The Journal of Open Source Software

OGX: An Open-Source, Vendor-Neutral Generative AI Application Server

Francisco Javier Arceo[1¶], **Sébastien Han**[1], **Matthew Farrellee**[3], **Charlie Doern**[1], **Yuan Tang**[1], **Derek Higgins**[1], **Varsha Prasad Narsing**[1], **Gordon Sim**[1], **Sumanth Kamenani**[1], **Ben Browning**[1], **and Raghotham Murthy**[2]

1 Red Hat AI, USA **2** Meta, USA **3** Independent ¶ Corresponding author

DOI: 10.21105/joss.11234

Software

- Review
- Repository
- Archive

Editor: Fangzhou Xie

Reviewers:

- @showdawn
- @alexzerg

Submitted: 05 June 2026
Published: 03 November 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

OGX (Open GenAI Stack), formerly Llama Stack (Llama Stack Contributors, 2025), is an open-source AI application server and Python library with interchangeable backend providers (OGX Contributors, 2026b). Teams building retrieval-augmented generation (RAG) pipelines, conversational agents, and tool-calling workflows can change their inference engine and vector database through server configuration without rewriting application code.

OGX implements OpenAI, Anthropic, and Google APIs, with the Responses API for server-side orchestration as its primary focus (Open Responses Community, 2026). The cited v1.5.0 release supports configurable inference and vector storage providers. A companion Kubernetes Operator (OGX Contributors, 2026a) manages deployments, including shared and per-tenant instances. Together, OGX and its operator provide a self-hosted backend for AI-powered developer tools such as Claude Code, Codex CLI, OpenCode, and OpenHands.

Statement of Need

AI application development today is tightly coupled to proprietary API providers. While inference-only workloads can increasingly be swapped across providers—vLLM (Kwon et al., 2023), for example, supports the Responses API for basic inference—applications that rely on the full stack (retrieval, tool calling, conversation state, safety guardrails) remain difficult to migrate without rewriting significant application logic. This coupling limits reproducibility, makes comparisons across model providers difficult, and prevents teams from running AI workloads on controlled infrastructure—a requirement in regulated, privacy-sensitive, and air-gapped environments.

OGX addresses this need through standard APIs and configurable infrastructure. Researchers can compare backends while keeping retrieval, tool authorization, and conversation handling in one server. This separates the choice of client software development kit (SDK), model, and deployment environment, supporting reproducible configurations on controlled infrastructure.

State of the Field

OGX continues Llama Stack under a renamed, model-agnostic mission. Its server-side API layer complements inference engines, gateways, and client-side frameworks, which address different parts of the deployment problem.

Inference engines (vLLM (Kwon et al., 2023), SGLang (Zheng et al., 2024), Ollama (Ollama Contributors, 2026)) focus on model inference and serving. Some expose tool-calling APIs;

applications using them directly still need to arrange retrieval, persistent conversation state, and tool execution for agentic workflows. OGX supplies these application services around interchangeable inference engines.

API gateways (LiteLLM (BerriAI, 2026), OpenRouter (OpenRouter, 2026)) provide unified access and routing across inference providers. Their feature sets also include capabilities beyond request translation, such as LiteLLM's guardrails and Model Context Protocol (MCP) gateway. OGX combines provider routing with its own vector stores, persistent conversations, and server-side Responses orchestration in a self-hosted application server.

Client-side frameworks (LangChain (LangChain, Inc., 2023), LangGraph (LangChain, Inc., 2024), LlamaIndex (LlamaIndex, 2022), CrewAI (CrewAI, Inc., 2024), Haystack (deepset, 2023)) provide rich developer abstractions for building agents and RAG pipelines. However, they execute orchestration client-side, distributing security-critical logic across application code. These frameworks are complementary to OGX: they compose agent logic while OGX provides the server-side execution target they call into.

Proprietary platforms (OpenAI's Responses API (OpenAI, 2025), Databricks Mosaic AI (Databricks, 2025)) offer integrated experiences but couple applications to a specific vendor's infrastructure and pricing.

