Istanbul Technical University | 2025

LangGraph

Presentation by Abdulkadir Külçe

LangChain Ecosystem

LangChain

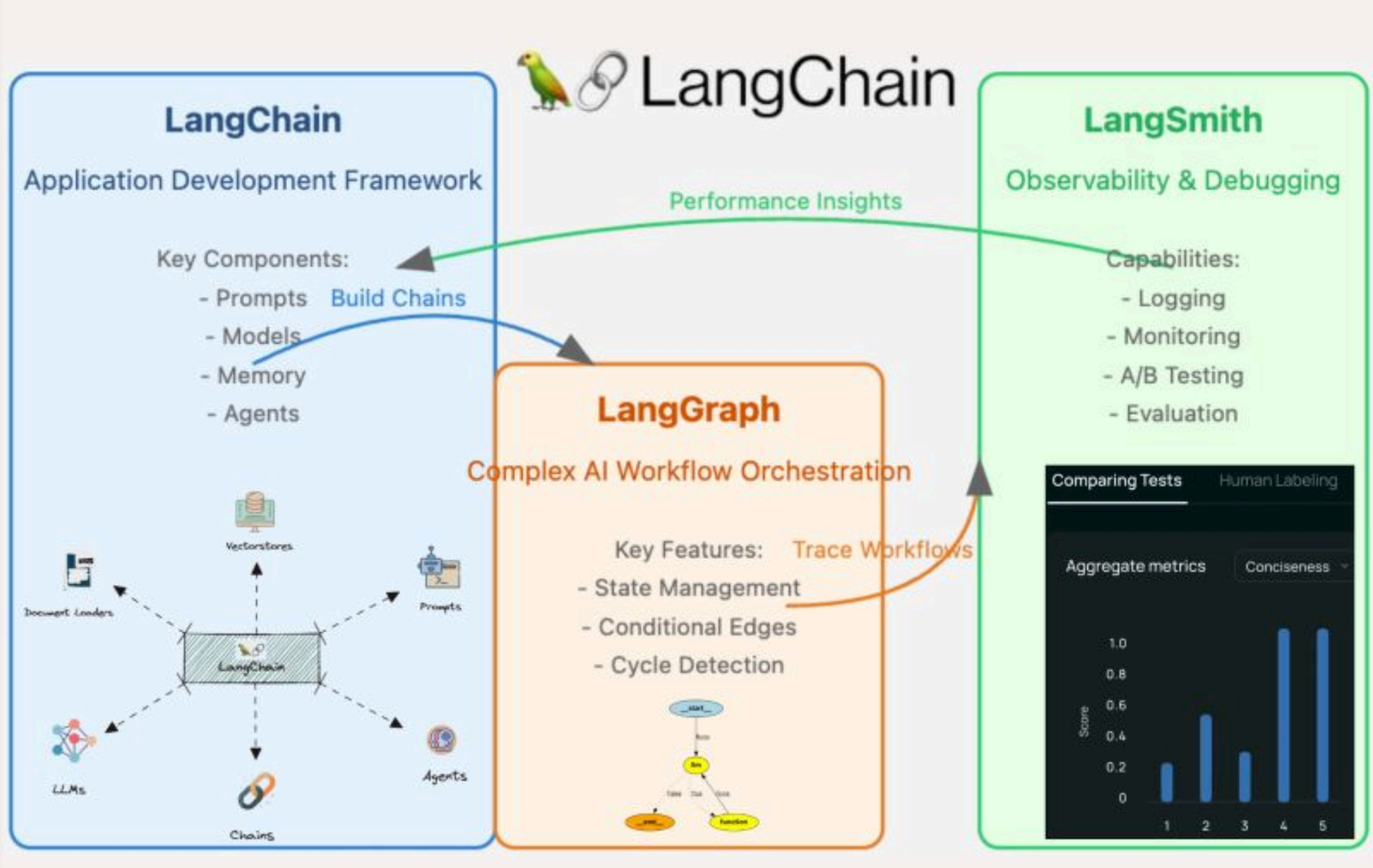
A framework for building production-grade applications that integrate language models with data sources, APIs, and custom logic through modular, composable components.

LangGraph

A framework for building reliable, stateful agentic systems with explicit control flow, built-in persistence, and support for human-in-the-loop workflows.

