Faysal Aziz

Agentic AI Engineering

Build practical AI agents that use Python, APIs, n8n, on-premise systems, and event-driven multi-agent orchestration - 2 days

FA-0046 · 2 days

Agentic & Generative AI · Software Development · Automation, RPA & Power Platform

---

Introduction

Agentic AI is becoming a practical engineering discipline, not just a prompting technique. Technical teams are now expected to build systems where AI can interpret tasks, call tools, interact with APIs, automate workflows, coordinate with other agents, and operate within real enterprise constraints. Modern agent frameworks now emphasize tools, handoffs, guardrails, tracing, and structured execution, while automation platforms such as n8n provide AI-agent workflow capabilities with connected tools and memory. For on-premise environments, local inference platforms such as vLLM and Ollama support OpenAI-compatible interfaces, making it easier to connect enterprise agents to internal infrastructure without depending entirely on public cloud APIs. Apache Kafka and similar pub-sub systems remain suitable foundations for event-driven coordination between distributed services and agents.

This 2-day course is built for technical professionals who need to design and implement agentic AI solutions in practical environments. The course focuses on Python-based agent development, API integration, n8n automation, on-premise deployment considerations, and multi-agent orchestration using a pub-sub system such as Kafka. Kafka will be used only as a coordination mechanism; the course will not teach Kafka administration or internals.

The instructor has over **30 years of industry experience** and will use real industry-demanded content instead of treating the subject as an academic exercise. The emphasis is on how agentic AI systems are actually designed, integrated, controlled, and operated in technical environments.

Learning outcomes

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

- Understand the architecture and behavior of agentic AI systems

- Design practical single-agent and multi-agent workflows
- Build Python-based agents that use tools and APIs
- Integrate agents with n8n workflows and webhooks
- Connect agentic systems to internal and external APIs
- Understand on-premise agent deployment requirements
- Use a pub-sub system such as Kafka for multi-agent coordination
- Design message contracts for agent-to-agent communication
- Identify operational risks in production agentic AI systems

Prerequisites

- Good Python scripting knowledge
- Experience with REST APIs and JSON
- Working understanding of Linux or server-side environments
- Experience working with LLMs, prompts, and model APIs
- Working knowledge of workflow automation concepts
- Good understanding of enterprise IT or application environments

Training outline

1. Agentic AI Fundamentals

What Agentic AI Means

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

Core Agent Architecture

1. Model layer
2. Instruction layer
3. Tool layer
4. Memory and context layer

5. State layer
6. Guardrail layer
7. Observability layer

Practical Technical Use Cases

1. IT operations automation
2. API-driven task execution
3. Internal support assistants
4. Workflow triage
5. Incident enrichment
6. Report generation
7. Infrastructure request handling

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. Runtime control
8. Error handling

Agent Tool Design

1. Tool purpose definition
2. Tool input schema
3. Tool output schema
4. Tool validation
5. Tool execution logging
6. Tool failure handling
7. Tool permission boundaries

3. API Integration for Agents

REST API Concepts for Agentic Systems

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

Agent-to-API Integration

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

Enterprise API Considerations

1. Internal service endpoints

2. API gateways
3. Service accounts
4. Access control
5. Certificates
6. Firewall rules
7. Sensitive data handling

4. n8n for Agentic Workflow Automation

n8n Workflow Foundations

1. Trigger nodes
2. Action nodes
3. Conditional branches
4. Data mapping
5. Webhooks
6. Credentials
7. Execution history
8. Error paths

n8n AI Agent Workflows

1. AI agent node concepts
2. Prompt configuration
3. Tool connections
4. Memory usage
5. Input mapping
6. Output mapping
7. Human approval steps
8. Workflow retries

Python and n8n Integration

1. Calling Python services from n8n
2. Calling n8n webhooks from Python
3. Passing JSON between systems
4. Managing correlation IDs

5. Handling callbacks
6. Coordinating execution status

5. On-Premise Agentic AI Architecture

On-Premise Deployment Drivers

1. Data residency
2. Internal security policies
3. Restricted networks
4. Latency requirements
5. Cost control
6. Enterprise integration needs

On-Premise Components

1. Local model serving
2. Agent runtime services
3. n8n automation layer
4. API gateway
5. Pub-sub layer
6. Secrets management
7. Logging and monitoring
8. Storage and state

Local Model Serving

1. OpenAI-compatible endpoints
2. vLLM integration considerations
3. Ollama integration considerations
4. Model endpoint configuration
5. Context limits
6. Latency considerations
7. Capacity planning

Security and Operations

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

6. Multi-Agent System Design

Multi-Agent Concepts

1. Agent specialization
2. Agent delegation
3. Coordinator agents
4. Worker agents
5. Validator agents
6. Human escalation agents
7. Shared versus private state

Multi-Agent Interaction Patterns

1. Sequential handoff
2. Parallel task execution
3. Supervisor-worker pattern
4. Planner-executor pattern
5. Reviewer pattern
6. Human-in-the-loop pattern
7. 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. Audit metadata

7. 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 an event transport
2. Kafka as a 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. Task accepted event
3. Task progress event
4. Tool execution event
5. Agent handoff event
6. Validation event
7. Approval request event
8. Completion event

9. 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