FULL-STACK AI ORCHESTRATION

MODULAR AGENTS, OBSERVABILITY, AND THE EDGE

Jamal Sinclair O'Garro
Senior Software Engineer
Netflix
JS Nation US
November 17, 2025

Whoami

- Engineer at Netflix (Data Platform)
- Previously worked in the Netflix Infrastructure, Platform Engineering, and Data Science & Engineering organizations.
- Before Netflix, I spent my career in finance focusing on electronic and algorithmic trading on Wall Street.
- I’m a Hip-Hop head, sneaker-head, Yankees, Giants, Rangers, and Knicks fan.

Whoami

- Engineer at Netflix (Data Platform)
- Previously worked in the Netflix Infrastructure, Platform Engineering, and Data Science & Engineering organizations.
- Before Netflix, I spent my career in finance focusing on electronic and algorithmic trading on Wall Street.
- I’m a Hip-Hop head, sneaker-head, Yankees, Giants, Rangers, and Knicks fan.
- **LET’S GO KNICKS!!!**

Setting the Stage

What Is Artificial Intelligence (AI)?

artificial intelligence

[ahr-t*uh*-**fish**-*uh*l in-**tel**-i-*juh*ns] ◉ Phonetic (Standard) ○ IPA

noun, *Computers, Digital Technology.*

1 a the capacity of a computer, robot, programmed device, or software application to perform operations and tasks analogous to learning and decision making in humans, such as speech recognition or question answering. : AI, A.I.

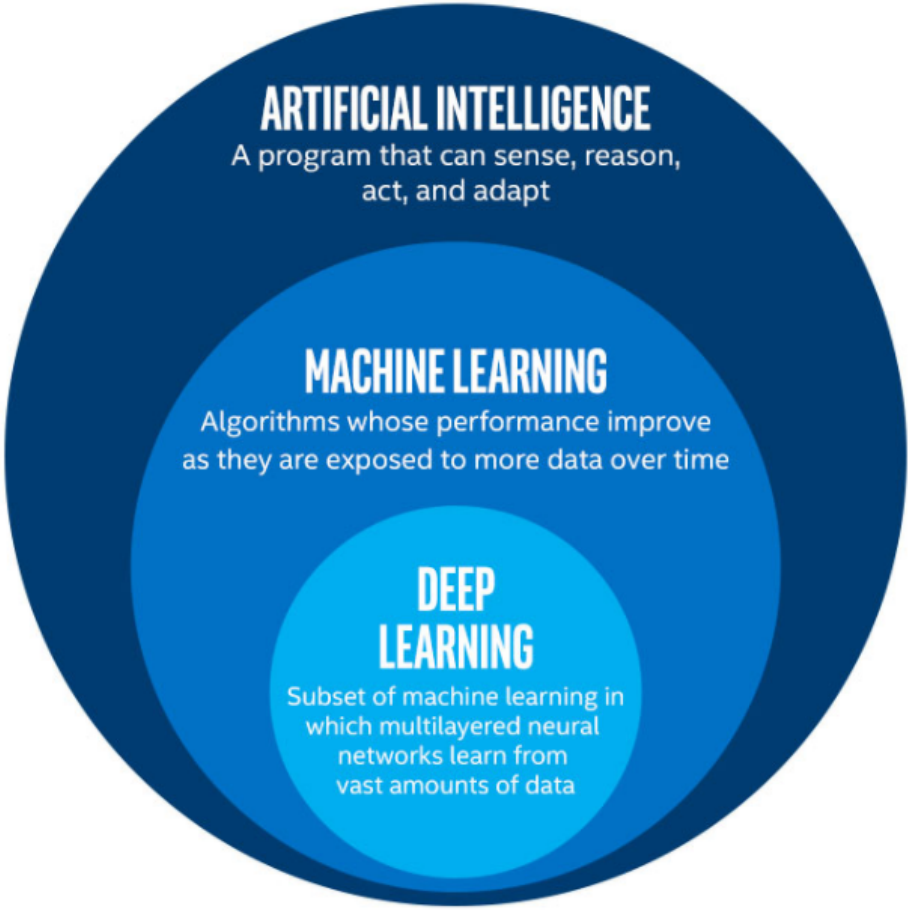
b a computer, robot, programmed device, or software application having this humanlike capacity: : AI, A.I.

> *teaching human values to artificial intelligences.*

2 the branch of computer science involved with the design of computers, robots, programmed devices, and software applications having the capacity to imitate human intelligence and thought. : AI, A.I.

