

COURSE CURRICULUM

Advanced RAG and AI Agents

with Hugging Face, Microsoft Foundry, MCP and Agent Framework

4 DAYS | 32 HOURS | INSTRUCTOR-LED | HANDS-ON

From transformer foundations to adaptive RAG, orchestrated agents and production observability

Final Table of Contents

Course Snapshot

Course Name: Advanced RAG and AI Agents with Hugging Face, Microsoft Foundry, MCP and Agent Framework

Duration: 4 Days | 32 Hours

Delivery: Instructor-led with guided demonstrations and practical labs

Audience: Developers, AI engineers, data professionals and solution architects building grounded AI applications and agents

Course Overview

A practical course for building grounded AI applications and agents with Hugging Face and Microsoft Foundry. Participants progress through four RAG patterns, then create Foundry agents, connect MCP capabilities, orchestrate multi-agent workflows, and apply evaluation and observability.

Course Objectives

- Explain transformer architecture and use Hugging Face models, tokenizers and pipelines.
- Build Standard, Hybrid, Corrective, and Agentic or Adaptive RAG solutions.
- Create and extend agents using Microsoft Foundry SDK and MCP capabilities.
- Apply Microsoft Agent Framework orchestration patterns to controlled agent workflows.
- Evaluate RAG and agent quality and observe execution through Microsoft Foundry.

Prerequisites

- Basic Python programming knowledge.
- Familiarity with APIs, JSON and Generative AI concepts is helpful.
- Basic knowledge of search or document processing is beneficial but not mandatory.

Platform and Access Requirements

- Python access with permission to install and run required packages.
- A suitable code editor like VS Code and internet/network access for practical exercises.
- Client-provided Microsoft Azure and Microsoft Foundry access with required model credentials.

Day 1 | AI Foundations, Hugging Face and Microsoft Foundry

Outcome: Participants understand transformer-based models, use Hugging Face components, and prepare a Microsoft Foundry project for RAG and agent development.

MODULE 01

AI Foundations and the Generative AI Lifecycle

Establish the concepts required to understand models, retrieval and agentic applications.

- Explain Artificial Intelligence, Machine Learning, deep learning, Generative AI, foundation models and Large Language Models in one connected learning path.
- Describe tokens, embeddings, context windows, inference, hallucination and the limits of standalone models.
- Compare direct model calls, RAG applications, tool-enabled agents and orchestrated multi-agent systems.

Hands-on: Call a hosted model and inspect tokens, responses and common failure behaviour.

MODULE 02

Transformer Architecture

Explain the architecture that powers modern language models without turning the module into a mathematics-heavy deep dive.

- Follow text through tokenization, embeddings, positional information, attention layers and output generation.
- Explain self-attention, multi-head attention, feed-forward layers, residual connections and normalization at a conceptual level.
- Compare encoder-only, decoder-only and encoder-decoder architectures and connect each to suitable tasks.

Hands-on: Inspect tokenization and attention-aware model output for a short text sample.

MODULE 03

Hugging Face Models, Tokenizers and Pipelines

Use the Hugging Face ecosystem to explore and run transformer models.

- Navigate model information, task types and model configuration while considering suitability, licensing and resource needs.
- Load tokenizers and pretrained models and use pipelines for common language tasks.
- Generate embeddings and compare model outputs for summarization, classification or text generation.

Hands-on: Run one pretrained model pipeline and generate embeddings for sample content.

MODULE 04

Introduction to Microsoft Foundry

Prepare the cloud environment and establish where models, agents, tools, evaluation and observability fit.

- Explore Foundry projects, model access, deployments, connections and the Agent Service experience.
- Use the Foundry SDK client and authentication pattern to connect Python code to a project.
- Compare a basic model application with a managed Foundry agent and identify the later course build path.

Hands-on: Connect to a Foundry project and invoke a selected model through the SDK.

Day 2 | Standard and Hybrid RAG

Outcome: Participants build strong retrieval foundations and improve relevance through chunking, hybrid search, filtering and reranking.

MODULE 05

RAG Foundations and Ingestion Pipeline

Build the common foundation shared by all four RAG patterns.

- Explain the RAG lifecycle from ingestion and indexing to retrieval, context assembly and grounded generation.
- Prepare documents through extraction, cleaning, metadata enrichment, embedding generation and index creation.
- Design responses that use retrieved context, preserve source information and handle insufficient evidence.

Hands-on: Create a small searchable knowledge base and generate a grounded answer.

MODULE 06

Pattern 1: Standard RAG with Advanced Chunking

Compare three chunking approaches and select one based on document structure and retrieval behaviour.

- Use recursive chunking to split content by progressively smaller separators while controlling chunk size and overlap.
- Use structural chunking to preserve headings, sections, tables or other document boundaries.
- Use semantic chunking to group passages by meaning and compare retrieval results across the three approaches.

Hands-on: Build one Standard RAG pipeline and compare recursive, structural and semantic chunk retrieval.

MODULE 07

Pattern 2: Hybrid RAG

Combine lexical and semantic retrieval to improve recall across varied enterprise questions.

- Contrast keyword, vector and semantic search and explain why hybrid retrieval can combine complementary strengths.
- Create a hybrid retrieval flow that merges candidate results before final context selection.
- Tune the retrieval depth and context budget to balance relevance, latency and answer quality.

Hands-on: Build a hybrid search pipeline and compare results with vector-only retrieval.

