ALTA3 RESEARCH

Agentic AI on AWS

- 3 Days Course
- Hands-on Labs and Lecture

Course Overview

This course teaches engineers how to design, reason about, and implement production-shaped agentic AI systems on Amazon Web Services (AWS). Participants use Amazon Bedrock for managed model access, open orchestration patterns for agent execution, and Model Context Protocol (MCP) as a standardized interface between agents and external tools and services.

The course establishes the architectural foundations of agentic systems, including control loops, tool use, context, memory, orchestration, reliability, evaluation, security, and governance. These concepts are then mapped to AWS services and current AWS agent infrastructure without making any single framework the architectural center of the course.

Participants learn how Amazon Bedrock AgentCore capabilities can support production agent runtimes, MCP-based tool integration, identity, observability, and operational controls. AgentCore Gateway is used to demonstrate how APIs, Lambda functions, AWS services, and existing MCP servers can be exposed through standardized MCP interfaces while maintaining AWS authentication and authorization boundaries.

By the end of the course, participants will be able to design and implement agentic workflows that are operable, secure, observable, cost-aware, resilient, and aligned with AWS production practices.

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
- Engineering teams responsible for deploying and operating agentic applications on AWS

What You'll Learn

- Understand the architectural invariants of production agentic AI systems
- Design agents that plan, act, verify, and adapt using explicit control loops
- Use Amazon Bedrock as a managed model access and inference layer
- Build agent runtimes using open orchestration patterns and AWS infrastructure
- Understand MCP hosts, clients, servers, tools, resources, and prompts
- Expose AWS capabilities to agents through MCP-compatible interfaces
- Use Amazon Bedrock AgentCore Gateway to provide controlled MCP access to APIs, Lambda functions, and other services
- Design safe agent tools using schemas, validation, IAM, and least-privilege permissions
- Implement context, memory, persistent state, checkpointing, and retrieval
- Apply single-agent and multi-agent orchestration patterns appropriately
- Build observability, evaluation, and regression testing for agentic workflows
- Implement authority boundaries, human approval, failure recovery, and resource controls

- Deploy and operate agentic systems within AWS security, cost, and reliability constraints

Outline

1. Foundations of Agentic Systems on AWS

- The limits of single-prompt systems and traditional workflow automation
- Why autonomy emerges in production AI
- Agentic AI versus generative AI and deterministic workflows
- What qualifies a system as agentic: control loops, decisions, tools, state, and authority
- Architectural invariants versus framework and cloud implementation details
- Understanding where models, agent runtimes, protocols, tools, and AWS infrastructure fit
- Autonomy tiers and determining how much authority an agent should receive
- Starting with deterministic workflows and introducing autonomy only where justified
- Mapping agentic architecture concepts to AWS
- Lab: Decomposing a real problem into agent responsibilities, AWS dependencies, and autonomy tiers

2. Anatomy and Runtime of an Agent

- Core agent responsibilities: planning, execution, verification, and observation
- The agent control loop: observe, reason, act, verify, and continue or stop
- Separating model reasoning from application actions
- State, context, continuity, and explicit termination conditions
- The agent runtime as the execution layer between models, tools, state, policies, and applications
- Separating orchestration logic from cloud infrastructure
- Context construction, context budgets, and context isolation
- Tool registries and capability discovery
- Policy enforcement, permissions, and execution boundaries
- Evaluation and observability hooks within the runtime
- Custom agent runtimes versus managed agent runtime infrastructure
- Lab: Building a basic agent control loop with explicit state, tools, execution limits, and termination conditions