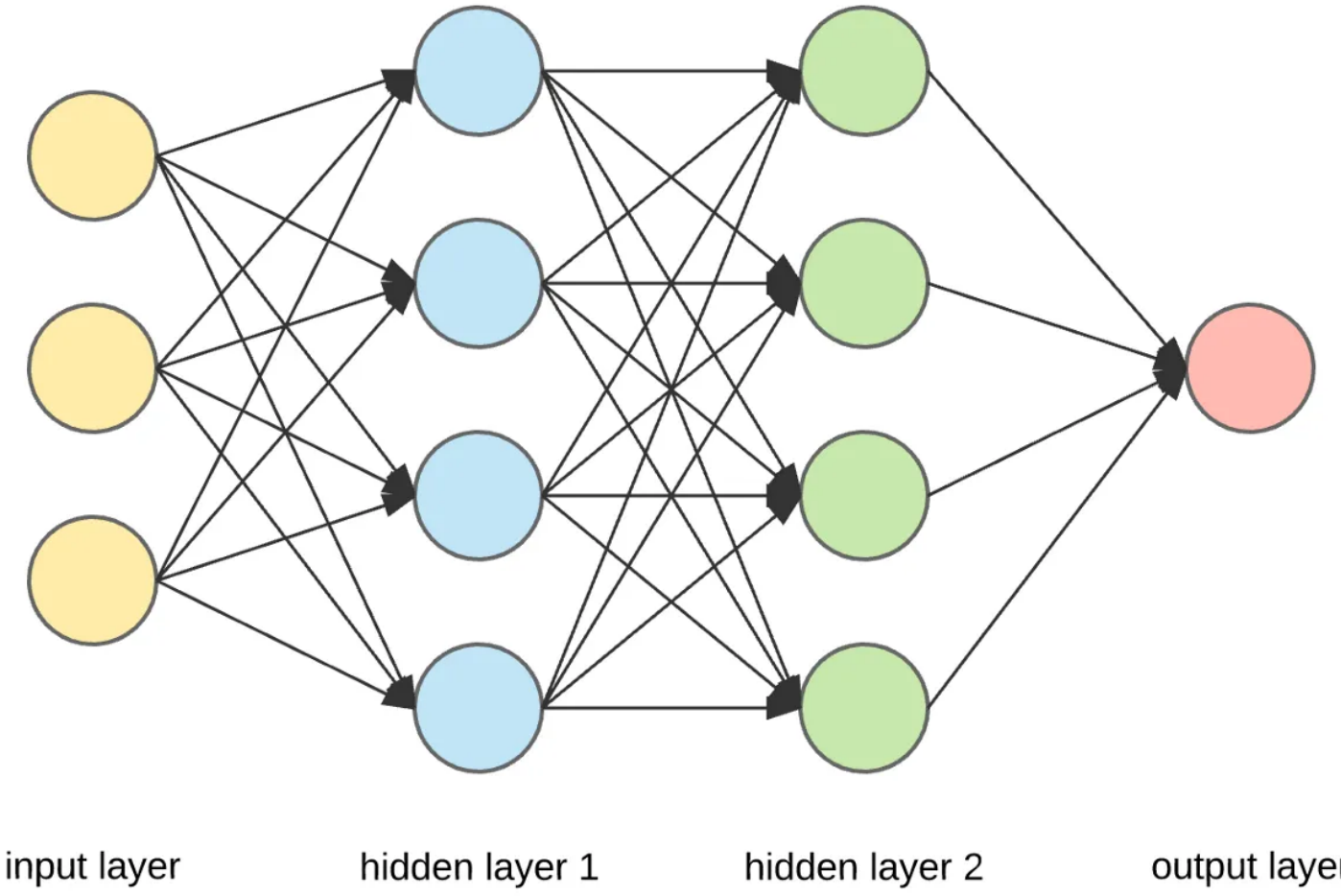
Source: https://www.dictionary.com/browse/artificial-intelligence

Areas of Artificial Intelligence

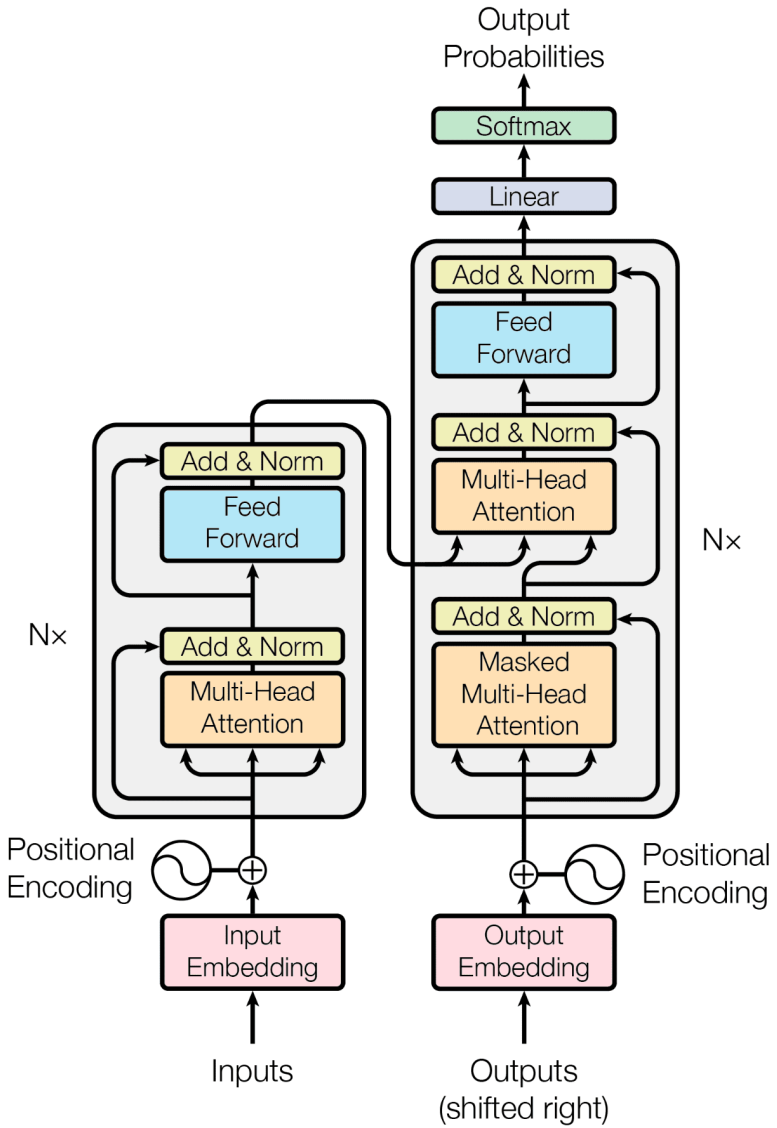


Source: https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55

Artificial Neural Networks



Transformers



Source: "Attention is All You Need", Vaswani et al

Large Language Models (LLMs)

