Faysal Aziz

Advanced OpenClaw

Engineer OpenClaw beyond the defaults - 1 day

FA-0032 · 1 day

Agentic & Generative AI · Software Development

Introduction

Running OpenClaw is one thing. Engineering it into a controlled technical environment is another.

For advanced practitioners, the interesting part of OpenClaw begins where basic installation and configuration end: connecting agents to private infrastructure, choosing and tuning model-serving architectures, writing custom capabilities, controlling tool execution, designing isolation boundaries, debugging model/tool interactions, and deciding which workloads should remain entirely inside the organization's network.

This becomes particularly important with **on-premises LLMs**. OpenClaw currently supports local and self-hosted inference through Ollama, vLLM and SGLang, as well as LM Studio and custom OpenAI-compatible endpoints. Its documentation specifically positions vLLM, SGLang and similar servers as options for higher-throughput self-hosted inference, while Ollama provides a comparatively straightforward operational path for local model deployment.

An on-prem deployment, however, is not simply a matter of replacing a cloud API URL with a local IP address. Agentic workloads depend heavily on **context capacity, reliable structured tool calling, inference performance and model capability**. OpenClaw's current guidance warns that smaller or aggressively quantized local models can create problems with context handling and prompt-injection resistance. The choice of model and serving infrastructure therefore becomes part of the agent's security and reliability architecture.

This intensive one-day course approaches OpenClaw from that engineering perspective. Participants will work at the boundary between **OpenClaw, application code, APIs, Linux, containers, GPU-hosted inference and enterprise security controls**. The focus includes private LLM connectivity, OpenAI-compatible endpoints, Ollama and vLLM, custom skills, programmatic tools and plugins, multi-agent design, sandboxing, permissions, troubleshooting and production architecture.

This is deliberately **not an introductory OpenClaw course and not a no-code AI workshop**. Coding is mandatory. Participants who cannot comfortably read, modify and troubleshoot application code, JSON/JSON5 configuration, API requests and Linux command-line operations will not have the necessary foundation for the course.

The instructor has **over 30 years of industry experience**, and the material is built around real industry-demanded engineering skills rather than an academic treatment of AI agents. The objective within a realistic one-day timeframe is depth in the architectural components that matter most—not superficial coverage of every OpenClaw feature.

Learning outcomes

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

- Design an advanced OpenClaw deployment architecture.
- Integrate OpenClaw with on-premises LLM infrastructure.
- Configure Ollama and vLLM-based inference environments.
- Connect OpenClaw to OpenAI-compatible private model endpoints.
- Evaluate local LLM suitability for agent and tool-calling workloads.
- Develop custom OpenClaw skills and capabilities.
- Integrate agents with internal APIs and applications.
- Implement programmatic tools for agent execution.
- Design controlled multi-agent environments.
- Configure model routing and fallback strategies.
- Apply sandboxing and least-privilege execution controls.
- Assess an OpenClaw architecture for production deployment.

Prerequisites

This is an advanced technical course. The following prerequisites are mandatory.

Participants must have:

- **Practical programming experience; coding is mandatory.**
- Ability to write and troubleshoot **Python and/or JavaScript/TypeScript**.
- Strong command-line proficiency on Linux.
- Working knowledge of shell scripting.
- Practical understanding of REST APIs.
- Ability to read and construct JSON/JSON5 payloads.
- Understanding of HTTP methods, headers, authentication and status codes.
- Working knowledge of Git and source-code repositories.

- Practical Docker/container experience.
- Understanding of environment variables and secrets.
- Basic TCP/IP, DNS, ports and firewall knowledge.
- Working understanding of LLMs, tokens and context windows.
- Familiarity with AI inference APIs and tool/function calling.
- Previous OpenClaw experience equivalent to the introductory technical course.
- Administrator/root or equivalent access to the training environment.

Participants without coding, Linux command-line and API experience should not attend this advanced course.

Training outline

1. Advanced OpenClaw Architecture

1. Gateway architecture
2. Agent runtime
3. Model-provider layer
4. Tool execution pipeline
5. Skills architecture
6. Plugin architecture
7. Sessions and workspaces
8. Execution boundaries
9. Local, remote and hybrid deployment models

2. Advanced Configuration Engineering

1. Configuration hierarchy
2. Environment-based configuration
3. Agent-specific configuration
4. Provider configuration
5. Model configuration
6. Model aliases
7. Primary and fallback models
8. Secrets handling
9. Configuration validation

3. On-Premises LLM Fundamentals

1. Local inference architecture
2. CPU versus GPU inference
3. GPU memory requirements
4. Model size considerations
5. Quantization
6. Context-window requirements
7. Tool-calling capability
8. Reasoning model considerations
9. Latency and throughput
10. Concurrent inference

4. Ollama with OpenClaw

1. Ollama server architecture
2. Model acquisition and management
3. Local model serving
4. Remote Ollama servers
5. OpenClaw provider configuration
6. Model discovery
7. Model selection
8. Tool-capable models
9. Context configuration
10. Connectivity troubleshooting

5. vLLM with OpenClaw

1. vLLM architecture
2. GPU-backed model serving
3. OpenAI-compatible API
4. Model deployment
5. /v1/models
6. Chat-completion endpoints
7. Authentication
8. OpenClaw provider configuration
9. Model discovery

10. Multi-GPU considerations
11. Throughput and concurrency

6. Private OpenAI-Compatible LLM Endpoints

1. Provider abstraction
2. Custom base URLs
3. Authentication
4. Model identifiers
5. Chat-completion APIs
6. Responses APIs
7. Internal proxy architectures
8. TLS considerations
9. Endpoint validation
10. Failure handling

7. Hybrid LLM Architectures

1. Local-first inference
2. Cloud fallback
3. Agent-specific model selection
4. Task-based model routing
5. Private versus public workloads
6. Performance versus privacy
7. Cost versus capability
8. Failure and fallback strategies

8. Coding Custom Agent Capabilities

1. Programmatic tool architecture
2. Tool schemas
3. Input validation
4. Structured outputs
5. Python-based integrations
6. JavaScript and TypeScript integrations
7. REST API integration
8. Internal service integration

9. Error handling
10. Timeout handling

9. Advanced Skills Development

1. Skill structure
2. Skill metadata
3. Instruction design
4. Tool-aware skills
5. Script-backed skills
6. Dependency management
7. Environment requirements
8. Skill precedence
9. Skill testing
10. Version control

10. Advanced Tool Execution

1. Tool profiles
2. Tool allowlists
3. Tool denylists
4. Command execution
5. Filesystem operations
6. Process execution
7. Network-access controls
8. Execution approvals
9. Elevated execution
10. Tool failure diagnostics

11. Multi-Agent Engineering

1. Agent specialization
2. Agent-specific models
3. Agent-specific workspaces
4. Agent-specific tools
5. Agent delegation
6. Sub-agents

7. Inter-agent communication
8. Execution isolation
9. Resource boundaries

Disclaimer

This course outline is intended to serve as a professional training framework and general guide to the anticipated scope of instruction. The trainer reserves the right, at their professional discretion and without prior notice, to amend, reorganize, consolidate, substitute, expand or omit individual topics where necessary to accommodate participant competency, available infrastructure, hardware capabilities, software or model changes, security requirements, time constraints or evolving industry practices while preserving the intended learning objectives of the course.

© 2026 Faysal Aziz · Kuala Lumpur