Bifrost 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 Bifrost, the high-performance open-source AI Gateway built in Go by Maxim AI. Designed for microsecond-scale proxy overhead, Bifrost serves as a high-throughput control plane connecting applications and AI agents to 20+ model providers (including OpenAI, Anthropic, AWS Bedrock, Google Vertex AI, Azure OpenAI, Groq, DeepSeek, 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 via virtual keys, model access policies, high-availability routing and failovers, semantic caching, PII/secret guardrails, and governing AI agent tool authority using Bifrost's advanced Model Context Protocol (MCP) gateway, including tool filtering and MCP Code Mode.

Who Should Attend

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

What You'll Learn

- Control AI Spend & Latency: Reduce token expenses and lower response times by deploying semantic vector caching and multi-tenant virtual keys with strict budget caps and rate limits.
- Establish Governed AI Access: Centralize application and agent traffic behind Bifrost, applying consistent workload identity, model access rules, rate limits, guardrails, and telemetry.
- Build Provider Resilience: Configure automated multi-provider failover chains, circuit breakers, and load balancing across upstream model endpoints to maintain service stability during provider outages.
- Reduce Provider Coupling: Migrate compatible applications by redirecting supported SDK/API traffic to Bifrost while managing provider-specific model behavior at the gateway layer.
- Govern Agent Tool Authority & Code Mode (MCP): Centralize, authenticate, and restrict external tools using Bifrost's MCP gateway, leveraging MCP Code Mode to optimize agent tool context and reduce token overhead.
- Filter PII & Sensitive Content: Implement real-time PII redaction and secret detection guardrails on both LLM completions and MCP tool executions.
- Gain Comprehensive Audit Visibility: Track token consumption per virtual key, inspect full request logs, and stream operational metrics to Prometheus and OpenTelemetry collectors.
- Enforce Granular Access Policies: Scope virtual keys to restrict specific providers, models, budgets, rate limits (RPM), and MCP tool permissions per team, project, or agent workload.

Outline

1. Bifrost Architecture & Deployment

- Lecture: The Role of AI Gateways in High-Performance Production Infrastructure
- Lecture: Bifrost Core Architecture: Go Proxy Engine, Providers, Plugins, and Control Plane
- Lecture: Performance Architecture & Benchmark Methodologies (Published Microsecond-Scale Overhead)
- Lecture: Deployment Models: Local, Docker, Kubernetes, VPC, and Air-Gapped Environments
- Lecture + Lab: Deploying Bifrost Gateway via Docker / Bifrost CLI
- Lecture + Lab: Configuring OpenAI, Anthropic, and Local/Alternate Model Providers
- Lecture + Lab: Navigating the Bifrost Web Dashboard and Sending Multi-Provider Requests

2. Provider Routing, Virtual Keys & Resilience

- Lecture: OpenAI-Compatible Provider Abstraction Layer
- Lecture: API Compatibility vs. Model Equivalence (Handling Reasoning, Tool-Calling, and Output Variance)
- Lecture: Gateway Virtual Keys as Workload Identities vs. Raw Provider Credentials
- Lecture: Defining Provider and Model Access Policies (Restricting Experimental or High-Cost Models)
- Lecture: Weighted Load Balancing, Circuit Breakers, Retries, and Fallback Strategies
- Lecture + Lab: Migrating an OpenAI-Compatible Application to Bifrost (Updating Base URLs)
- Lecture + Lab: Creating Scoped Virtual Keys with Model/Provider Restrictions
- Lecture + Lab: Configuring Multi-Provider Fallback Chains and Simulating Upstream Outages

3. Cost Control, Caching & Guardrails

- Lecture: Token and Request Economics at the Gateway Layer
- Lecture: Virtual Key Budgeting and Rate Limiting (RPM Enforcement)
- Lecture: Exact Caching vs. Semantic Vector Caching
- Lecture: Semantic Similarity Thresholds and Cache Correctness Risks in Production
- Lecture: Request/Response Guardrail Architecture: PII Redaction and Secret Detection
- Lecture + Lab: Enforcing Virtual Key Budgets, Rate Limits, and Handling Exception Triggers
- Lecture + Lab: Configuring Semantic Caching and Benchmarking Response Speeds on Repeated Queries
- Lecture + Lab: Implementing Guardrails to Detect and Redact Sensitive Inputs/Secrets

4. MCP Gateway, Code Mode & Enterprise Observability

- Lecture: Bifrost as an MCP Gateway (Client + Server Integration, Tool Discovery, and Namespacing)
- Lecture: Governing Agent Authority: Virtual Key Tool-Level Filtering and OAuth Handling
- Lecture: Architectural Deep Dive: Standard MCP Tool Catalogs vs. Bifrost MCP Code Mode
- Lecture: Telemetry Architecture: OpenTelemetry Tracing and Prometheus Metric Exports
- Lecture: Architecture Spotlight: Endpoint Policy Extension via Bifrost Edge
- Lecture + Lab: Registering External MCP Tools and Enforcing Tool-Level Permissions per Virtual Key
- Lecture + Lab: Testing MCP Code Mode for Token Reduction in Multi-Tool Agent Workflows
- Lecture + Lab: Capstone Exercise: Operating the Gateway under Failure Conditions
- Scenario: Deploy an agent service through Bifrost configured with multi-provider failover (OpenAI $\rightarrow$ Anthropic), virtual key budgets, PII guardrails, and scoped MCP tool access (DB Read permitted, DB Write blocked).
- Operational Challenges: Ingest live traffic, simulate an upstream provider outage, inject a disallowed MCP tool call, trigger a PII violation, and verify policy enforcement and telemetry exports via Prometheus/OpenTelemetry.

Prerequisites

- Basic familiarity with Docker, RESTful APIs, and command-line interfaces.
- Practical experience writing basic Python, Node.js, or Go code.