MODULE 08

Metadata Filtering and Reranking

Improve Hybrid RAG by narrowing the search space and reordering candidate passages.

- Design metadata for document type, business area, date, access scope or other useful retrieval constraints.
- Apply pre-filtering or post-filtering where appropriate and inspect the effect on recall.
- Rerank retrieved candidates before prompt construction and retain the strongest evidence for generation.

Hands-on: Add metadata filtering and reranking to the Hybrid RAG pipeline.

Day 3 | Corrective, Agentic and Adaptive RAG

Outcome: Participants grade retrieval quality, correct weak evidence, decompose complex questions and adapt retrieval decisions.

MODULE 09

Pattern 3: Corrective RAG

Add quality-control steps that decide whether retrieved evidence is suitable before generation.

- Grade retrieved passages for relevance and identify weak, conflicting or incomplete evidence.
- Correct the retrieval path through query rewriting, alternate search or additional retrieval when the first attempt is insufficient.
- Grade groundedness after generation and route unsupported answers for revision or safe response handling.

Hands-on: Build a Corrective RAG flow with retrieval relevance and groundedness grading.

MODULE 10

Pattern 4: Agentic RAG with Query Decomposition

Use an agent to break a complex request into focused retrieval tasks.

- Decompose multi-part questions into subqueries and retrieve evidence for each part.
- Coordinate retrieval tools, intermediate findings and answer synthesis within an agentic flow.
- Maintain source traceability while combining evidence from several retrieval steps.

Hands-on: Build an Agentic RAG workflow that decomposes and answers a multi-part question.

MODULE 11

Adaptive RAG and Routing

Allow the application to select the retrieval strategy based on the request and available evidence.

- Classify requests that need no retrieval, a direct lookup, hybrid search, decomposition or a corrective loop.
- Route questions to the appropriate retrieval path and apply explicit stop or fallback conditions.
- Compare adaptive behaviour against a single fixed RAG pipeline using representative test cases.

Hands-on: Create an adaptive router across Standard, Hybrid and Corrective retrieval paths.

MODULE 12

RAG Evaluation

Measure retrieval and response quality before integrating the solution into a broader agent.

- Create a compact test dataset with questions, expected evidence and acceptance criteria.
- Evaluate retrieval relevance, groundedness, response relevance and citation or source quality.
- Use evaluation findings to adjust chunking, filters, reranking, prompts and routing decisions.

Hands-on: Evaluate the four RAG patterns on a shared test set and document one improvement.

Day 4 | Foundry Agents, MCP, Orchestration and Observability

Outcome: Participants create a Foundry agent, connect MCP tools, orchestrate specialist agents, and apply evaluation and observability.

MODULE 13

Creating Agents with the Foundry SDK

Move the RAG capability into a managed or hosted agent development workflow.

- Create an agent with model configuration, instructions, knowledge and tools through the Foundry SDK.
- Manage messages, conversations and structured responses while testing tool selection and agent behaviour.
- Connect the strongest RAG pattern from Day 3 as enterprise knowledge for the agent.

Hands-on: Create and test a Foundry agent that can answer grounded enterprise questions.

MODULE 14

Creating and Connecting an MCP Server

Expose reusable business capabilities through MCP and make them available to the Foundry agent.

- Explain MCP hosts, clients, servers and the roles of tools, resources and prompts.
- Create a Python MCP server with validated inputs, structured outputs and clear errors.
- Connect the MCP server to the agent and test capability discovery and invocation.

Hands-on: Build a small MCP server and invoke one MCP tool from the Foundry agent.

MODULE 15

Agent Framework Orchestration Patterns

Use Microsoft Agent Framework patterns when the solution requires coordinated specialist behaviour.

- Compare sequential, concurrent, handoff, group chat and manager-led orchestration patterns.
- Select the lowest level of orchestration complexity that reliably meets the business requirement.
- Combine specialist agents, RAG capabilities and MCP tools with controlled message flow and human approval where needed.

Hands-on: Build a two-agent orchestration using one suitable Agent Framework pattern.

MODULE 16

Foundry Evaluation, Observability and Capstone

Close the development lifecycle by measuring quality and tracing the complete execution path.

- Evaluate agent outcomes and process quality through test data and suitable built-in or custom criteria.
- Use Foundry tracing, metrics and monitoring to inspect model calls, retrieval steps, MCP tools and agent decisions.
- Complete an integrated solution that combines Adaptive RAG, a Foundry agent, MCP capability and an orchestrated specialist flow.

Hands-on: Demonstrate the capstone with one complex query, one MCP call, one orchestration flow and one trace or evaluation review.

Capstone Project

Enterprise Knowledge and Operations Agent: participants build a Microsoft Foundry agent that routes questions through the appropriate RAG pattern, decomposes complex requests when required, invokes a business capability through MCP, and coordinates a specialist agent flow. The final demonstration includes grounded evidence, agent evaluation and an observable execution trace.

Four-Day Learning Journey

- Day 1** | Understand transformers, use Hugging Face models and connect to Microsoft Foundry.
- Day 2** | Build Standard and Hybrid RAG with advanced chunking, filtering and reranking.
- Day 3** | Implement Corrective, Agentic and Adaptive RAG and compare their quality.
- Day 4** | Create Foundry agents, connect MCP, orchestrate specialists and apply evaluation and observability.