LangSmith

A platform for tracing, evaluating, and optimizing LLM applications with production monitoring, dataset management, and automated quality assurance.



LangChain, LangGraph & LangSmith

What is LangGraph?

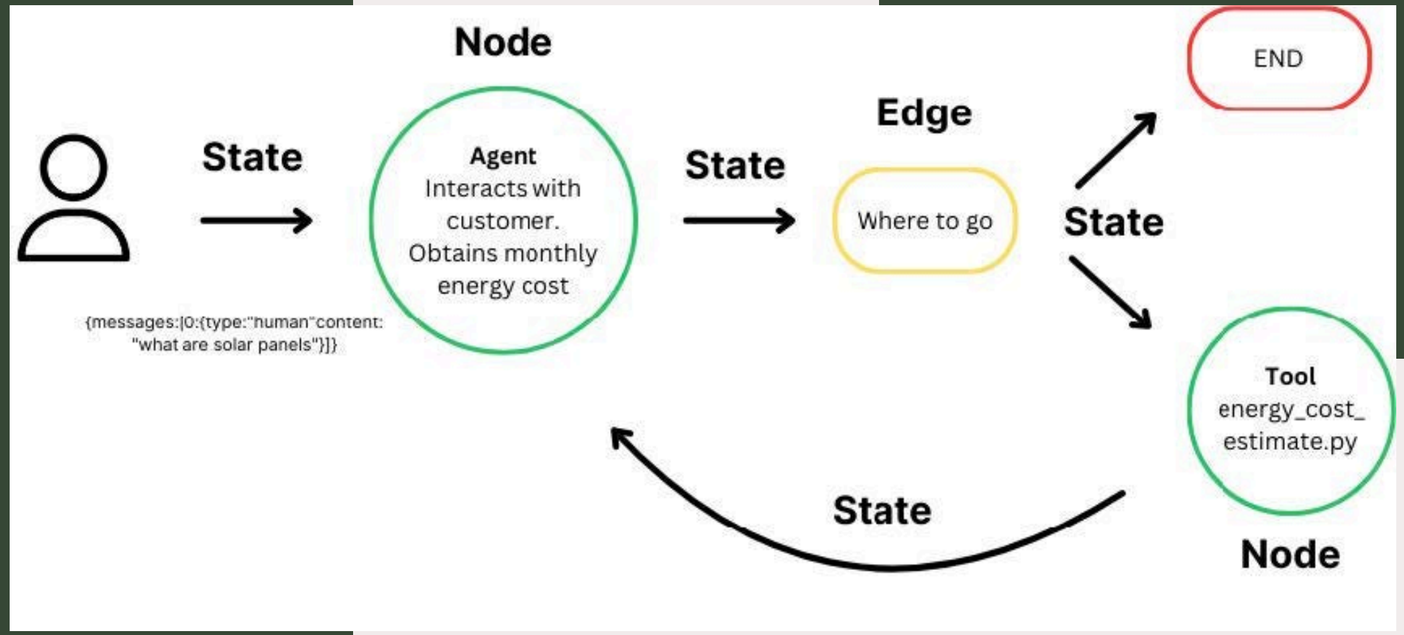
Low-level orchestration framework designed to build, manage, and deploy long-running and stateful agents.

The 3 Core Concepts of LangGraph

1 - Node = A Task

2 - Edge = What Comes Next?

3 - State = The Shared Memory



Why is it needed?

- **Linear Chains**
- **Standardized Usage of Different LLMs**
- **Prompt Specification**

