Building Enterprise RAG & Workflows with Dify

- 2 Days Course
- Hands-on and Lecture

Course Overview

Students stand up, configure, extend, and evaluate a multi-tenant Dify deployment. They integrate Knowledge Pipelines, parent-child chunking, hybrid retrieval, deterministic and model-directed visual workflows, external MCP tools, human-in-the-loop approvals, and programmatically interface with Dify via its REST and Python SDK boundaries.

Who Should Attend

- AI Engineers & Developers (Suggested)
- Enterprise Systems Architects (Suggested)
- LLM Application Developers (Suggested)
- Technical Platform Managers (Suggested)

What You'll Learn

- Understand Dify's decoupled platform architecture and multi-tenancy model
- Configure advanced Knowledge Base ingestion using parent-child chunking and hybrid search
- Integrate external LLM, embedding, and reranking endpoints (vLLM, TEI, Cohere)
- Design visual AI workflows combining deterministic logic, dynamic agent routing, and code execution nodes
- Implement MCP tool connectors and Human-in-the-Loop approval nodes for sensitive actions
- Programmatically integrate Dify workflows via REST API and Python SDK with dynamic runtime context

Outline

1. Dify Architecture & Platform Core

- Understanding Dify's decoupled architecture: API Server, Worker Nodes, Web UI, Vector Database (Qdrant/Milvus/Weaviate), and Redis
- Connecting external model infrastructure: Decoupling LLM generation (vLLM, Ollama) from specialized embedding and reranking endpoints (TEI)
- Multi-tenancy and workspace boundaries: Organizing team permissions, workspaces, and dataset scoping

2. Knowledge Base Ingestion, Chunking & Knowledge Pipelines

- Data indexing modes: High-Quality vs. Economical modes and embedding engine configurations
- Chunking and segmentation techniques: General, rule-based splitting, and Parent-Child retrieval strategies (retrieving against smaller child chunks while returning parent context)

- Enterprise Knowledge Pipelines: Configuring pipelines as reusable workspace assets with test runs, intermediate-variable inspection, and metadata management

3. Hybrid Search & Retrieval Optimization

- Setting up Dify's hybrid retrieval pipeline combining Dense Vector similarity and Full-Text Keyword search
- Integrating cross-encoder reranking endpoints (Cohere, Jina, local TEI rerankers) to rescale score distributions
- Inspecting retrieval testing panels, comparing retrieval relevance, and tuning Top-K boundaries and score thresholds
- Lab: Dify Deployment, Knowledge Pipeline Setup & Retrieval Tuning
- Deploy Dify using containerized orchestration connected to external vLLM and TEI inference endpoints
- Ingest unstructured corporate documents into Dify Knowledge Bases using configurable parent-child segmentation
- Interactively inspect retrieval results while tuning hybrid search weights, top-k filters, and reranker thresholds

4. Enterprise Identity, Security Boundaries & Tool Authority

- Deep dive into Dify's native security model: Workspace identity, role scoping, and dataset-level isolation
- Enterprise identity integration patterns: Mapping workspace identity against external SAML/OIDC identity providers
- Decoupling security boundaries: Distinguishing Authentication ("Who is Alice?") vs. Document Authorization ("Which chunks can Alice retrieve?") vs. Tool Authority ("Which actions can Alice's workflow execute?")
- Delegated credentials and tool authorization: Mapping application JWTs to permitted MCP actions and external API nodes

5. Deterministic, Model-Directed & Human-in-the-Loop Workflows

- Building program-controlled workflows: Knowledge retrieval, IF/ELSE conditional logic, and sandboxed Python/JS code execution nodes
- Building model-directed workflows: Agent nodes, dynamic routing, and tool selection (Built-in tools, OpenAPI, Workflow tools, MCP)
- Mixed orchestration patterns: Embedding model-controlled tool choice inside deterministic validation loops
- Human-in-the-Loop approval: Pausing visual workflows prior to sensitive actions and resuming execution upon explicit human authorization

6. Application Integration, Operations & APIs

- Interfacing with Dify programmatically: Consuming Dify's Chat/Workflow REST APIs and Python SDK
- Injecting dynamic query variables, session keys, and runtime security context into API calls
- Inspecting Dify application logs and workflow execution traces, and integrating external observability providers (Langfuse and LangSmith)
- Lab: Visual Workflow Orchestration, MCP Tools & Human Authority
- Construct a visual workflow combining Knowledge Base retrieval, Intent Classification, and an external MCP tool node
- Inject a Human-in-the-Loop approval node to pause execution before issuing external API writes
- Integrate the workflow programmatically into a custom web application using the Python SDK/REST API with dynamic user context parameters