Faysal Aziz

Agentic AI for Tech Pros

Build enterprise-ready AI agents that retrieve trusted knowledge, call tools through MCP, integrate with APIs, run on-premise, and coordinate through event-driven multi-agent systems - 2 days

FA-0048 · 2 days

Agentic & Generative AI · Software Development

Introduction

Agentic AI is moving from experimentation into real technical implementation. The useful question is no longer whether an AI model can answer a prompt, but whether it can retrieve the right enterprise knowledge, choose the right tool, call the right API, follow operational rules, and coordinate safely with other agents. That is where RAG and MCP become important. RAG helps ground model responses in relevant data retrieved from controlled knowledge sources, while MCP provides a standardized way for AI applications to connect with external tools, data sources, prompts, and workflows.

This 2-day course is designed for technical professionals who want to build practical agentic AI systems using Python, APIs, Retrieval-Augmented Generation, MCP, on-premise deployment patterns, and multi-agent orchestration using a pub-sub system such as Kafka. Kafka will be used only as an event-driven coordination mechanism; the course will not teach Kafka administration, cluster tuning, or Kafka internals.

The course reflects current industry direction: modern agent frameworks support agents with tools, handoffs, guardrails, and tracing; MCP servers expose tools that models can invoke against external systems; and on-premise model-serving platforms such as vLLM and Ollama support OpenAI-compatible interfaces for easier integration with existing AI application patterns.

The instructor has over **30 years of industry experience** and will use real industry-demanded content instead of making the course academic. The emphasis is on practical architecture, integration, controls, deployment thinking, and implementation patterns that technical teams can recognize in real enterprise environments.

Learning outcomes

By the end of this course, participants will be able to:

- Explain the architecture of practical agentic AI systems
- Design Python-based agents that use tools, APIs, and structured outputs
- Build RAG pipelines for enterprise knowledge retrieval
- Understand chunking, embeddings, vector search, retrieval, and response grounding
- Use MCP concepts to expose tools, resources, and prompts to agentic applications
- Integrate agents with internal and external APIs
- Understand on-premise agentic AI deployment requirements
- Design multi-agent systems using coordinator, worker, and validator patterns
- Use a pub-sub system such as Kafka for event-driven agent orchestration
- Apply guardrails, logging, tracing, validation, and human approval controls

Prerequisites

- Good Python scripting knowledge
- Experience with REST APIs and JSON
- Working experience of Linux or server-side environments
- Usage experience of LLMs, prompts, tokens, and model APIs
- Good understanding of databases or document repositories
- Working understanding of enterprise IT or application environments
- Prior experience with vector databases or MCP is helpful but not mandatory

Training outline

1. Agentic AI Foundations

Understanding Agentic AI

1. Agents versus chatbots
2. Agents versus scripts
3. Agents versus traditional automation
4. Planning, acting, observing, and correcting
5. Autonomy boundaries
6. Human control points

Core Agent Architecture

1. Model layer
2. Instruction layer
3. Tool layer
4. Retrieval layer
5. Memory and context layer
6. State layer
7. Guardrail layer
8. Observability layer

Practical Technical Use Cases

1. Internal knowledge assistants
2. API-driven task execution
3. IT operations support
4. Incident enrichment
5. Technical document search
6. Infrastructure request handling
7. Multi-system workflow coordination

2. Python-Based Agent Development

Python Environment Setup

1. Virtual environments
2. Dependency management
3. Environment variables
4. Configuration files
5. Secrets handling
6. Logging structure

Building Agents in Python

1. Agent instructions
2. Model client configuration
3. Tool registration

4. Function calling
5. Structured inputs
6. Structured outputs
7. Error handling
8. Runtime control

Tool Design for Agents

1. Tool purpose definition
2. Input schema
3. Output schema
4. Validation logic
5. Permission boundaries
6. Execution logging
7. Failure handling

3. Retrieval-Augmented Generation

RAG Fundamentals

1. Purpose of RAG
2. Knowledge grounding
3. Retrieval versus generation
4. Semantic search
5. Source attribution
6. Hallucination reduction
7. Enterprise knowledge access

RAG Pipeline Design

1. Document ingestion
2. Text extraction
3. Chunking strategy
4. Embedding generation
5. Vector indexing
6. Query embedding
7. Similarity search

8. Context assembly
9. Answer generation

Retrieval Quality

1. Chunk size
2. Chunk overlap
3. Metadata filtering
4. Hybrid search concepts
5. Reranking concepts
6. Retrieval thresholds
7. Context window management
8. Relevance evaluation