- **AgentExecutor too Rigid**
- **BlackBox**
- **Hard to Debug when Fails**

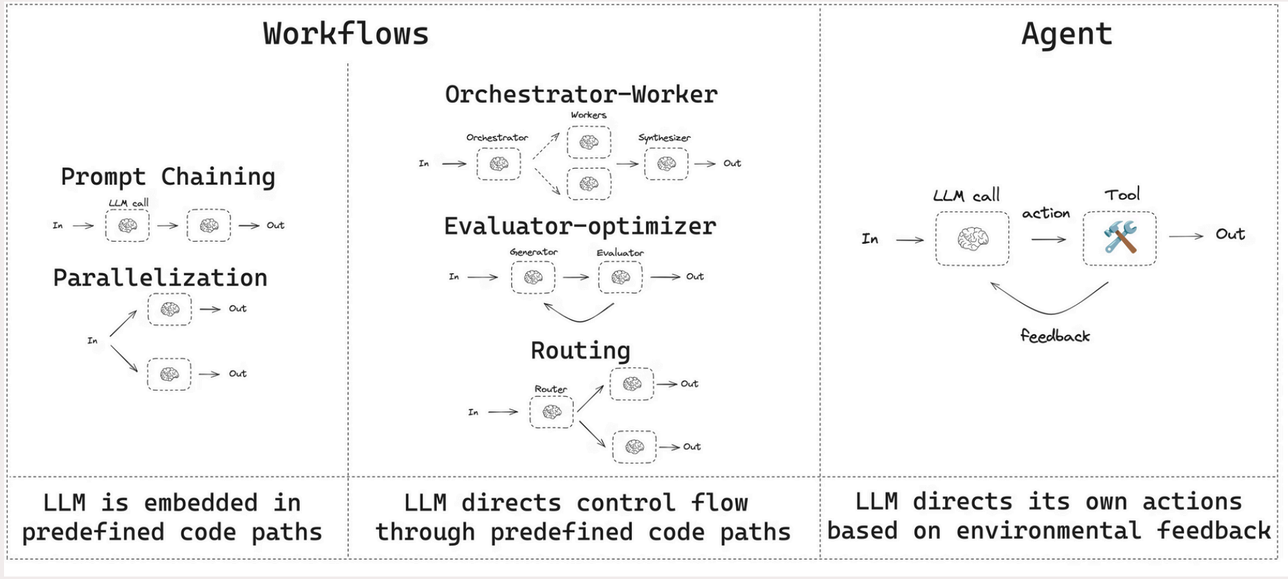
Workflow vs. Agent

"Agent" can be defined in several ways. Some customers define agents as fully autonomous systems that operate independently over extended periods, using various tools to accomplish complex tasks. Others use the term to describe more prescriptive implementations that follow predefined workflows. At Anthropic, we categorize all these variations as ***agentic systems***, but draw an important architectural distinction between workflows and agents:

Workflows are systems where LLMs and tools are orchestrated through **predefined** code paths.

Agents, on the other hand, are systems where LLMs **dynamically** direct their own processes and tool usage, maintaining control over how they accomplish tasks.

LangGraph helps us to build agentic systems.



LangChain Docs : Workflows and Agents

Comparison

	LangGraph	Agents SDK	Google ADK	LangChain	AutoGen	CrewAI	Agno	Temporal	smolagents	LlamaIndex	DSPy	Pydantic AI	Letta	Mastra
Is this a full workflow orchestration framework, or a collection of agent abstractions?														
Workflow orchestration Framework	✅	❌	❌	❌	✅	⚠️	❌	✅	❌	✅	✅	⚠️	❌	✅
Agent Abstractions	✅	✅	✅	✅	✅	✅	✅	❌	✅	✅	✅	✅	✅	✅
Multi Agent Abstractions	✅	✅	✅	❌	✅	✅	✅	❌	✅	✅	❌	✅	✅	⚠️
If a full workflow orchestration framwork, what API does it expose?														
Declarative API	✅				✅	✅		❌		✅	❌	✅		✅
Imperative API	⚠️				❌	❌		✅		❌	✅	❌		❌
Outside of being an orchestration framework what value does this framework provide in production?														
Short term memory storage	✅	❌	✅	✅	✅	✅	✅	❌	❌	✅	❌	❌	✅	✅
Long term memory storage	✅	❌	✅	❌	✅	✅	✅	❌	❌	❌	❌	❌	✅	✅
Human in the loop	✅	❌	❌	❌	❌	❌	❌	⚠️	❌	✅	❌	❌	❌	✅
Human on the loop	✅	❌	❌	❌	❌	❌	❌	❌	❌	✅	❌	❌	❌	❌
Optimization	❌	❌	❌	❌	❌	❌	❌	❌	❌	❌	✅	❌	❌	❌
Tracing Platform	✅	✅	❌	✅	❌	❌	❌	❌	❌	❌	❌	✅	❌	❌
Studio	✅	❌	✅	✅	✅	❌	✅	❌	❌	❌	❌	❌	✅	✅
Low code builder	❌	❌	❌	❌	✅	✅	❌	❌	❌	❌	❌	❌	✅	❌
Prescribed project setup	⚠️	❌	✅	❌	❌	✅	❌	❌	❌	⚠️	❌	❌	❌	✅
Orchestration Features														
Fault tolerance	✅				❌	❌		✅		✅		❌		❌

What changed in v1.0?

