graph.
- **State Inspection:** View the State content at every step.
- **Retry & Edit:** Replay failed nodes with modified state for rapid debugging.



chat-langchain - LangSmith
Chat LangChain
smith.langchain.com/o/ebbaf2eb-769b-4505-aca2-d11de10372a4/projects/p/f53ccf51-57c1-4c97-afdf-7ca5569945cd?columnVisibilityModel=%7B"outputs"%3Afalse%2C"feedback_stats"...
LangChain Inc. › Projects › chat-langchain
Edit
Traces
LLM Calls
Monitor
Setup
Last 7 days
eg. eq(run_type, "chain")
Columns
Load more

chat-langchain

See: https://chat.langchain.com/

Name	Input	Start Time	Latency	Dataset	Annotation Queue	Tokens	Cost	Fi
/chat		2/13/2024, 12:29:01 ...				478	$0.000727	10
/chat	langchain-cli 主要作用...	2/13/2024, 12:28:45 ...	3.20s			3,146	$0.0047565	25
/chat	数据分析如何做?	2/13/2024, 12:28:38 ...	12.23s			3,067		43
/chat	peux-tu me synthétise...	2/13/2024, 12:28:35 ...	4.12s			4,166	$0.0062695	22
/chat	peux-tu me synthétise...	2/13/2024, 12:28:03 ...	1.85s			6,082	$0.0091505	12
/chat	Explain multi-vector re...	2/13/2024, 12:27:40 ...	7.18s			4,175		71
/chat	I have these set into m...	2/13/2024, 12:26:58 ...	3.87s			7,147	$0.0108	55
/chat	how to use directorylo...	2/13/2024, 12:26:10 ...	4.28s			5,189		25
/chat	output import sentenc...	2/13/2024, 12:25:45 ...	4.30s			5,656	$0.008496	18
/chat	File "C:\Users\rahul.pe...	2/13/2024, 12:25:43 ...	2.98s			4,464		47
/chat	how to modify in a way...	2/13/2024, 12:25:38 ...	6.94s			7,028	$0.010624	15
/chat	ValueError: Expected ...	2/13/2024, 12:25:32 ...	1.08s			4,074		88
/chat	how do i get a function...	2/13/2024, 12:25:28 ...	4.60s			2,518	$0.0038745	17
/chat	same for functions? su...	2/13/2024, 12:25:02 ...	6.45s			5,561	$0.0084185	18

Details

RUN COUNT
26,434

TOTAL TOKENS
204,328,819 / $95.2410105

MEDIAN TOKENS
5,907

ERROR RATE
1%

% STREAMING
89%

LATENCY
P50: 5.47s P99: 55.20s

FIRST TOKEN
P50: 1.49s P99: 10.36s

FEEDBACK
ALWAYS_PASS : 1
CORRECTNESS : 1
FAITHFULNESS_SCORE : 0.5
GRAMMATIC_CORRECTNESS-OPTIMALITY-HALLUCINATIO
JR;;P : 42
WOW WHAT DUDE : -1
USER_SCORE : 0.35
USER_CLICK

Filters

Full-Text Search

Search...

Feedback

- ALWAYS_PASS
  - ALWAYS_PASS : 1
- CORRECTNESS
  - CORRECTNESS : 1
- FAITHFULNESS_SCORE
  - FAITHFULNESS_SCORE : 0.5
- GRAMMATIC_CORRECTNESS-OPTIMALITY-HALLUC
  - GRAMMATIC_CORRECTNESS-OPTIMALITY-HAI
- JR;;P
  - JR;;P : 42
- USER_CLICK
- USER_SCORE
  - USER_SCORE : 0

Key Takeaways

Control

Move beyond rigid chains. Build flexible, custom agent architectures that fit your specific logic.

Loops

Enable true agency with first-class support for cyclic reasoning and self-correction loops.

Persistence

Fault-tolerant memory that persists across sessions, enabling long-running interactions.

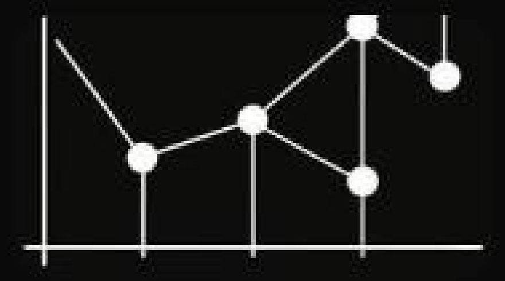
Visibility

Debug and optimize complex agent behaviors with deep LangSmith integration.

Q&A

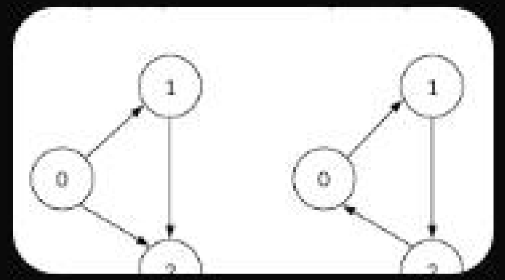
Thank you for attending.

Image Sources



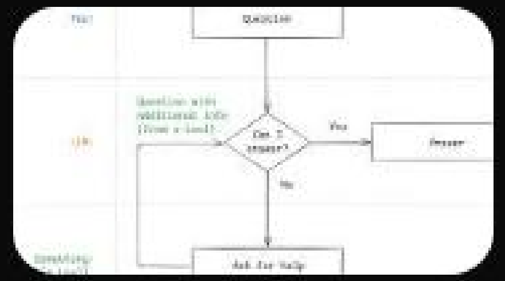
https://static.vecteezy.com/system/resources/thumbnails/054/335/838/small/network-graph-visualization-nodes-edges-data-points-trend-analysis-vector.jpg

Source: www.vecteezy.com



https://i.imgur.com/2z9J2E5.png

Source: guides.codepath.com



https://slobodan.me/images/posts/ai-agents/agents.png

Source: slobodan.me

https://mintcdn.com/langchain-5e9cc07a/H9jA2WRyA-MV4-H0/langsmith/images/project.png?fit=max&auto=format&n=H9jA2WRyA-MV4-H0&q=85&s=2426200ab2e619674636e41f11246c0d

Source: docs.langchain.com

https://miro.medium.com/v2/resize:fit:1400/1*-sBYXtudEzI5tuicj1bTyg.webp

Source: medium.com