OGX integrates inference, retrieval, tool execution, conversation management, and safety behind Open Responses-compatible APIs (Open Responses Community, 2026). Combining these responsibilities in a server addresses a different trust boundary from extending a client-side framework: applications share centralized policy enforcement and provider configuration while retaining their own agent logic.

Software Design

Provider Architecture

OGX's core abstraction is the pluggable provider. Provider-backed capabilities (inference, vector storage, tool runtime, file processing) are defined by Protocol interfaces in the lightweight `ogx-api` package, allowing third-party providers to implement the contract without depending on the full server. Concrete providers implement these interfaces for specific backends: `remote::openai` and `remote::anthropic` for hosted APIs, `remote::vllm` for self-hosted GPU inference, `inline::faiss` and `remote::pgvector` for vector search, and so on. A routing layer dispatches requests to provider instances based on logical resource identifiers, enabling multiple providers to serve the same API simultaneously—for example, Ollama handling local models while OpenAI handles hosted models, both behind `/v1/chat/completions`.

A *distribution* packages a specific set of providers and configuration into a deployable unit, decoupling application logic from infrastructure selection. This design favors reproducible configuration over ad hoc application code: teams can prototype with lightweight local backends (Ollama, sqlite-vec (Garcia, 2024)) and deploy to production backends (vLLM, pgvector) by changing the distribution, not the application.

Content moderation uses a separate integration path. Optional Responses guardrails send input and generated text to a server-configured, OpenAI-compatible `moderation_endpoint`. A request enables these checks with `guardrails=true`; there is no Safety provider protocol. Changing the moderation service requires a compatible endpoint and updated server configuration.

Dual Deployment Model

OGX runs in two modes. **Server mode** exposes HTTP endpoints accessible from any language or tool. **Library mode** runs in process through direct Python import, avoiding client/server

HTTP overhead and suiting notebooks and scripts. Embedding-model initialization may still download model assets or check Hugging Face metadata, even for cached models; offline use requires the assets to be cached and `HF_HUB_OFFLINE=1`. Both modes use identical provider routing and API semantics.

Multi-SDK Compatibility

OGX serves three client SDK protocols from a single server. The stable **OpenAI-compatible endpoints** (`/v1/chat/completions`, `/v1/responses`, `/v1/vector_stores`) are the primary interface and the Responses API implementation conforms to the Open Responses specification (Open Responses Community, 2026). The **Anthropic Messages endpoint** (`/v1/messages`) is also stable. The **Google GenAI Interactions endpoint** (`/v1alpha/interactions`) is experimental and may change incompatibly between releases. This portability comes with a trade-off: OGX must normalize provider-specific behavior into API contracts while still exposing enough backend-specific configuration for real deployments.

Server-Side Agentic Orchestration

The Responses API implements server-side agentic orchestration: the inference-tool-inference loop executes within the server process, not the client (OpenAI, 2025). This centralizes security enforcement, tool authorization, and conversation state management, at the cost of moving some flexibility from application code into server configuration. Built-in tools include file search (RAG over vector stores), web search, code interpretation, and MCP (Anthropic, 2026) integration for external tool servers.

Unlike inference engines and API gateways that treat requests as stateless, OGX manages state natively: the Conversations API persists multi-turn history with tenant-scoped isolation, the Prompts API provides versioned prompt templates, and a resource registry tracks models, vector stores, and files as first-class server objects. A Compaction API summarizes long histories to manage context window limits. Telemetry is built on OpenTelemetry (OTEL) with MLflow (Zaharia et al., 2018) tracing integration for logging spans, tool calls, and retrieval steps to existing ML experiment tracking infrastructure. This state and observability layer is what makes OGX a complete application server rather than a stateless proxy.

For multitenant deployments, OGX provides attribute-based access control (ABAC) that enforces tenant isolation at the retrieval, tool execution, state management, and API routing layers. The security properties of this architecture have been formally analyzed and empirically validated (Arceo & Narsing, 2026).

