Agentic AI with Cloud-Neutral Architectures

- 4 Days
- Lecture and Hands-on Labs

Course Overview

This course teaches engineers how to design, reason about, and implement agentic AI systems using cloud-neutral, portable architectures that can run across public cloud, private cloud, or on-prem environments. This course emphasizes architecture over APIs and operational reality over demos, preparing participants to implement agentic systems safely regardless of their eventual platform choice. By the end of the course, participants will be able to design and build agentic workflows that are portable, operable, and resilient to tooling and vendor change.

Review this course online at https://www.alta3.com/courses/ai-neutral-agent

Who Should Attend

- Platform and Systems Engineers
- Application Developers building AI-enabled systems
- ML Engineers and MLOps practitioners
- Technical Architects and Technical Leads
- Engineering teams operating in hybrid or multi-cloud environments
- Teams evaluating cloud platforms but not yet committed

What You'll Learn

- Understand architectural invariants of agentic AI systems.
- Design agents that plan, act, and adapt across environments.
- Apply orchestration patterns using open frameworks.
- Implement portable and resilient agentic workflows.

Outline

Foundations of Agentic Systems

1. The limits of single-prompt systems
2. Why autonomy emerges in production AI
3. Agentic AI versus generative AI and workflow automation
4. What qualifies a system as 'agentic'
5. Control loops, decision cycles, and authority boundaries Lab: Decomposing a real problem into agent responsibilities #### Anatomy of an Agent
6. Core agent roles and responsibilities
7. Planner
8. Executor
9. Critic
10. Observer

11. Separating reasoning from action
12. Tools as contracts rather than code
13. State, context, and continuity Lab: Designing an agent interface and decision loop #### Tool Use and Action Design
14. What a tool represents in an agentic system
15. Designing safe and explicit tool interfaces
16. Schemas, constraints, and validation boundaries
17. Error handling, retries, and fallback strategies
18. Tool orchestration using LangChain concepts
19. Preventing runaway action loops Lab: Modeling tool contracts and failure scenarios #### Memory, State, and Retrieval
20. Short-term context versus long-term memory
21. Persistent state and agent identity
22. Retrieval-augmented reasoning patterns
23. Tradeoffs between recall, relevance, and cost
24. Designing memory layers independent of vendor services Lab: Designing a memory strategy for an agentic workflow #### Multi-Agent Decomposition and Orchestration
25. When multi-agent architectures are justified
26. Role-based decomposition and delegation
27. Coordination, handoffs, and shared context
28. Conflict resolution and arbitration strategies
29. Expressing orchestration patterns using open abstractions Lab: Designing a multi-agent architecture for a complex task #### Reliability, Observability, and Evaluation
30. Why autonomous systems fail differently than traditional software
31. Observability signals specific to agentic behavior
32. Tracing decisions, actions, and outcomes
33. Evaluating agent behavior without relying on hidden reasoning
34. Using ClearML concepts for observability and evaluation Lab: Defining metrics and traces for agent reliability #### Deployment Readiness and Governance (Cloud-Neutral)
35. Agentic systems as long-lived services
36. Latency, cost, and throughput considerations
37. Versioning agents, tools, and policies
38. Authority boundaries and execution permissions
39. Human-in-the-loop and kill switch design
40. Governance and auditability without vendor lock-in Lab: Designing a deployment-ready agentic architecture #### Building an End-to-End Agentic Workflow
41. Selecting an appropriate agentic workflow scope
42. Defining agent roles and responsibilities
43. Using LangChain-style primitives for planning and tool execution
44. Wiring tools to external services and APIs Lab: Implementing a production-shaped agentic workflow using open-source tooling #### Memory, Failure Handling, and Control
45. Choosing memory strategies for the workflow
46. Managing short-term context and persistent state
47. Handling tool failures and partial execution
48. Designing retries, fallbacks, and escalation paths Lab: Injecting failures and validating recovery behavior #### Deployment Shape and Operational Readiness
49. Running agents as portable services
50. Deployment shape across containers, VMs, and local environments
51. Cost awareness without managed service guardrails
52. Observability and operational signals Lab: Reviewing and validating the final deployment architecture

Labs

- Decomposing a real problem into agent responsibilities

- Designing an agent interface and decision loop
- Modeling tool contracts and failure scenarios
- Designing a memory strategy for an agentic workflow
- Designing a multi-agent architecture for a complex task
- Defining metrics and traces for agent reliability
- Designing a deployment-ready agentic architecture
- Implementing a production-shaped agentic workflow using open-source tooling
- Injecting failures and validating recovery behavior
- Reviewing and validating the final deployment architecture

Prerequisites

- Familiarity with software systems and APIs
- General understanding of AI or LLM concepts
- No cloud-specific experience required