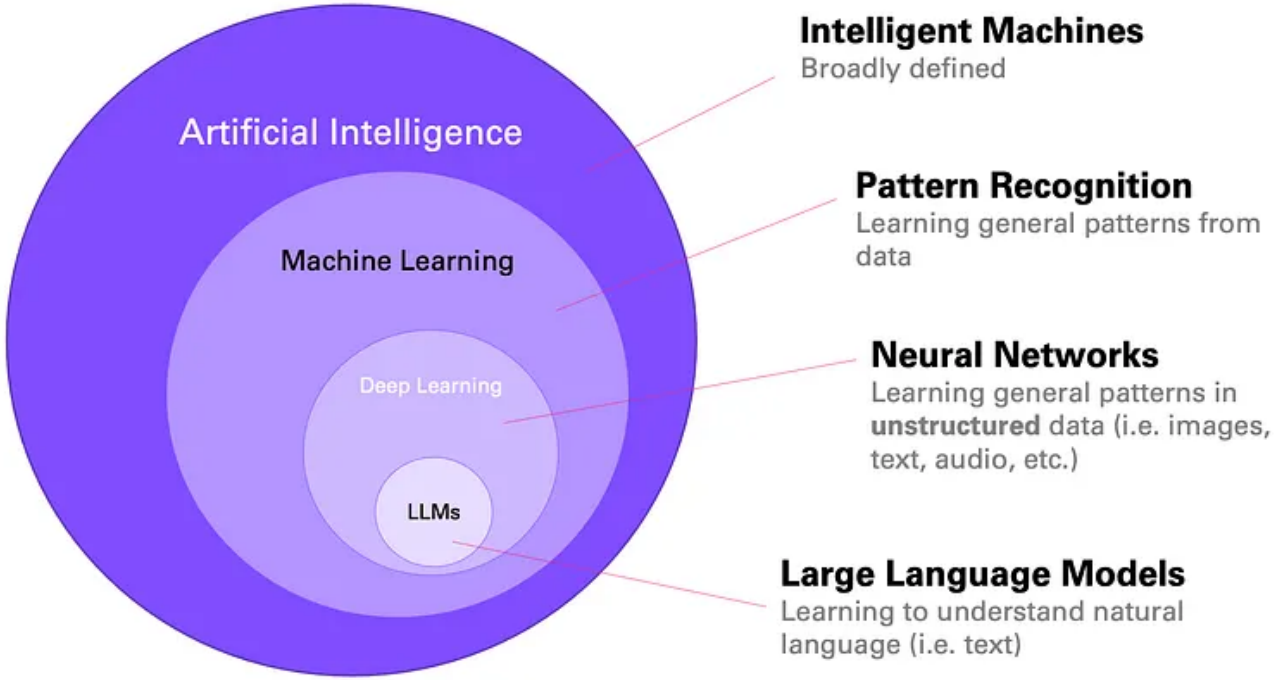
ChatGPT

Claude

deepseek

Gemini

LLMs in the AI Spectrum



Source: https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Enter Agents

AI Agents

- Intelligent software that can interpret natural language, make decisions, interact with APIs and tools, and generate responses or actions based on context.
- Built using LLMs or other agents (multi-agent systems).
- Main capabilities:
  - Tool calling
  - Memory
  - Planning and reasoning capabilities

Agentic Systems

- A software system powered by or integrated with AI agents.
- Agentic systems differ from traditional AI systems by allowing the LLM to receive feedback and decide its own actions directly.
- This allows the application flow to be dynamically inferred in real-time.

Source: https://medium.com/data-science-at-microsoft/how-large-language-models-work-91c362f5b78f

Building Agentic AI Systems for Production

Question: Show of Hands

- How many people have **used** an LLM-powered application?

Question: Show of Hands

- How many people have **used** an LLM-powered application?
- How many people in the audience have **built** an LLM-powered application?

Question: Show of Hands

- How many people have **used** an LLM-powered application?
- How many people in the audience have **built** an LLM-powered application?
- How many people have **deployed** one of these systems and had it run reliably for months?

Building Agentic AI Systems Is Hard!

But Why?

The Challenges of Building Agentic AI Systems

- It’s no longer just a model-training and deployment problem.
- AI Engineering evolved from simple, stateless request-response models to complex, stateful, and **autonomous agents.**
- Becomes a distributed systems orchestration problem.

What Could Possibly Go Wrong?

- Requests to LLM providers can timeout or fail, becoming a point of failure in the prompt/response lifecycle.
- Model behavior is unpredictable, making debugging difficult.
- LLMs can hallucinate answers, degrading the user experience.
- Token usage can become expensive if not managed properly.
- How can we make our systems resilient under these circumstances?

Solution: A Modern Enterprise AI Stack

- Successfully deploying AI solutions requires robust, scalable, and efficient architectures.

Solution: A Modern Enterprise AI Stack

- Successfully deploying AI solutions requires robust, scalable, and efficient architectures.
- We need systems that handle retries, durability, state management, and error handling.

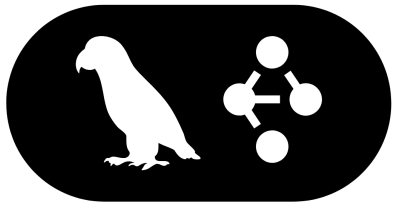
Solution: A Modern Enterprise AI Stack

- Successfully deploying AI solutions requires robust, scalable, and efficient architectures.
- We need systems that handle retries, durability, state management, and error handling.
- A Modern Stack (The “Glue”)
  - **LangGraph:** Provides the "glue" to connect modern enterprise agent patterns.
  - **LangSmith:** Makes observability and evaluation first-class citizens.
  - **Temporal.io:** Empowers full-stack applications with stateful orchestration and durable execution.

The AI Agent as a Modular Monolith

- In this system, the AI agent is a **"Modular Monolith"**—a single, deployable service built from composable, testable, graph-based components.
- Emphasizes separating functionality into independent, interchangeable modules.
- This modular architecture allows complex problems to be divided into tractable units of work, each targeted by specialized agents.
- **LangGraph** is a low-level, customizable agent framework built by LangChain that enables the creation of these systems.

