Agentic AI on AWS

- 4 Days
- Lecture and Hands-on Labs

Course Overview

This course teaches engineers how to design, reason about, and implement agentic AI systems on Amazon Web Services, using Amazon Bedrock as a managed model access layer and LangChain as a tool orchestration layer. The first three days establish the architectural foundations of agentic systems, including planning, tool use, memory, orchestration, reliability, and governance. These concepts are mapped to AWS execution primitives without tying architectural authority to any single framework or product. Amazon Bedrock is used for model access and inference management; the course does not focus on Bedrock-specific agent frameworks. The fourth day is a hands-on build day, where participants implement a production-shaped agentic workflow on AWS, focusing on tool wiring, memory choices, failure handling, and deployment shape rather than framework internals. By the end of the course, participants will be able to design and build agentic workflows that are realistic, operable, and aligned with AWS operational constraints.

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

Who Should Attend

- Application Developers building AI-powered systems on AWS
- Platform and Cloud Engineers supporting AI workloads
- ML Engineers and MLOps practitioners
- Technical Architects and Technical Leads

What You'll Learn

- Understand architectural invariants of agentic AI systems.
- Design and deploy adaptive agents using LangChain concepts.
- Map agent state and memory to AWS services.
- Evaluate and govern agentic workflows within AWS constraints.

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 #### 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 #### 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. LangChain as a tool orchestration and routing layer
19. Preventing runaway action loops #### 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. Mapping memory concepts to AWS-managed storage #### 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 LangChain primitives #### 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. AWS-native observability touchpoints #### Deployment Readiness and Governance on AWS
35. Agentic systems as long-lived services
36. Latency, cost, and throughput considerations
37. Versioning agents, tools, and policies
38. Identity, access, and permission boundaries
39. Least-privilege IAM considerations for agent tool execution
40. Human-in-the-loop and kill switch design
41. Governance and auditability in AWS environments #### Building an End-to-End Agentic Workflow
42. Selecting an appropriate agentic workflow scope
43. Defining agent roles and responsibilities
44. Using LangChain primitives for planning and tool execution
45. Wiring tools to AWS services #### Memory, Failure Handling, and Control
46. Choosing memory strategies for the workflow
47. Managing short-term context and persistent state
48. Handling tool failures and partial execution
49. Designing retries, fallbacks, and escalation paths #### Deployment Shape and Operational Readiness
50. Running agents as managed services on AWS
51. Deployment shape and service boundaries
52. Cost awareness and execution limits
53. Observability and operational signals

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 Amazon Bedrock
- 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
- Prior AWS experience is helpful but not required