SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 03 · Agentic AI Engineering

13

LangGraph Deep Dive

Stateful graph-based agents with checkpointing & HITL

Module 13 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Stateful graph-based agents with checkpointing & HITL. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	4	13/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 03

02 Learning Objectives

- Model agent control flow as a state graph.
- Add checkpointing to persist and resume runs.
- Insert human-in-the-loop (HITL) approval steps.
- Build a browser-controlling agent.

03 Core Concepts

Graphs over chains

LangGraph models an agent as nodes (steps) and edges (transitions) over a shared typed state. Unlike linear chains, graphs support loops, branches, and conditional routing — exactly what real agents need.

State & checkpointing

A checkpointer persists graph state after each node, so runs can pause, resume, recover from failure, and support multi-session memory. This durability is what separates demos from production agents.

Human-in-the-loop

You can interrupt the graph before a risky action (sending email, executing code) to get human approval, then resume. HITL is critical wherever mistakes are costly.

Tool & browser control

Nodes can call tools — including a browser driver (Playwright) — letting the agent navigate pages, fill forms, and extract data under graph control with checkpoints between steps.

04 Visual Diagram

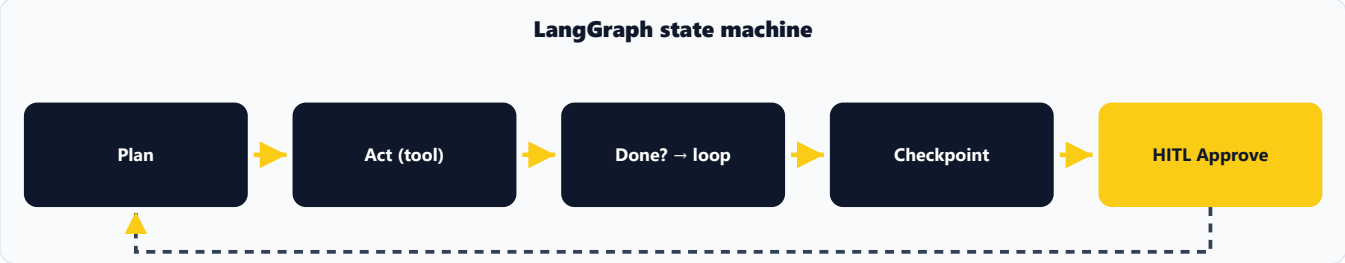

Figure 13 — Nodes and edges over shared state, with checkpoints for resume and a human-in-the-loop gate.

05 Key Frameworks & Tools

LangGraph | Playwright | Checkpointer (SQLite/Postgres) | LangSmith

06 Hands-On Build

🛠 Browser Sidekick Project (Playwright + LangGraph)

1. Define the graph state and nodes (plan, act, observe).
2. Add Playwright tools for navigate, click, and extract.
3. Enable a checkpointer to persist and resume runs.
4. Insert a HITL approval node before any form submission.

07 Code Walkthrough

A minimal LangGraph state machine — PYTHON

```python
from langgraph.graph import StateGraph, END
from typing import TypedDict

class State(TypedDict):
    task: str
    steps: list

g = StateGraph(State)
g.add_node("plan", plan_node)
g.add_node("act", act_node)
g.set_entry_point("plan")
g.add_edge("plan", "act")
g.add_conditional_edges("act", lambda s: END if done(s) else "act")
app = g.compile(checkpointer=checkpointer)
```

08 🧪 Lab Assistant

Lab 13 · A resumable graph agent

Prerequisites:pip install langgraph · A checkpointer (SQLite)

1. **Define State + plan/act nodes with a conditional edge**
   > Expected: The graph loops until done
2. **Compile with a checkpointer and interrupt the run**
   > Expected: State persists; the run resumes from the last node
3. **Add a HITL approval before a risky action**
   > Expected: Execution pauses for your approval

Troubleshooting

- Progress lost on crash → you forgot the checkpointer.
- Never terminates → fix the 'done' conditional.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Putting risky actions in the graph with no HITL gate.
- Forgetting a checkpointer, losing all progress on failure.
- Unbounded loops without a termination condition.

✓ KEY TAKEAWAYS

- Graphs give agents loops, branches, and durable state.
- Checkpointing turns fragile demos into resumable systems.
- HITL gates protect against costly autonomous mistakes.

📺 **Companion video:** Watch the module 13 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

LangGraph docs • Playwright Python docs • LangGraph HITL guide