LangChain

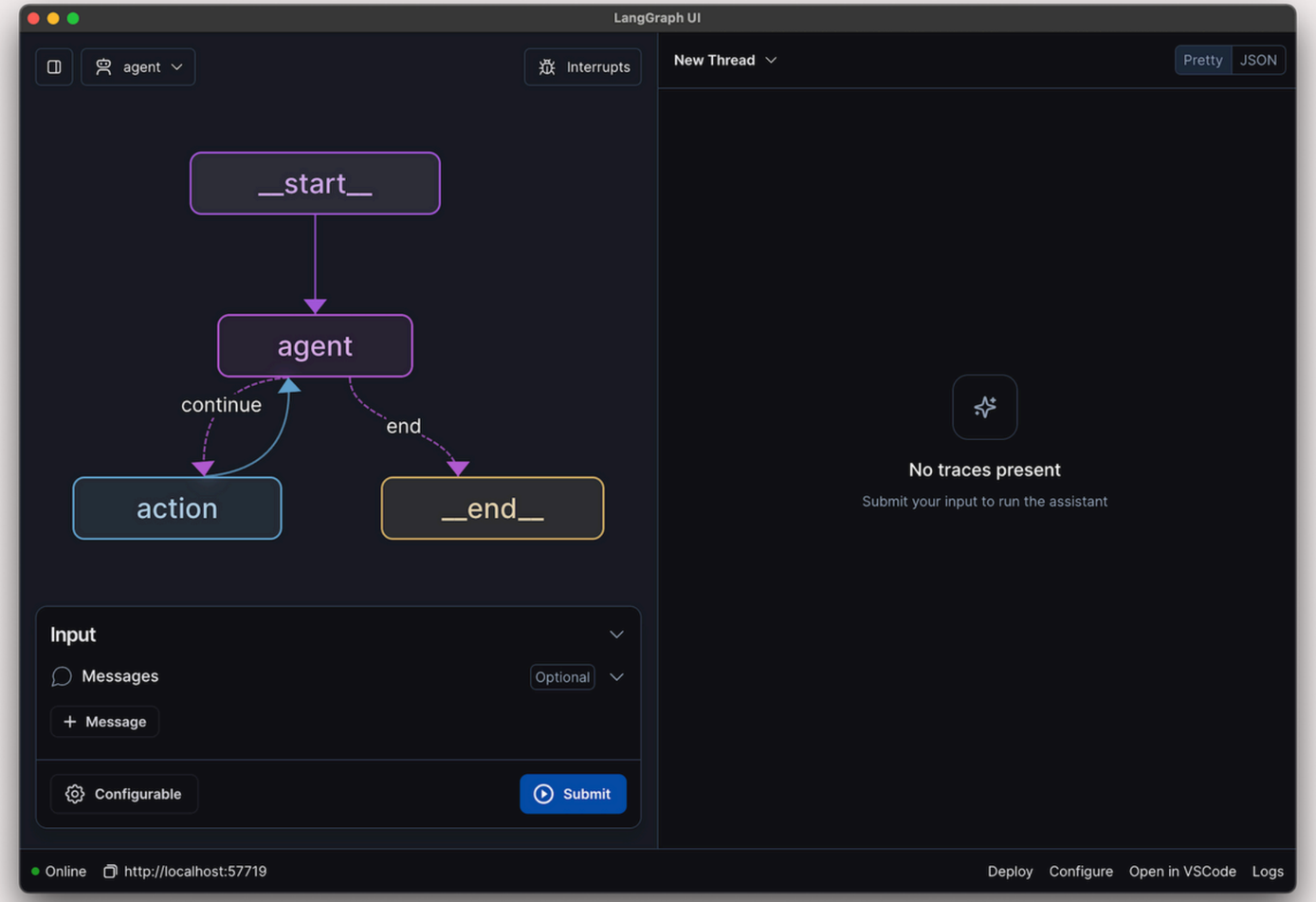
Category	**LangChain v0.x**	**LangChain v1.0**	**Improvement**
Agent Creation	langgraph.prebuilt.create_react_agent	langchain.agents.create_agent	Simpler, standardized interface with greater customization via middleware
Customization Model	Limited hooks, direct graph manipulation required	**Middleware system** with 6 lifecycle hooks	Composable, reusable patterns for context engineering
Built-in Middleware	None	3 prebuilt: PIIMiddleware, SummarizationMiddleware, HumanInTheLoopMiddleware	Production-ready solutions for common patterns
Tool Output Handling	Requires separate LLM calls for structured output	**Integrated structured output** in main loop	25-40% cost reduction, faster execution
Content Access	Provider-specific APIs (Anthropic ≠ OpenAI)	**Unified content_blocks property**	Provider-agnostic: reasoning, citations, tool calls, web search
Message Types	Scattered across modules	Centralized: langchain.messages	Easier imports, cleaner code organization
Package Scope	Monolithic (chains, retrievers, hub, all integrations)	**Focused core** → legacy moved to langchain-classic	Smaller bundle, faster imports, clearer dependencies
Reliability Features	Manual implementation	Built-in: persistence, streaming, human-in-the-loop, time travel	Production-ready without extra code
Learning Curve	Must learn LangGraph for advanced features	Works out-of-the-box; optional LangGraph deep dive	Accessible to beginners, scalable to experts
Migration Path	N/A	Drop-in replacement for most use cases	Easy adoption with guide provided