Enterprise RAG Considerations

1. Internal document sources
2. Access-controlled retrieval
3. Sensitive data handling
4. Document freshness
5. Index update strategy
6. Source traceability
7. Retrieval audit logs
8. Data residency requirements

4. MCP for Agent Tool and Context Integration

MCP Fundamentals

1. Purpose of MCP
2. MCP clients
3. MCP servers
4. Tools
5. Resources
6. Prompts
7. Tool discovery
8. Tool invocation

MCP Tool Design

1. Tool naming
2. Tool descriptions
3. Input schemas
4. Output structures
5. Error responses
6. Permission boundaries
7. Tool selection clarity
8. Tool execution safety

MCP and Enterprise Systems

1. API-backed MCP tools
2. Database-backed MCP tools
3. File and document access tools
4. Internal service integration
5. Authentication considerations
6. Access control mapping
7. Auditability
8. Tool governance

MCP with RAG and Agents

1. MCP as a retrieval access layer
2. MCP as an API tool layer
3. MCP for enterprise context exposure
4. Agent-controlled retrieval
5. Agent-controlled tool execution
6. Guarded MCP tool usage
7. MCP failure handling

5. API Integration for Agentic Systems

API Concepts for Agents

1. REST methods

2. Headers
3. Authentication
4. JSON payloads
5. Response codes
6. Pagination
7. Rate limits
8. Error responses

Agent-to-API Implementation

1. API client design
2. API wrapper functions
3. Request validation
4. Response parsing
5. Retry handling
6. Timeout handling
7. Result normalization
8. Audit logging

Internal API Considerations

1. Internal service endpoints
2. API gateways
3. Service accounts
4. Certificates
5. Firewall rules
6. Role-based access
7. Sensitive data controls

6. On-Premise Agentic AI Architecture

On-Premise Deployment Drivers

1. Data residency
2. Security policies
3. Restricted networks
4. Internal-only systems

5. Latency requirements
6. Cost control
7. Compliance needs

On-Premise Components

1. Local model serving
2. Python agent runtime
3. RAG indexing layer
4. Vector database
5. MCP server layer
6. API gateway
7. Pub-sub layer
8. Secrets management
9. Logging and monitoring

Local Model Serving

1. OpenAI-compatible endpoints
2. vLLM integration considerations
3. Ollama integration considerations
4. Model endpoint configuration
5. Context window limits
6. Latency considerations
7. GPU and CPU capacity
8. Model lifecycle management

Security and Operations

1. Network segmentation
2. TLS and certificates
3. Credential storage
4. Service accounts
5. Access reviews
6. Audit logging
7. Backup planning
8. Change control

7. Multi-Agent System Design

Multi-Agent Concepts

1. Agent specialization
2. Agent delegation
3. Coordinator agents
4. Worker agents
5. Retrieval agents
6. Tool execution agents
7. Validator agents
8. Human escalation agents

Multi-Agent Interaction Patterns

1. Sequential handoff
2. Parallel execution
3. Supervisor-worker pattern
4. Planner-executor pattern
5. Reviewer pattern
6. Retrieval-then-action pattern
7. Human-in-the-loop pattern
8. Event-driven pattern

Agent Communication Design

1. Message structure
2. Task identifiers
3. Correlation identifiers
4. Agent identifiers
5. Status fields
6. Error fields
7. Source references
8. Audit metadata

8. Multi-Agent Orchestration with Pub-Sub

Pub-Sub Role in Agentic Systems

1. Event-driven coordination
2. Producer agents
3. Consumer agents
4. Topic-based routing
5. Asynchronous processing
6. Decoupled execution
7. Task queues
8. Result queues

Kafka Usage Scope

1. Kafka as event transport
2. Kafka as coordination backbone
3. Kafka topics for agent tasks
4. Kafka producers for agent outputs
5. Kafka consumers for agent workers

Event Flow for Multi-Agent Workloads

1. Task request event
2. Retrieval request event
3. Retrieval result event
4. MCP tool request event
5. MCP tool result event
6. Agent handoff event
7. Validation event
8. Approval request event
9. Completion event
10. Failure event

Failure Handling

1. Retry events

2. Timeout handling
3. Duplicate message handling
4. Dead-letter topic concepts
5. Partial failure handling
6. Agent unavailability handling
7. Human escalation

Disclaimer

This course outline is provided as a professional planning guideline and does not constitute a fixed or binding delivery specification. The trainer reserves the right to amend, restructure, expand, reduce, replace, or adjust any topic, sequence, tool, demonstration, exercise, or technical emphasis without prior notice, based on participant readiness, classroom progress, available infrastructure, technology changes, operational constraints, and the trainer's professional judgment.

© 2026 Faysal Aziz · Kuala Lumpur