LiteLLM AI Gateway Essentials

- 1 Day Course
- Hands-on and Lecture

Course Overview

This intensive, hands-on workshop equips engineering teams to deploy, configure, and operate LiteLLM Proxy, a widely adopted open-source AI Gateway designed to route across 100+ LLM providers. Moving beyond basic LLM proxying, students dive directly into local and containerized deployments, setting up an OpenAI-compatible interface to unify providers including OpenAI, Anthropic, AWS Bedrock, Azure OpenAI, Google Vertex AI, and local Ollama instances.

Participants will learn to establish governed AI access across their organization, moving away from unmanaged programmatic LLM traffic. The curriculum covers core operational patterns: multi-tenant access control with virtual keys, database-backed budget enforcement (PostgreSQL), high-availability routing and fallbacks, Redis-backed semantic caching, real-time PII guardrails (Presidio), model access policies, and governing AI agent tool authority using LiteLLM's Model Context Protocol (MCP) gateway.

Who Should Attend

- Backend Engineers
- DevOps & Platform Engineers
- AI & MLOps Engineers
- Systems Administrators managing LLM infrastructure

What You'll Learn

- Lower API Expenses & Latency: Reduce token spend and eliminate duplicate queries by deploying Redis semantic vector caching and multi-tenant virtual keys with strict budget caps.
- Establish Governed AI Access: Bring unmonitored application traffic into a single managed gateway with centralized identity, rate limits, model access policies, and encrypted credential management.
- Enhance System Resilience: Configure automated multi-provider failover chains, cooldown periods, and load balancing across upstream model endpoints to maintain service stability during provider outages.
- Prevent Provider Lock-In: Build a vendor-agnostic AI layer using an OpenAI-compatible abstraction to swap providers with zero-to-minimal application code changes.
- Govern Agent Tool Authority (MCP): Centralize, authenticate, and restrict external tools, databases, and agent API connections using LiteLLM's built-in Model Context Protocol (MCP) gateway.
- Filter PII & Sensitive Content: Implement real-time PII masking (via Microsoft Presidio containers) and pre/post-call guardrails on both LLM completions and MCP tool executions.
- Gain Comprehensive Audit Visibility: Track token consumption per team/virtual key, inspect full request logs, and stream operational metrics to Prometheus, OpenTelemetry, or Langfuse.
- Enforce Model Access Policies: Control precisely which models, rate limits (TPM/RPM), and tool entitlements are permitted per team, project, or virtual key.

Outline

1. The Enterprise AI Gateway & Deployment

- Lecture: Why Applications Should Not Independently Own Provider Credentials
- Lecture: AI Gateway vs. API Gateway vs. Model Router: Defining the Control Plane
- Lecture: LiteLLM Core Architecture & OSS vs. Enterprise Feature Boundaries
- Lecture + Lab: Deploying LiteLLM Proxy via Docker, Docker Compose, and Helm
- Lecture + Lab: Database Initialization (PostgreSQL) for Persistent State & Audit Logs
- Lecture + Lab: Centralizing API Keys & Sending Multi-Backend OpenAI-Compatible Requests

2. Identity, Virtual Keys, Budgets & Model Access Policy

- Lecture: Gateway Virtual Keys vs. Provider Credentials: Workload Identity
- Lecture: Multi-Tenant Governance: Users, Teams, Organizations, and Projects
- Lecture: Defining Model Access Policies (Restricting Expensive/Experimental Models per Team)
- Lecture: Rate Limiting (TPM/RPM) and Spend Tracking Architecture
- Lecture + Lab: Issuing Virtual Keys with Scoped Model Access and Budget Limits
- Lecture + Lab: Deliberately Triggering Rate Limit and Budget Exceeded Exceptions

3. Routing, Resilience, Caching & Guardrails

- Lecture: Load Balancing & Priority Routing Across Identical and Cross-Provider Models
- Lecture: Failovers, Cooldowns, Retries, and Provider-Specific Semantic Differences
- Lecture: What LiteLLM Cannot Normalize: Reasoning Controls, Tool Schema Nuances, and Structured Output Variance
- Lecture: Caching Strategies: Exact Match, Prompt Caching, and Redis Vector Caching
- Lecture + Lab: Configuring Cross-Provider Fallback Chains and Simulating Outages
- Lecture + Lab: Deploying Redis Semantic Caching & Implementing PII Masking via Microsoft Presidio

4. Observability & Agent/MCP Tool Governance

- Lecture: The AI Gateway as a Centralized Telemetry Hub (Tokens, Costs, Latency, Errors)
- Lecture + Lab: Integrating Metrics with OpenTelemetry, Prometheus, and Langfuse
- Lecture: Agent Architecture Evolution: Model Authority vs. Tool Authority
- Lecture: Model Context Protocol (MCP) Gateway: Centralized Registration, OAuth Handling, and Tool Namespacing
- Lecture: Enforcing Guardrails and Scoped Entitlements on MCP Tool Executions
- Lecture + Lab: Capstone: Deploying an Application Route -> LiteLLM -> Multi-Model Failover + MCP Tool Execution with Active Guardrails

Prerequisites

- Basic familiarity with Docker, Redis, PostgreSQL, and RESTful APIs.
- Practical experience writing basic Python, Node.js, or Go.