What changed in v1.0?

LangGraph

Category	LangGraph v0.x	LangGraph v1.0	What Changed?
Focus	General agent runtime + graph building	**Stability-focused** agent runtime	Core APIs frozen, improvements added
Core APIs (State, Nodes, Edges)	Functional	**Unchanged**	✅ Backward compatible - old code works
Execution Model	Functional	**Unchanged**	✅ Execution semantics preserved
LangChain Integration	Separate components	**Tightly integrated**	create_agent runs on top of LangGraph
Durable Execution	Basic support	**First-class feature**	Automatic checkpointing, battle-tested
Checkpointing	Manual configuration	**Built-in, automatic**	Thread ID + Checkpointer = works out-of-box
Persistence Modes	checkpoint_during=True/False	**3 modes: "exit", "async", "sync"**	Fine-grained durability control
Durability Parameter	Deprecated	**New standard**	Replace checkpoint_during with durability
Human-in-the-Loop	Available, manual	**Built-in + improved**	interrupt() + Command primitives
Streaming	Available	**Improved + optimized**	Better token, tool call, reasoning streams
Memory Management	Manual state handling	**Comprehensive memory system**	Short-term + long-term memory abstractions
API Consistency	Language-specific variations	**Unified across Python/TypeScript**	More consistent developer experience
Entry Path	Complex for beginners	**Simple → Advanced progression**	Start with LangChain, drop to LangGraph when needed

LangGraph Studio



LangGraph Studio : The first agent IDE

- **The First "Agent-Native" IDE:** Moves beyond traditional text editors to address the specific needs of non-linear, state-heavy LLM applications.
- **Live Visualization:** Automatically renders your Python code into an interactive graph, making complex loops and logic flows instantly visible.
- **Step-by-Step Debugging:** Allows you to pause execution, inspect the agent's "brain" (state) at any node, and watch tool calls happen in real-time.
- **Interactive "Time Travel":** Enables you to interrupt the agent, manually edit the state (memory) or the last response, and resume execution to simulate different outcomes.
- **Instant Iteration Loop:** Detects code or prompt changes immediately and lets you rerun specific nodes to test fixes without restarting the entire process.

LangGraph in Production

Klarna.

AI that scales customer support

Uber

AI that tests and (and fixes) code

elastic

AI that identifies security attacks

CISCO

LinkedIn
Connect to Opportunity™

AI that hires top talent

GitLab

References

Github Demo

Blogs:
https://dev.to/jamesli/two-basic-streaming-response-techniques-of-langgraph-ioo

https://docs.langchain.com/oss/python/langgraph/overview

https://blog.langchain.com/how-to-think-about-agent-frameworks/

https://blog.langchain.com/langgraph-studio-the-first-agent-ide/

https://www.veribilimiokulu.com/langchain-ve-langgraph-1-0-ile-neler-degisti/ (Turkish)

Official Course:
https://academy.langchain.com/courses/take/intro-to-langgraph/lessons/58238107-course-overview

Chat with Documentation:
Chat LangChain
Copy MCP server