3. Models and Agent Execution with Amazon Bedrock

- Amazon Bedrock as a managed model access and inference layer
- Separating model selection from agent architecture
- Selecting models based on capability, latency, context, and cost
- Model configuration and inference parameters
- Structured model outputs for reliable application integration
- Model routing and fallback strategies
- Handling throttling, model failures, and malformed responses
- Controlling inference consumption and execution budgets
- Using open agent and orchestration frameworks with Amazon Bedrock
- Amazon Bedrock AgentCore Runtime and managed agent execution
- Choosing between application-managed and AWS-managed runtime components
- Keeping agent logic separable from model and runtime infrastructure
- Lab: Connecting an agent runtime to Amazon Bedrock and validating structured model interaction, execution limits, and fallback behavior

4. Model Context Protocol (MCP) Fundamentals

- Why standardized agent-tool interoperability matters
- Framework-specific tools versus protocol-based capabilities
- MCP architecture: hosts, clients, servers, and capabilities

- MCP tools as callable actions
- MCP resources as contextual data
- MCP prompts as reusable interaction templates
- Tool discovery and capability negotiation
- Tool schemas, inputs, outputs, and validation
- MCP transport and deployment considerations
- Stateless versus stateful MCP interactions
- Protocol versioning and compatibility considerations
- Separating agent implementation from tool implementation
- Designing reusable MCP capabilities for multiple agents and frameworks
- Lab: Building and testing an MCP server that exposes a controlled tool and resource to an agent

5. MCP and AWS Service Integration with AgentCore Gateway

- Amazon Bedrock AgentCore Gateway as a managed connectivity boundary for agentic traffic
- Connecting an MCP client to AgentCore Gateway
- Exposing AWS Lambda functions as MCP tools
- Exposing REST APIs and OpenAPI-defined services as MCP tools
- Connecting existing MCP servers through AgentCore Gateway
- Aggregating multiple MCP capabilities behind a unified gateway
- Tool discovery and invocation through the Gateway
- Using MCP tools, resources, and prompts with AgentCore Gateway
- Inbound authentication for agent access
- Outbound authorization and credential management for backend services
- IAM and OAuth across agent, gateway, and tool boundaries
- Trust boundaries between agents, Gateway, MCP servers, and AWS resources
- Hosting MCP servers using AgentCore Runtime
- Choosing between direct MCP connectivity and Gateway-mediated access
- Lab: Configuring AgentCore Gateway and exposing an AWS-backed capability to an agent through MCP

6. Context, Memory, State, and Retrieval on AWS

- Distinguishing context, memory, persistent state, and retrieval
- Short-term working context versus long-term memory
- Persistent state and agent identity
- Checkpointing and durable execution
- Resuming interrupted agent workflows
- Retrieval-augmented reasoning patterns
- Selecting what should be stored, retrieved, summarized, or discarded
- Tradeoffs between recall, relevance, latency, context consumption, and cost
- Mapping state and memory requirements to AWS storage capabilities
- Selecting storage based on data semantics rather than framework defaults
- Exposing retrieval capabilities through MCP where appropriate
- Protecting stored agent context and memory
- Maintaining state independently from individual model invocations
- Lab: Implementing persistent state and retrieval for an AWS-hosted agentic workflow

7. Agent Orchestration and Multi-Agent Patterns

- When orchestration should remain deterministic
- When an agent should control workflow progression
- When multi-agent architectures are actually justified
- Single-agent versus multi-agent tradeoffs
- Sequential, parallel, hierarchical, supervisor, and handoff patterns

- Role-based decomposition and delegation
- Shared state versus isolated agent context
- Coordination, arbitration, and conflict resolution
- Preventing circular delegation and uncontrolled agent spawning
- Sharing MCP capabilities across multiple agents
- Agent-to-tool interaction versus agent-to-agent communication
- Combining deterministic application logic with probabilistic agent decisions
- Expressing orchestration with open framework primitives
- Lab: Implementing an orchestrated workflow with delegation, shared state, MCP tools, and explicit control boundaries

