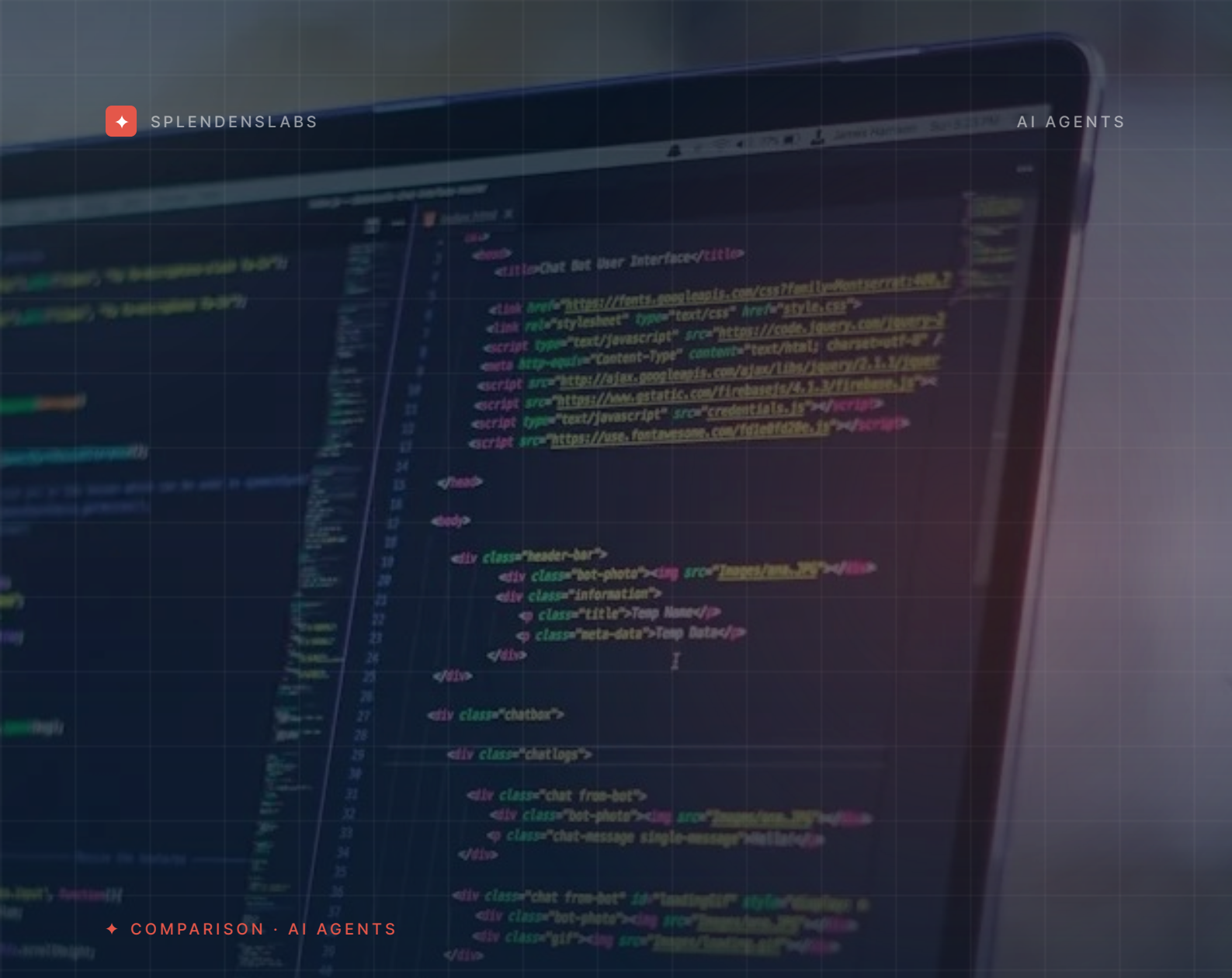


✦ COMPARISON · AI AGENTS

LangChain vs LangGraph

Two related open-source libraries from the same team, built for different jobs. When to reach for LangChain's components and agents, when you need LangGraph's stateful orchestration, and how to use both.

