Faysal Aziz

Advanced Distributed Harness Engineering

Multi-Agent Orchestration Across On-Premises and Cloud AI - 3 days

FA-0511 · 3 days

Agentic & Generative AI · Software Development

Introduction

Engineer coordinated, event-driven agent systems that operate across enterprise boundaries.

Advanced Harness Engineering moves beyond a single agent working within a single process. Enterprise systems may require specialised agents for planning, research, implementation, validation, security and human approval. These agents must communicate reliably, preserve task state, recover from failures and operate under clearly defined security boundaries.

This three-day course focuses on constructing a distributed multi-agent harness using Python, an agent-orchestration framework and Apache Kafka. Kafka provides durable event streams through which services can publish, subscribe to, store and process events. These capabilities make it suitable for decoupling agents, distributing workloads and preserving communication between independently deployed components.

Participants will design supervisor-and-worker and event-driven agent patterns rather than relying on uncontrolled group conversations. Modern orchestration frameworks support specialised subagents, shared workflow state, durable execution, streaming and human intervention. The course will examine these capabilities while maintaining framework-independent architectural principles.

The resulting platform will combine locally hosted models with approved cloud-model services. On-premises models may be exposed through an OpenAI-compatible endpoint using platforms such as vLLM, allowing the harness to apply a consistent client interface while routing tasks according to privacy, capability, cost and availability requirements.

The course is intentionally limited to a functional reference architecture rather than a complete enterprise agent platform. Participants will concentrate on orchestration, Kafka communication, hybrid model routing, reliability, security and operational visibility. The instructor has more than 30 years of industry experience

and will use practical, industry-demanded approaches rather than treating the subject as a purely academic exercise.

Learning outcomes

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

- Design a distributed multi-agent harness architecture
- Define specialised agent roles and communication contracts
- Implement supervisor, worker and event-driven orchestration patterns
- Use Kafka topics, producers, consumers and consumer groups for agent communication
- Route workloads between on-premises and cloud models
- Maintain distributed task state and correlation
- Implement retries, timeouts, idempotency and dead-letter handling
- Apply security and human-approval boundaries
- Monitor multi-agent execution and diagnose failures
- Assemble and assess a production-oriented reference harness

Prerequisites

- Completion of practical Harness Engineering training or equivalent experience
- Strong working knowledge of Python
- Experience with APIs, JSON and asynchronous processing
- Familiarity with AI agents and tool calling
- Basic understanding of Apache Kafka
- Familiarity with containers and Linux
- General knowledge of authentication, logging and distributed systems
- Access to an on-premises model endpoint and an approved cloud-model service

Training outline

1. Distributed Harness Architecture

1. Single-agent versus multi-agent systems
2. Agent responsibilities and boundaries
3. Supervisor-and-worker architecture
4. Event-driven agent architecture

5. Control plane and execution plane
6. Shared services and external tools
7. Human oversight boundaries

2. Multi-Agent Design

1. Planner agents
2. Specialist worker agents
3. Validation agents
4. Security and policy agents
5. Agent capability registration
6. Task delegation
7. Result aggregation
8. Agent termination conditions

3. Orchestration Workflows

1. Deterministic workflows
2. Dynamic agent routing
3. Graph-based orchestration
4. Shared workflow state
5. Parallel and sequential execution
6. Conditional branches
7. Human approval nodes
8. Workflow recovery

4. Kafka Communication Architecture

1. Brokers, topics and partitions
2. Producers and consumers
3. Consumer groups
4. Agent request and response topics
5. Event envelopes
6. Correlation and causation identifiers
7. Message keys and partitioning
8. Schema management

5. Reliable Agent Messaging

1. Delivery semantics
2. Idempotent task processing
3. Offset management
4. Retry topics
5. Dead-letter topics
6. Timeouts and stale tasks
7. Duplicate-event handling
8. Ordering requirements

6. Python Agent Services

1. Asynchronous agent workers
2. Kafka producer integration
3. Kafka consumer integration
4. Agent service lifecycle
5. Structured message validation
6. Tool execution adapters
7. State updates
8. Graceful shutdown

7. Hybrid AI Model Integration

1. On-premises model serving
2. OpenAI-compatible local endpoints
3. Cloud-model API integration
4. Unified model adapters
5. Capability-based routing
6. Data-sensitivity routing
7. Fallback and failover
8. Model availability controls

8. Distributed State and Memory

1. Task state
2. Agent-local state

3. Shared workflow state
4. Conversation and context state
5. Checkpointing
6. State recovery
7. Context isolation
8. Retention and deletion policies

9. Security and Governance

1. Service authentication
2. Agent identity
3. Topic access controls
4. Secret management
5. Tool permissions
6. Sensitive-data boundaries
7. Cloud egress controls
8. Audit records
9. Human escalation

10. Observability and Evaluation

1. Distributed logging
2. Correlation identifiers
3. Agent execution traces
4. Kafka consumer lag
5. Model latency and failures
6. Tool-call monitoring
7. Token and resource usage
8. Workflow success criteria

11. Resilience and Operational Controls

1. Agent failure isolation
2. Circuit breakers
3. Back-pressure
4. Rate limiting
5. Model-provider outages

6. Worker scaling
7. Poison-message handling
8. Controlled workflow cancellation

12. Advanced Harness Assembly

1. Architecture integration
2. Supervisor-agent deployment
3. Specialist-agent deployment
4. Kafka event-flow integration
5. Hybrid model routing
6. Validation-agent integration
7. Failure and recovery testing
8. Human approval integration
9. End-to-end system assessment

13. Production Readiness

1. Container deployment
2. Environment separation
3. Configuration management
4. Versioned prompts and policies
5. Kafka capacity considerations
6. Model-serving capacity
7. Operational runbooks
8. Release and rollback controls

Disclaimer

This outline is provided as a professional instructional guideline for a three-day advanced programme. The trainer reserves the right to amend, reorder, reduce, expand or substitute topics at their professional discretion to reflect participant capability, infrastructure availability, organisational requirements, security constraints, framework changes or instructional priorities. Such adjustments may be made without prior notice where reasonably necessary to preserve the relevance, safety and practical value of the training.

© 2026 Faysal Aziz · Kuala Lumpur