LangGraph

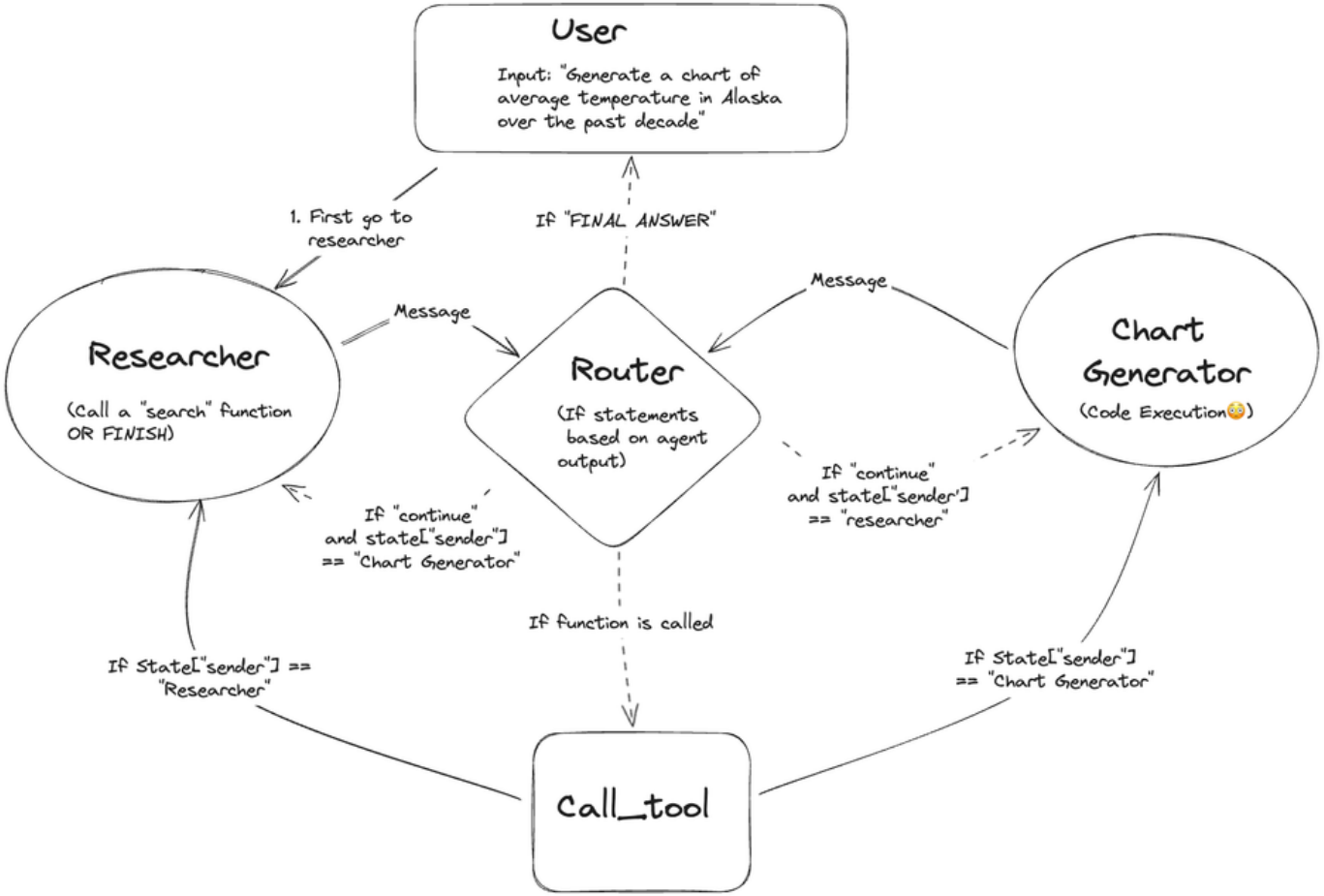


LangGraph

LangGraph

- LangGraph uses a graph structure to define, coordinate, and execute multiple LLM agents (or chains) in a structured and efficient manner.
- Core Components:
  - **State:** A shared data structure that represents the current snapshot of your application. It can be any data type, but is typically defined using a shared state schema.
  - **Nodes:** Functions that encode the logic of your agents. They receive the current state as input, perform some computation or side-effect, and return an updated state.
  - **Edges:** Functions that determine which node to execute next based on the current state. They can be conditional branches or fixed transitions.
- **Cycles:** The graph structure explicitly supports **cycles**, which are a critical component of most agent runtimes and represent reasoning loops

The Agentic Reasoning Loop



https://blog.langchain.com/langgraph-multi-agent-workflows/

Benefits of this Approach

- **Modularity:** Each node in the graph is independent and testable.
- **Flexibility:** Enables conditional routing based on state.
- **Reusability:** Nodes can be reused across different agents.
- **Observability**: Have clear visibility into each step of our agents' workflow.

