Choosing an Agentic Solution

- 2 Days Course
- Hands-on and Lecture

Course Overview

Stop treating "agent" as a product category. Determine where control, state, authority, and transitions actually exist, then choose a solution.

You will take one real job through script, chain, workflow, and agent; locate agency in software or in model context (including mixed systems); compare application-owned mechanisms and whether to own that machinery at all; then decide tool boundary, model ownership, runtime ownership, topology, and tests. You leave with a completed decision record. This course selects a framework, it does not implement a framework.

Who Should Attend

- Engineering leads and architects choosing an agent approach
- Staff engineers asked to pick a framework or a managed agent platform
- ML / MLOps engineers leaving chatbots and RAG wrappers
- Platform engineers who will operate whatever is chosen
- Product and engineering pairs who must agree what "an agent" means

What You'll Learn

- Determine where control, state, authority, and transitions exist in an application.
- Evaluate and compare application-owned mechanisms (LangGraph, CrewAI, AutoGen, raw tool calling) and managed agent platforms.
- Define tool boundaries, model ownership, runtime ownership, and topology for agentic systems.
- Build comprehensive evaluation and test strategies including deterministic checks, trajectory scoring, and budget bounds.
- Complete a decision record selecting the optimal agentic solution architecture for a specific real-world job.

Outline

1. LangGraph vs CrewAI vs AutoGen vs Rolling Your Own: Building Agentic AI That Won't Break

- Why linear chains die when the job is not a straight line
- Two machines, not four products: programmed control structure vs emulated control structure, plus a thin loop you type yourself
- LangGraph: explicit graph and state
- CrewAI: role- and task-oriented multi-actor
- AutoGen: conversational / event-driven multi-actor
- Rolling your own: application-owned tool_calls loop
- Same job, three runs: interrupt-and-resume on a graph; the same gate negotiated by a role-play loop; the same gate in ten lines of raw tool calling

- What breaks: state, loops, tool execution, recovery, cost
- Conclusion of this chapter (and of the webinar): choose a kind of machine. Not a cloud. Not a factory SKU. Not 'therefore on-prem.'
- Lab: Name the kind that would not break on a job you already have. Do not pick a vendor yet.

2. What You Are Actually Trying to Automate

- Outcome, frequency, blast radius, human already in the loop
- Constraints: audit, latency, cost ceiling, stop condition
- Open the decision record
- Lab: Write the job card

3. Script, Chain, Workflow, or Agent

- Script: fixed code, no model in the loop
- Chain: model in a line, no authority to select further actions
- Workflow: executable control structure; a model may fill a slot or route a branch
- Agent: a system with bounded authority to select and execute actions toward an objective based on observed state
- An LLM router inside a workflow does not, by itself, make the system an agent
- Lab: Classify the job. Defend no agent if that is the truth

4. Programmed Agents vs. Emulated Agents

- Programmed: control structure exists as executable software. State, transitions, authority, termination, and recovery can be implemented and inspected
- Emulated: control structure lives substantially in prompts and is enacted by the model
- State in software vs state in context
- Explicit transitions vs generated transitions
- Programmed authority vs prompted compliance
- Programmed agents may still contain probabilistic nodes
- Emulated agents can look deeper than the software underneath them
- Programmed and emulated are properties of control boundaries, not exclusive labels for an entire application
- Emulated reasoning can exist inside a programmed control boundary
- Lab: Sketch both, and one mixed system (programmed outer loop, emulated planning node, programmed approval and stop). Mark where state, authority, transitions, and termination live. Apply the random-string test as an architecture test, not a liveness test

5. LangGraph, CrewAI, AutoGen, or Rolling Your Own

- Conditional: these are application-owned mechanisms. Chapter 7 may retire the choice
- LangGraph: explicit graph and state orchestration
- CrewAI: role- and task-oriented multi-actor orchestration
- AutoGen: conversational / event-driven multi-actor orchestration
- Thin loop: application-owned tool_calls loop
- For each: where state lives, where transitions live, where the model has discretion, where authority is enforced
- What each makes easy, hides, and makes expensive to leave
- Lab: Score the four application-owned mechanisms against the job. Record a provisional winner and rejects, citing mechanism

6. Tools, Contracts, and MCP

- Tool as contract: schema, permissions, side effects, idempotency

- Framework-native tools vs MCP as a replaceable boundary
- Discovery is not authority
- Who may invoke, what may be retried, what must never loop
- Lab: Specify one tool, inputs, outputs, deny rules, timeout, retry, who may call it

7. Hosted Models vs. Models You Operate

- Who owns model execution
- Hosted inference vs weights and serving you run
- Data path, pin, routing, cost, swap
- Lab: Hosted, operated, or hybrid, plus the constraint that would force a change

8. Managed Agent Platform vs. Runtime You Operate

- Who owns agent execution: loop, state, tools, policy, not 'a VM somewhere'
- Now decide whether you should own the machinery scored in Chapter 4
- A managed agent platform can remove the framework choice
- Application-owned runtime: you keep the control structure on metal, VM, or container
- Hosted application delivery ≠ hosted agent runtime
- Lab: Draw the boundary. Label each box: managed platform, application-owned hosted process, or application-owned process you operate. Confirm or retire the Chapter 4 provisional winner

9. One Agent vs. Many

- Default: one agent
- Minimize independent centers of agency
- A second agent must independently bound at least one of: authority, tools, context, trust boundary, lifecycle, evaluation target
- Different prompts or personas are not different agents
- Supervisors, handoffs, and conversations of agents are complexity you justify
- If you cannot name what must be independently bounded, do not create another agent
- Lab: Keep one, or name the second agent's independent bound

10. How You Will Know It Works

- Score: outcome + trajectory + policy + termination + budget, not chain-of-thought
- Golden tasks and deterministic checks; statistical eval only for the rest
- A 97% correct agent that occasionally loops at $80 is not 97% successful
- Lab: Write five failing tests. At least two must not use an LLM judge. One must catch non-termination or budget breach

11. Making the Choice

- Complete the decision record. The architecture falls out of chapters 1–10
- A valid choice resolves every applicable dimension, it is not a pick from a flat menu
- What this course will not teach next week
- Lab: copy the answers from the previous labs, strike anything Chapter 8 retired, you now have the winning agentic solution requirements.. Peer review for smuggled emulators, persona-as-agent, a Chapter 5 winner that Chapter 8 retired in silence, and a managed platform that was never considered

Prerequisites

- Can read Python and an HTTP API
- Have used an LLM in an application