Securing the Agent: Vendor-Neutral, Multitenant Enterprise Retrieval and Tool Use

Varsha Prasad
Red Hat AI
Boston, USA
vnarsing@redhat.com

Francisco Javier Arceo
Red Hat AI
Boston, USA
farceo@redhat.com

Abstract

Retrieval-Augmented Generation (RAG) and agentic AI systems are increasingly prevalent in enterprise AI deployments. However, real enterprise environments introduce challenges largely absent from academic treatments and consumer-facing APIs: multiple tenants with heterogeneous data, strict access-control requirements, regulatory compliance, and cost pressures that demand shared infrastructure.

A fundamental problem underlies existing RAG architectures in these settings: dense retrieval ranks documents by semantic similarity, not by authorization—so a query from one tenant can surface another tenant's confidential data simply because it is the nearest neighbor. We formalize this gap and analyze additional failure modes—including tool-mediated disclosure, context accumulation across turns, and client-side orchestration bypass—that arise when agentic systems conflate relevance with authorization. To address these challenges, we introduce a layered isolation architecture combining policy-aware ingestion, retrieval-time gating, and shared inference, enforced through server-side agentic orchestration. Unlike client-side agent patterns, this approach centralizes tool execution, state management, and policy enforcement on the server, creating natural enforcement points for multitenant isolation and eliminating per-tenant infrastructure duplication.

An open-source, vendor-neutral implementation using Llama Stack—which realizes the Responses API paradigm with server-side multi-turn orchestration—and its Kubernetes Operator demonstrates that secure multitenancy, cost-efficient resource sharing, and the flexibility of modern agentic systems are simultaneously achievable on shared cluster infrastructure.

CCS Concepts

• **Computing methodologies** → *Machine learning*; • **Information systems** → Data management systems.

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

Llama Stack, Kubernetes operator, LLM systems, MLOps, LLMOps

ACM Reference Format:
Varsha Prasad and Francisco Javier Arceo. 2026. Securing the Agent: Vendor-Neutral, Multitenant Enterprise Retrieval and Tool Use. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

1.1 Motivation

Enterprise adoption of generative AI has evolved beyond simple prompt-response interactions toward agentic systems—AI applications that autonomously reason, use tools, retrieve information, and execute multi-step workflows to accomplish complex tasks [7, 12, 14]. This evolution reflects a fundamental shift: rather than treating large language models (LLMs) as sophisticated text generators, organizations now deploy them as reasoning engines capable of taking actions in the world.

An API-first paradigm for these systems is to unify multi-turn inference, tool use, and retrieval behind a single endpoint. OpenAI's Responses API has emerged as a de facto interface pattern for building such systems by providing a unified interface for chat-style inference, tool invocation, retrieval tools, and stateful workflows [10]. In parallel, open implementations increasingly expose OpenAI-compatible endpoints to decouple applications from a specific model provider.

However, real-world enterprise deployments differ sharply from the assumptions embedded in consumer-facing APIs and many academic prototypes. Enterprise environments exhibit characteristics that demand specialized architectural consideration:

- **Multiple tenants:** distinct business units, customers, or partners served from shared infrastructure, with strict isolation requirements.
- **Heterogeneous data:** document collections vary in format, sensitivity classification, and access requirements.
- **Strict access control:** regulatory frameworks require fine-grained governance with auditable access patterns.
- **Operational control:** visibility into agent behavior, tool execution sequences, and data access patterns is required for debugging and compliance, consistent with lessons from production ML engineering [4].

- **Vendor independence:** lock-in to a single AI provider creates business risk; enterprises require on-prem and hybrid options.

Naïve approaches to addressing these requirements replicate the entire agentic stack per tenant: separate vector stores, dedicated inference endpoints, and isolated tool configurations. This strategy incurs substantial costs — infrastructure scales linearly with tenants rather than with actual usage — and creates operational fragmentation that amplifies common ML systems maintenance risks [13].