Kubernetes Operator

The OGX Kubernetes Operator (OGX Contributors, 2026a) provides declarative deployment through the `OGXServer` custom resource. A single custom resource specifies the distribution, replica count, storage, inference backend, and network policies. The operator manages full lifecycle reconciliation with support for both vanilla Kubernetes and OpenShift, ConfigMap-driven image overrides for fleet-wide updates, multi-architecture builds (amd64/arm64) with images compliant with Federal Information Processing Standards (FIPS), and shared instances with ABAC isolation, per-tenant namespace isolation, and hybrid topologies.

Example Usage

This RAG example uses the standard OpenAI SDK against OGX, with inference and vector storage selected through server configuration:

```
from openai import OpenAI
```

```python
client = OpenAI(base_url="http://localhost:8321/v1", api_key="unused")

# Create a vector store and upload documents
vector_store = client.vector_stores.create(name="docs")
with open("manual.pdf", "rb") as file:
    client.vector_stores.files.upload(vector_store_id=vector_store.id, file=file)

# Query with server-side RAG via the Responses API
response = client.responses.create(
    model="meta-llama/Llama-3.2-3B-Instruct",
    input="What are the installation requirements?",
    tools=[{"type": "file_search", "vector_store_ids": [vector_store.id]}],
)
print(response.output_text)
```

Switching from Ollama to vLLM changes the server configuration while preserving this client code. Published tutorials cover IBM watsonx.ai with Milvus (IBM Community, 2025) and Oracle Cloud Infrastructure with OCI AI Blueprints (Gowda & Kennetz, 2026).

Research Impact Statement

OGX has realized impact through both public deployments and customer production use. Publicly documented examples under the former Llama Stack name include an intelligent OpenShift operations agent combining RAG, web search, and MCP tool integration for automated incident response (Red Hat, 2025), enterprise RAG pipelines on IBM watsonx.data with Milvus (IBM Community, 2025), and generative AI application development on Oracle Cloud Infrastructure (Gowda & Kennetz, 2026). Beyond these publicly documented cases, maintainers are also aware of additional production use by enterprise customers in sectors including telecommunications, semiconductor manufacturing, financial services, insurance, and consulting; these engagements are confidential and not independently documented, so we report them here as self-reported maintainer knowledge rather than citable evidence. The framework has been presented at Meta Connect (Meta, 2024) and IBM TechXchange (Clyburn, 2025) as a standardization layer for enterprise AI applications.

The security architecture—specifically, the multitenant isolation model combining ABAC-gated retrieval, server-side orchestration, and pluggable provider backends—was formally analyzed in a peer-reviewed publication at the ACM Conference on AI and Agentic Systems (Arceo & Narsing, 2026). OGX conforms to the Open Responses specification (Open Responses Community, 2026) and serves as a reference implementation for open, vendor-neutral agentic AI APIs.

As of June 2026, the project has over 8,400 GitHub stars, 242 contributors, 4,000 commits, and 68 releases across nearly two years of public development. Community engagement includes weekly contributor calls, an active Discord server, and integrations contributed by external organizations including Red Hat, IBM, Oracle, and Infinispan.

AI Usage Disclosure

Generative AI tools, including GitHub Copilot, Anthropic Claude, and OpenAI Codex (GPT-6), were used for code completion, documentation drafting, and paper drafting. Assistance was limited to generating candidate text or code that human contributors reviewed, edited, tested, and validated. Core architectural decisions, API design, the security model, and final paper content were made by human authors.

Acknowledgements

We thank Meta for creating and open-sourcing Llama Stack, now renamed OGX. We are grateful to Red Hat for supporting the development of OGX through employee time and infrastructure support. We thank the OGX contributor community for their sustained contributions to the project.