LangSmith

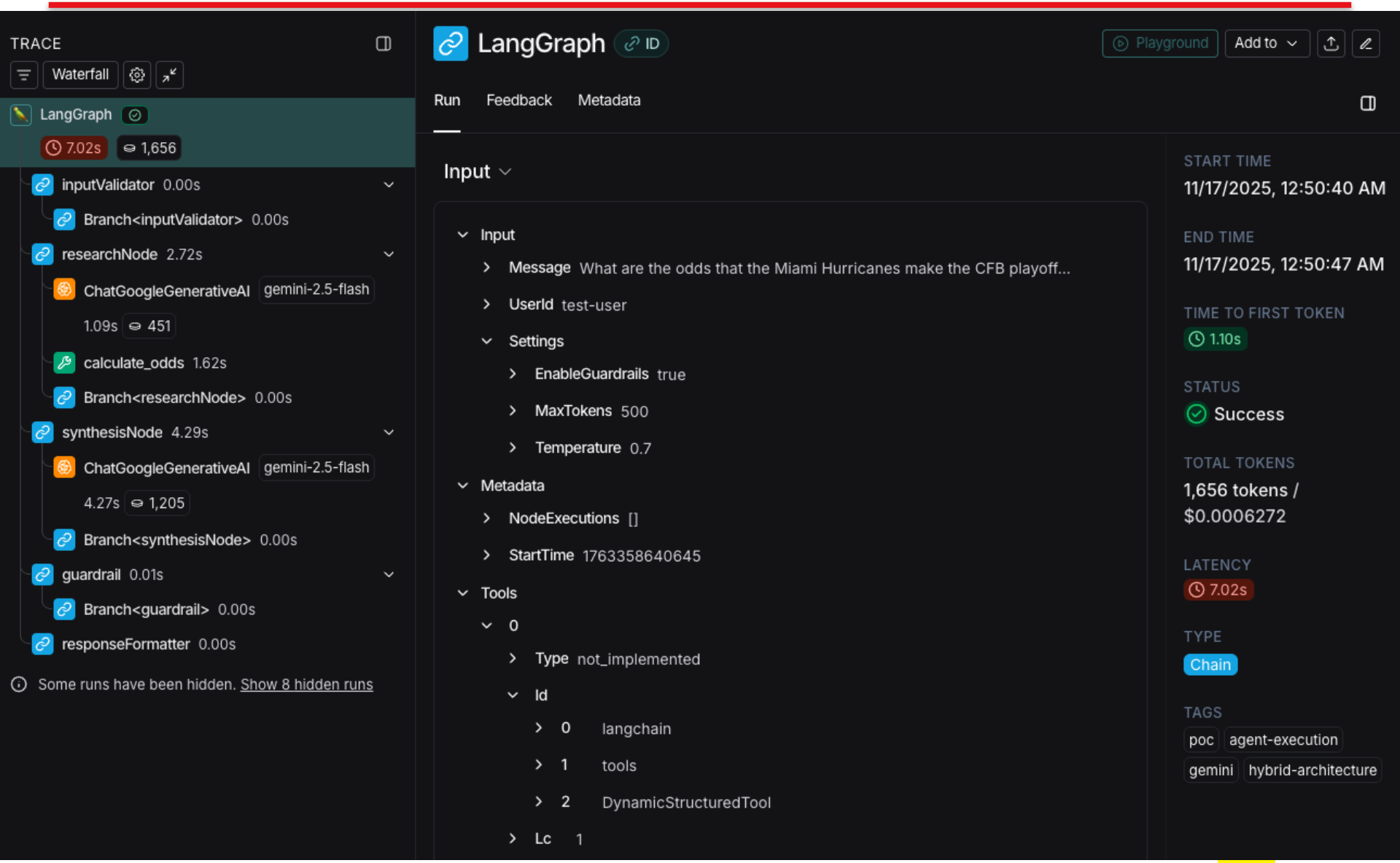
Observability as a First-Class Citizen

- **Observability** is a critical requirement because LLMs are non-deterministic; the same prompt can produce different responses, making debugging complex.
- The explicit graph structure of LangGraph is inherently **optimized for observability.**
- **LangSmith** is the specialized platform to test, debug, and evaluate LLM applications, enhancing quality assurance.

End-to-End Tracing with LangSmith

- A **trace** records the sequence of steps an application takes from input to output, capturing the full record of what happened.
- **Key Tracing Features:**
  - **Visibility:** Provides end-to-end visibility into the agent's complex behaviors, allowing developers to inspect, debug, and validate execution.
  - **Trace Management:** Enables filtering traces, querying traces via SDK, comparing different trace results, and sharing traces publicly.
  - **Cost Analysis:** Can calculate **token-based costs** for traces, linking performance directly to expenditure.
  - **Projects and Threads:** Traces are grouped into **Projects** (containers for applications) and **Threads** (sequences of traces for multi-turn conversations).

LangSmith Trace

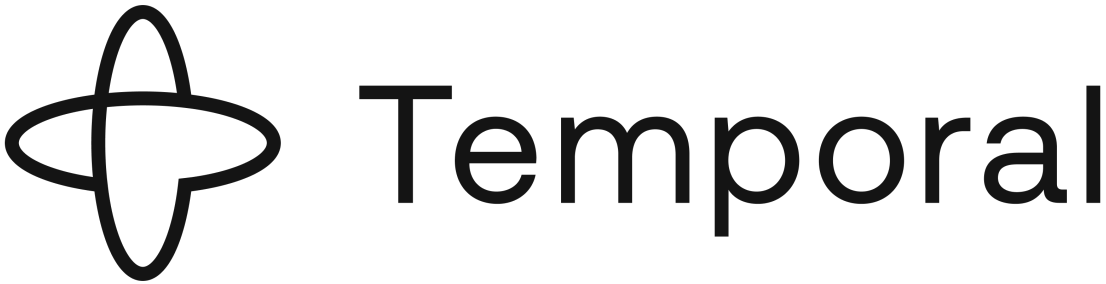


When Things Flip Upside Down

When Things Flip Upside Down

- What happens when your LLM API call fails after 15 seconds?
- Work done up to that point is lost.
- No retries.
- State is not persisted.
- Have to invoke the LLM again and waste tokens.

Temporal



Guaranteed Execution for AI Agents

- AI agent workflows are complex distributed systems that must handle long-running operations and are prone to failures from flaky tools, API rate limits, and complexity of state management.
- **Temporal.io** is a durable workflow platform that ensures applications run reliably, providing **built-in tools** for state management, automatic retries, durability, and scalability.
- Temporal uses **durable execution** to guarantee that complex workflows —even those whose paths are dynamically decided by an LLM—run to completion, even through infrastructure failures.