8. Reliability, Observability, and Evaluation on AWS

- Why autonomous systems fail differently than traditional applications
- Failure modes across models, MCP tools, state, orchestration, and policies
- Observing complete agent decision loops
- Tracing model interactions, MCP calls, state transitions, outcomes, latency, and cost
- AWS logging, metrics, tracing, and operational signals
- AgentCore observability capabilities and agent execution traces
- Evaluating observable behavior without relying on hidden model reasoning
- Defining success criteria before evaluating an agent
- Golden datasets and repeatable evaluation scenarios
- Outcome, trajectory, tool-use, and policy-compliance evaluation
- Deterministic checks versus probabilistic evaluation
- LLM-as-judge techniques and limitations
- Regression testing across model, prompt, tool, MCP server, and policy changes
- Evaluation gates for deployment
- Detecting behavioral degradation in production
- Lab: Instrumenting an agent workflow and evaluating successful, failed, and policy-violating executions

9. Security, Identity, and Authority Boundaries

- Agent identity versus user identity
- IAM boundaries for agent runtimes, gateways, and tools
- Least-privilege execution
- Authentication and authorization across MCP boundaries
- Inbound versus outbound authorization
- Credential isolation and avoiding credential exposure to models
- Separating model capability from execution authority
- Read, propose, approve, and execute permission models
- Autonomy gradients: recommend, prepare, request approval, execute, and execute autonomously
- Human-in-the-loop approval gates
- Prompt injection, indirect instructions, tool poisoning, and data leakage
- MCP server and tool trust boundaries
- Sandboxing high-risk actions
- Execution limits and resource budgets
- Kill switches and emergency containment
- Auditing agent decisions, MCP calls, tool actions, and human approvals
- Lab: Adding IAM boundaries, MCP authorization, an approval gate, execution limits, and an auditable action path

10. Failure Handling, Recovery, and Resource Control

- Designing for partial execution rather than assuming atomic agent operations

- Model failures, MCP server failures, tool failures, unavailable AWS services, and malformed responses
- Retryable versus non-retryable failures
- Exponential backoff, fallback, and circuit-breaker patterns
- Recovering from interrupted workflows
- Resuming safely from persistent state
- Handling unavailable MCP capabilities
- Preventing duplicate tool actions during retries and recovery
- Idempotency for actions with side effects
- Escalating ambiguous states to humans
- Bounding inference usage, tool calls, API calls, concurrency, retries, and execution duration
- Detecting runaway loops and pathological agent behavior
- Graceful degradation when models, tools, or dependent services become unavailable
- Lab: Injecting model, MCP, tool, state, and permission failures and validating recovery, fallback, escalation, and resource controls

11. Deploying and Operating Agentic Systems on AWS

- Agentic systems as long-lived production services
- Separating model access, agent runtime, MCP capabilities, state, and external integrations
- AWS deployment shapes for agent workloads
- Amazon Bedrock AgentCore Runtime
- Containers, serverless execution, managed compute, and Kubernetes
- Hosting MCP servers on AgentCore Runtime
- AgentCore Gateway as a production tool and integration boundary
- Selecting deployment infrastructure based on workload characteristics
- Externalizing model configuration, MCP endpoints, state stores, policies, and credentials
- Health checks, readiness, graceful shutdown, and durable restart
- Scaling agent execution and controlling concurrency
- Latency, throughput, inference cost, and resource considerations
- Versioning agents, prompts, MCP servers, tools, and policies
- Operational monitoring and production evaluation
- Governance and auditability in AWS environments
- Lab: Deploying and validating a production-shaped AWS agentic system using Amazon Bedrock, MCP-based tools, persistent state, security controls, and operational instrumentation

Prerequisites

- Familiarity with software systems and APIs
- General understanding of AI or LLM concepts
- Basic familiarity with Python
- Prior AWS experience is helpful but not required
- AWS Lab Access: Each participant is provided temporary access to an AWS training environment for the hands-on labs. Participants must use a workstation and network that permit access to the AWS web console and required AWS services. Customers that restrict access to AWS must provide an approved method of access for participants.