References

Anthropic. (2026). *Model Context Protocol specification*. https://modelcontextprotocol.io/specification/

Arceo, F. J., & Narsing, V. P. (2026). Securing the agent: Vendor-neutral, multitenant enterprise retrieval and tool use. *Proceedings of the ACM Conference on AI and Agentic Systems*, 862–872. https://doi.org/10.1145/3786335.3813145

BerriAI. (2026). *LiteLLM: An open-source AI gateway*. https://github.com/BerriAI/litellm

Clyburn, C. (2025). *Llama Stack: Kubernetes for RAG and AI agents in generative AI*. https://mediacenter.ibm.com/media/Llama+Stack+Kubernetes+for+RAG+AI+Agents+in+Generative+AI/1_xl78upq2

CrewAI, Inc. (2024). *CrewAI: Framework for orchestrating role-playing autonomous AI agents*. https://github.com/crewAIInc/crewAI

Databricks. (2025). *Mosaic AI Agent Framework and Mosaic AI Agent Evaluation are generally available (GA)*. https://docs.databricks.com/aws/en/release-notes/product/2025/march#mosaic-ai-agent-framework-and-mosaic-ai-agent-evaluation-are-generally-available-ga

deepset. (2023). *Haystack: End-to-end LLM framework for building production-ready applications*. https://github.com/deepset-ai/haystack

Garcia, A. (2024). *sqlite-vec: A vector search SQLite extension*. https://github.com/asg017/sqlite-vec

Gowda, A., & Kennetz, D. (2026). *Accelerating enterprise gen AI applications development on OCI with Llama Stack and OCI AI Blueprints*. https://blogs.oracle.com/ai-and-datascience/accelerating-enterprise-gen-ai-applications-development-on-oci-with-llama-stack-and-oci-ai-blueprints

IBM Community. (2025). *Build RAG with Llama Stack and watsonx.data Milvus*. https://community.ibm.com/community/user/blogs/divya13/2025/05/08/build-rag-with-llama-stack-and-watsonxdata-milvus

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. *Proceedings of the 29th ACM SIGOPS Symposium on Operating Systems Principles*, 611–626. https://doi.org/10.1145/3600006.3613165

LangChain, Inc. (2023). *LangChain: Build context-aware reasoning applications*. https://github.com/langchain-ai/langchain

LangChain, Inc. (2024). *LangGraph: Build resilient language agents as graphs*. https://github.com/langchain-ai/langgraph

Llama Stack Contributors. (2025). *Llama Stack*. https://github.com/llamastack/llama-stack

LlamaIndex. (2022). *LlamaIndex: Data framework for LLM applications*. https://github.com/run-llama/llama_index

Meta. (2024). *Llama Stack: Chapter one*. https://developers.facebook.com/m/meta-connect-

developer-sessions/llama-stack-chapter-one/

OGX Contributors. (2026a). *OGX Kubernetes Operator*. https://github.com/ogx-ai/ogx-k8s-operator/tree/7fa16532e1434bf74493ca305b1e21030914ae57

OGX Contributors. (2026b). *OGX (Open GenAI Stack)*. https://github.com/ogx-ai/ogx/releases/tag/v1.5.0

Ollama Contributors. (2026). *Ollama*. https://github.com/ollama/ollama

Open Responses Community. (2026). *Open Responses specification*. https://www.openresponses.org/

OpenAI. (2025). *Responses API reference*. https://platform.openai.com/docs/api-reference/responses

OpenRouter. (2026). *OpenRouter API documentation*. https://openrouter.ai/docs/quickstart

Red Hat. (2025). *Generative AI applications with Llama Stack: A notebook-guided journey to an intelligent operations agent*. https://www.redhat.com/en/blog/generative-ai-applications-llama-stack-notebook-guided-journey-intelligent-operations-agent

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., & Zumar, C. (2018). Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Engineering Bulletin*, *41*(4), 39–45. https://people.eecs.berkeley.edu/~matei/papers/2018/ieee_mlflow.pdf

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., & Sheng, Y. (2024). SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems*, *37*. https://doi.org/10.52202/079017-2000