Temporal Workflows and Activities

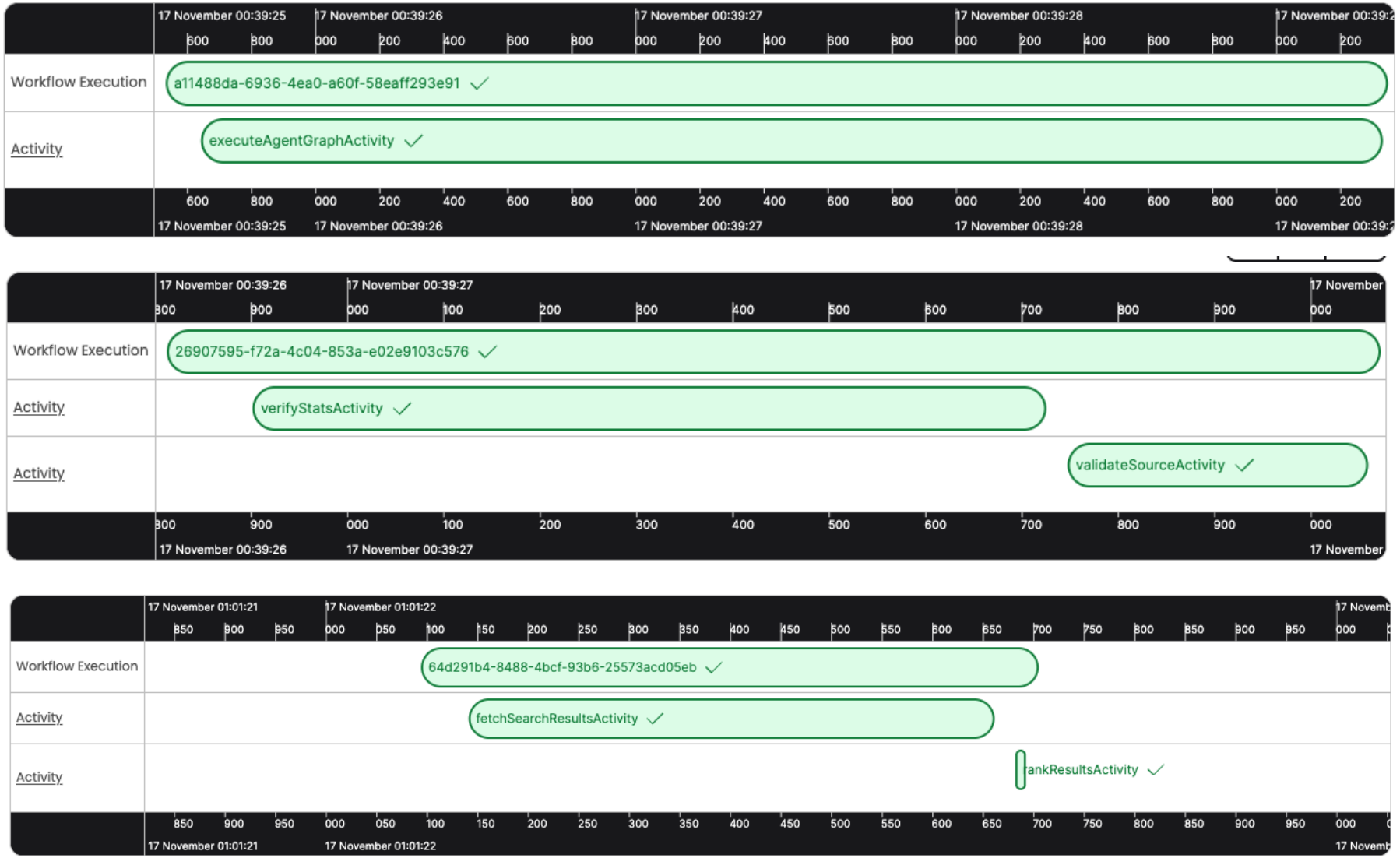
- **Workflow:** Orchestration of work to be done in the system.
- **Activity:** A deterministic unit of work that causes a side effect.
  - e.g., writing to a table, submitting an API request.
- **Integration with LangGraph:**
  - Each graph is run in a durable workflow.
  - Tools available to our agents are durable workflows and activities.
- **Two layers of Durability:**
  - Graph level
  - Tool calling level

Temporal Workflows

Status	Workflow ID	Run ID	Type
Completed	db463a44-add9-44c1-aed0-8df9c2b2dc50	a11488da-6936-4ea0-a60f-58eaff293e91	agentOrchestrationWorkflow
Completed	stats-verify-1763357966810-aczjcjewn	26907595-f72a-4c04-853a-e02e9103c576	statsVerificationWorkflow
Completed	f048eb50-2b73-434c-9bb3-e9392d906d32	71dfc72a-b9a7-4b22-8106-c27623248df6	agentOrchestrationWorkflow
Completed	9f9b770c-808a-4dda-91d1-14e262c71ced	f51cdf7b-41c9-4241-b6c6-d7615f5a8e2d	agentOrchestrationWorkflow
Completed	c6c11da3-07f3-4642-bb17-30a16a178108	135be17b-c455-4788-ade1-ac9255e52a54	agentOrchestrationWorkflow
Completed	5fb1251e-93cc-49e7-a20e-5e7064d7b5f9	012f1214-0d37-4496-bd94-9b754a7ce7ab	agentOrchestrationWorkflow
Completed	05e638f3-ab89-438e-b50b-c872dd78ce80	17cd9ea1-0acd-4b1d-b232-b1cc5348be3a	agentOrchestrationWorkflow
Completed	a3ecbd62-2cf2-4e38-a023-fc748d2f8c72	dc2131a5-5d6f-4141-8d7c-85e12bbc9374	agentWorkflow
Completed	35546b0f-47f0-4e7b-99f7-d5864f059161	d9f1bf20-77ee-45e6-9320-26ae4a81c4b9	agentWorkflow
Completed	1947cd96-dc14-42ca-af62-59dbcbfefcb5	5c024cb0-60ae-4497-801c-06c6c6f7af06	agentWorkflow
Completed	ed68ca84-2f02-4e79-9323-03abecf2cf87	8803b218-172c-423b-8247-f94886bcf9c6	agentWorkflow