FORMAT	LEVEL	READ TIME	UPDATED
Comparison	Intermediate	10 min	Oct 2026

OVERVIEW

What this comparison covers

LangChain and LangGraph are often presented as rivals. They are not: LangChain provides model integrations, tools, retrievers and a high-level agent interface, while LangGraph is a lower-level runtime for stateful, long-running, controllable agent workflows. This comparison explains where the line sits, what each costs you in complexity, and the questions we use to pick a starting point for client projects.

KEY TAKEAWAYS

- LangChain is the integration and component layer; LangGraph is the orchestration and state layer.
- Start with LangChain's high-level agent APIs for simple tool-calling agents; move to LangGraph when you need explicit control flow.
- LangGraph's strengths are persistence, checkpoints, human-in-the-loop interrupts and durable execution.
- Most production systems use both: LangChain components running inside LangGraph nodes.
- Both are optional — plain SDK calls are a legitimate choice for small, stable workflows.

WHO IT'S FOR

Engineering leads choosing an agent framework

AI engineers moving from prototype to production

Architects reviewing agent designs

Teams maintaining older LangChain code

HOW TO USE IT

Read 01–02 to understand the split, use the table in 03 to compare, then work through the decision checklist in 06 for your specific workflow.

CONTENTS

01 Two layers, not two rivals
02 How each one models a workflow
03 Side by side
04 When to choose which
05 Moving from prototype to production
06 Decision checklist

01 / POSITIONING

Two layers, not two rivals

The clearest way to think about the pair is as layers of one stack. LangChain gives you building blocks; LangGraph decides how and in what order they run, and remembers where it got to.

LAYER 01 **Observability**	LangSmith tracing · Evaluations · Prompt versions · Works with either library
↓	
LAYER 02 **Orchestration**	LangGraph state graph · Checkpointers · Interrupts / approvals · Streaming · Sub-graphs
↓	
LAYER 03 **Agents**	LangChain agent APIs · Tool-calling loop · Middleware / hooks · Structured output
↓	
LAYER 04 **Components**	Chat model integrations · Tools · Retrievers & vector stores · Document loaders · Output parsers

LANGCHAIN

The component and agent layer

A large catalog of integrations plus a high-level way to build a tool-calling agent in a few lines.

LANGGRAPH

The stateful runtime

Workflows as graphs of nodes and edges over shared state, with persistence so runs can pause, resume and recover.

VERSION NOTE

The libraries have converged

Recent LangChain releases build their agent abstractions on the LangGraph runtime, and older chain patterns have been moved to legacy packages. APIs change often; check the current documentation and migration guides before you start, and pin versions in production.

✦ 02 / MECHANICS

How each one models a workflow

The real difference is who owns control flow: the model, inside a loop, or your code, through an explicit graph.

LangChain agent: the model drives the loop

01 User request → 02 Model picks a tool → 03 Tool runs → 04 Result back to model → 05 Repeat until done → 06 Final answer

LangGraph: your graph drives the loop

01 Typed state → 02 Node: classify → 03 Edge: route by condition → 04 Node: act or ask human → 05 Checkpoint saved → 06 End or loop back

MODEL-DRIVEN LOOP
Fast to build and easy to read
Behavior emerges from prompt and tools
Harder to guarantee step order
Good for open-ended assistance

EXPLICIT GRAPH
Control flow is code you can test
Branches, retries and approvals are visible
More design work up front
Good for business processes with rules

✦ 03 / COMPARISON

Side by side

Both are open source and maintained by LangChain, Inc. Both have Python and JavaScript / TypeScript versions, though feature parity can lag between them.