100 1

Temporal Activities



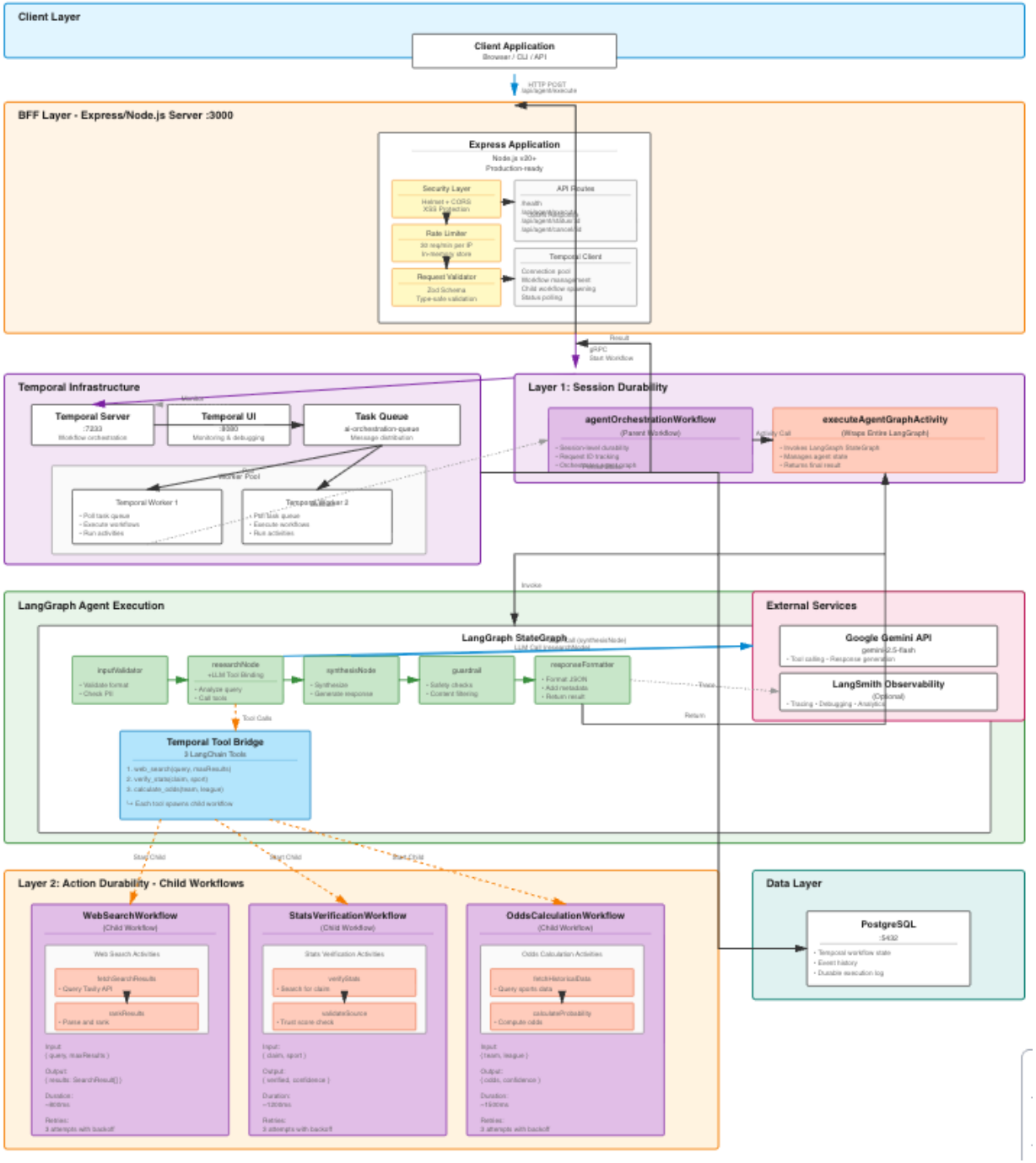
Temporal Inputs and Results

Benefits of this Approach

- **Durability:** AI workflows can survive crashes.
- **Automatic Retries:** Configurable retry policies.
- **Observability:** Preserve the full execution history of agent runs and tool calls.
- **Scaling:** Multiple worker processes run in parallel.

A BFF for Agent Orchestration

- Deploying the stateful AI agent as an **Intelligent Backend-for-Frontend (BFF)** is a recommended architectural pattern.
- This pattern involves creating a separate backend service tailored for specific frontend applications or interfaces (e.g., desktop or mobile).
- Agentic AI system can evolve separately from the front-end.
- Leverages stateful orchestration (via Temporal).
- All components can be containerized and composed together.


Client Layer
Client Application
Browser / CLI / API
HTTP POST
/api/agent/execute
BFF Layer - Express/Node.js Server :3000
Express Application
Node.js v20+
Production-ready
Security Layer
Helmet + CORS
XSS Protection
Rate Limiter
Request Validator
Zod Schema
Type-safe validation
API Routes
/health
/api/agent/execute
/api/agent/status/:id
/api/agent/cancel/:id
Temporal Client
Connection pool
Workflow management
Child workflow spawning
Status polling
Result
gRPC
Start Workflow
Temporal Infrastructure
Temporal Server
:7233
Workflow orchestration
Temporal UI
:8233
Monitoring & debugging
Task Queue
ai-orchestration-queue
Message distribution
Monitor
Worker Pool
Temporal Worker 1
- Poll task queue
- Execute workflows
- Run activities
Temporal Worker 2
- Poll task queue
- Execute workflows
- Run activities
Layer 1: Session Durability
agentOrchestrationWorkflow
(Parent Workflow)
- Session-level durability
- Request ID tracking
- Orchestrates LangGraph
Activity Call
executeAgentGraphActivity
(Wraps Entire LangGraph)
- Invokes LangGraph StateGraph
- Manages agent state
- Returns final result
Invoke
LangGraph Agent Execution
LangGraph StateGraph
LLM Call (synthesisNode)
LLM Call (researchNode)
inputValidator
- Validate format
- Check PII
researchNode
+LLM Tool Binding
- Analyze query
- Call tools
synthesisNode
- Synthesize
- Generate response
guardrail
- Safety checks
- Content filtering
responseFormatter
- Format JSON
- Add metadata
- Return result
Tool Calls
Temporal Tool Bridge
3 LangChain Tools
1. web_search(query, maxResults)
2. verify_stats(claim, sport)
3. calculate_odds(team, league)
↳ Each tool spawns child workflow
Traces
Return
External Services
Google Gemini API
gemini-2.5-flash
- Tool calling - Response generation
LangSmith Observability
(Optional)
- Tracing - Debugging - Analytics
Start Child
Start Child
Start Child
Layer 2: Action Durability - Child Workflows
WebSearchWorkflow
(Child Workflow)
Web Search Activities
fetchSearchResults
- Query Tavily API
rankResults
- Parse and rank
Input:
{ query, maxResults }
Output:
{ results: SearchResult[] }
Duration:
~800ms
Retries:
3 attempts with backoff
StatsVerificationWorkflow
(Child Workflow)
Stats Verification Activities
verifyStats
- Search for claim
validateSource
- Trust score check
Input:
{ claim, sport }
Output:
{ verified, confidence }
Duration:
~1200ms
Retries:
3 attempts with backoff
OddsCalculationWorkflow
(Child Workflow)
Odds Calculation Activities
fetchHistoricalData
- Query sports data
calculateProbability
- Compute odds
Input:
{ team, league }
Output:
{ odds, confidence }
Duration:
~1500ms
Retries:
3 attempts with backoff
Data Layer
PostgreSQL
:5432
- Temporal workflow state
- Event history
- Durable execution log

DEMO TIME!

Other Enhancements to Consider

- **Complex Agent Architectures:** Agent supervisor, hierarchical agent teams, multiple independent agent patterns.
- **Dynamic LLM Selection:** dynamically select an LLM based on the task.
  - E.g., a larger model for more complex tasks and a smaller model for simpler tasks.
- **Token Optimization:** Have agents keep track of their context windows, write to the shared state, and compact conversation when context grows.
- **Cost Optimization:** Agents dynamically shift their behavior to monitor and optimize costs associated with token usage.
- **Security and Governance:** Implement robust frameworks to protect AI models and data.
- **Scaling and Optimization:** Autoscaling configurations must handle the variable demands of AI workloads.

Key Takeaways

- When using LLMs in production, a number of things can go wrong.
- Systems have to be resilient to failures.
- Agent behavior needs to be traceable and debuggable.
- Agent sessions and tool calls must be durable.
- Deploying and scaling Agentic AI systems is a distributed systems orchestration problem.
- Agent-building tools like LangGraph, coupled with observability tools like LangSmith and orchestration tools like Temporal, can help us build robust, production-grade AI systems for the enterprise.

THANKS

Contact

- Github: @jsogarro
- Twitter: @jsogarro
- LinkedIn: /in/jsogarro/

- Sample code: https://github.com/jsogarro/modular_agents_poc

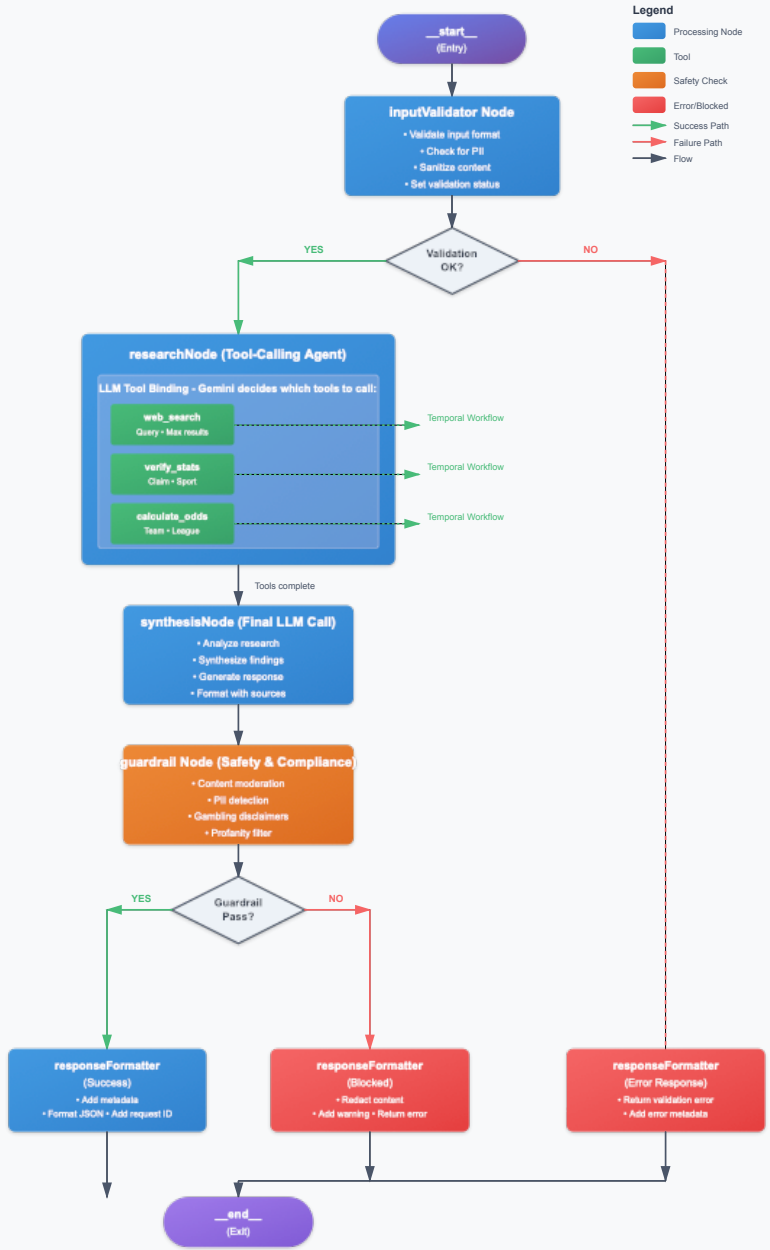

SCOOPS
AHOY
ICE CREAM PARLOR

Using the Graph for Complex AI Logic

- LangGraph allows the LLM to dynamically **control what happens next**
- Supported Workflow Patterns:
  - **Prompt Chaining:** Multiple LLM calls where the output of one process becomes the input for the next.
  - **Parallelization:** Fanning out tasks so independent processes (e.g., performing research, requesting data) run in parallel.
  - **Routing:** Using a conditional edge based on a structured LLM output (the router) to decide which path (research, fetch data) to take.
  - **Orchestrator/Worker:** An LLM planner dynamically generates a list of subtasks (e.g., report sections) and assigns them to independent workers, which then write back to the shared state.

LangGraph State Graph Structure

Agentic Workflow with Cycles & Tools



Framework: LangGraph (StateGraph) • LLM: Google Gemini 2.5 Flash • Tool Integration: Temporal Child Workflows

Generated: 2025-11-17