DIMENSION	LANGCHAIN	LANGGRAPH
Primary job	Integrations, components, quick agents	Orchestrating stateful, multi-step workflows

DIMENSION	LANGCHAIN	LANGGRAPH
Control flow	Mostly model-driven tool loop	Explicit nodes, edges and conditional routing
State	Message history, plus what you add	Typed, shared graph state as a first-class concept
Persistence	Via the underlying runtime	Checkpointers to memory, SQLite, Postgres and others
Human-in-the-loop	Available through agent hooks	Native interrupts: pause, inspect, edit, resume
Multi-agent	Possible, but you assemble the pattern	Supervisor, hierarchical and network patterns as graphs
Learning curve	Low for basics; wide surface area	Moderate; requires thinking in state and graphs
Debugging	Traces of chains and tool calls	Traces plus step-by-step state history and replay
Deployment	Any Python / JS service	Any service, or the vendor's managed agent hosting

What both share

- Model-agnostic: swap providers through the same chat model interface.
- Tool definitions, structured output and streaming work the same way.
- Tracing and evaluation through LangSmith or OpenTelemetry-compatible tools.
- Fast release cadence — upgrades need tests, not just a version bump.

04 / GUIDANCE

When to choose which

Defaults we use on client projects. Each assumes you will prototype the riskiest step first.

LANGCHAIN

Single agent, a handful of tools

A support or research assistant that calls a few APIs and answers in one session.

LANGCHAIN

RAG and integrations

You mainly need loaders, retrievers, vector store connectors and model swapping.

LANGGRAPH

Rules-bound business process

Claims, onboarding or approvals where steps, order and sign-offs must be guaranteed.

LANGGRAPH

Long-running or resumable work

Runs that wait on people or external systems for hours or days and must survive restarts.

LANGGRAPH

Multi-agent coordination

A supervisor routing work between specialized agents with shared state and clear hand-offs.

NEITHER

Small, stable pipeline

One or two model calls with fixed logic — a provider SDK and plain code are simpler to own.

✦ HYBRID OPTION

LangGraph for the skeleton, LangChain for the parts

Our usual production shape: LangGraph defines the workflow, state, checkpoints and approval points; each node uses LangChain model, tool and retriever integrations. You get explicit control without re-implementing connectors.

05 / PRODUCTION

Moving from prototype to production

Whichever you start with, the path to production looks similar. These steps keep a migration from LangChain-only to LangGraph cheap.

01 **Isolate tools and prompts**

Keep tool functions and prompt templates in their own modules so any orchestrator can call them.

02 **Write the workflow down as states**

Even before LangGraph, sketch the states, transitions and where a human must approve.

03 **Add persistence early**

Choose a durable checkpointer such as Postgres before you need resumability, not after an outage.

04 **Trace everything**

Turn on tracing from day one and keep a golden set of tasks to replay on every change.

05 **Pin and test upgrades**

Lock library versions, read release notes and run your eval suite before upgrading.

Common pitfalls

- Building a complex multi-agent graph before a single agent works reliably.
- Letting the model decide steps that business rules already define.
- Storing large documents in graph state instead of references to them.
- Following old tutorials written for deprecated chain APIs.

06 / DECIDE

Decision checklist

Answer these for the workflow you are building, not the platform you might build one day.

WORKFLOW

- [ ] Steps and their order are known and must be enforced

- [ ] Some steps need human approval or edits
- [ ] Runs can last longer than a single request

STATE

- [ ] Work must resume after a crash or deploy
- [ ] We need to inspect or replay past runs step by step
- [ ] Several agents must share and update the same context

TEAM

- [ ] The team is comfortable modeling work as states and transitions
- [ ] We have tests and evals to catch behavior changes on upgrade
- [ ] We have checked current docs, versions and hosting terms

Mostly ticks in **Workflow** and **State** point to LangGraph, with LangChain components inside it. Mostly blanks point to LangChain's high-level agent, or to plain SDK code if the task is small.

GET IN TOUCH

Build the agent, not the framework

We design and ship production agents and agentic workflows on LangGraph, LangChain or plain SDKs — whichever fits. Bring us the workflow and we will map it to the right architecture.

- Workflow-to-graph design session
- Prototype of your riskiest agent step
- Tracing, evals and human-in-the-loop setup

BOOK AN AI STRATEGY SESSION →

EMAIL
hello@splendenslabs.com

WEB
splendenslabs.com

OFFICE
Sona Incubation Foundation, Salem