1.2 Problem statement

This paper addresses a fundamental tension in enterprise agentic AI deployment:

> Enterprises require both shared infrastructure for cost efficiency and strict tenant isolation for security and compliance — while maintaining the flexibility of modern agentic architectures.

Standard agentic AI deployments exhibit assumptions incompatible with this requirement:

(1) **Client-side orchestration:** the application manages the inference–tool–inference loop, distributing security-critical logic to potentially untrusted clients and increasing operational complexity [4].
(2) **Homogeneous data access:** retrieval stacks assume uniform access to a corpus; dense retrieval methods (e.g., DPR) optimize similarity ranking rather than authorization [8].
(3) **Implicit trust boundaries:** tool execution is often treated as a capability extension without systematic verification of who may invoke tools or consume tool outputs, despite the centrality of tool use in modern agent designs [7, 12, 14].
(4) **Stateless isolation:** requests are treated independently, ignoring how conversation state and cached tool results can leak across boundaries; such hidden couplings are a classic source of ML systems fragility [13].

In multitenant settings, these assumptions create serious vulnerabilities. A document highly similar to a query may belong to a different tenant. A tool call may access resources outside the user's authorization scope. Conversation history may accumulate context that crosses security boundaries.

1.3 Contributions

This paper makes the following contributions:

(1) We formalize the problem of *multitenant enterprise RAG* under shared infrastructure, showing why semantic similarity alone (as used by dense retrievers and ANN indexes) is insufficient to enforce isolation and authorization [6, 8].
(2) We analyze failure modes of existing RAG and agentic systems in multitenant settings, including cross-tenant retrieval leakage, unauthorized context construction, and policy violations arising from client-side orchestration.
(3) We propose a layered isolation architecture for distributed-scale enterprise RAG that combines policy-aware ingestion, retrieval-time gating, and shared inference to achieve strict tenant isolation without per-tenant duplication of storage or models.
(4) We introduce server-side orchestration as a unifying enforcement layer that centralizes retrieval, tool execution, and state management, reducing the trusted computing base for secure multitenant RAG and aligning with production ML engineering best practices [4, 13].
(5) We present an open-source, vendor-neutral framework based on Llama Stack [1] and Kubernetes deployment via an operator [2, 5] that enables pluggable models, vector stores, and tools, demonstrating the feasibility of secure multitenant RAG on shared infrastructure.

Primary sources. Llama Stack repository: https://github.com/llamastack/ [1].
Kubernetes operator repository: https://github.com/llamastack/llama-stack-k8s-operator [2].

2 Background and Related Work

2.1 Agentic AI evolution and multitenancy challenges

LLM application architectures evolved from simple completion APIs to agentic systems that autonomously use tools, retrieve information, and execute multi-step workflows [7, 12, 14]. The Responses API paradigm [10] unifies chat, retrieval, and tool use but assumes single-tenant deployment.

Multitenancy in agentic AI differs from traditional database or ML serving patterns. Beyond data isolation, agentic systems require policy-aware tool execution, stateful conversation management, and orchestration controls. Production ML systems require systematic lifecycle management [3] and experience operational fragility when security invariants are distributed across components [4, 13]—a problem amplified in agentic deployments where reasoning, retrieval, and tool execution interleave autonomously.

3 Problem Definition: Multitenant Enterprise Agentic AI

3.1 System model and threat analysis

We formalize the multitenant agentic AI environment: tenants (T) share infrastructure, each with associated data, users, tools, and policies. Users possess attributes (roles, clearances, group memberships). Documents live in vector stores with metadata (tenant ownership, classification, access policy). Agent execution follows the tool-using agent pattern [7, 14]: $E = [(i_1, \phi_1, r_1), (i_2, \phi_2, r_2), \ldots, (i_n, \emptyset, r_n)]$.

Key threats include: cross-tenant data leakage via retrieval, tool-mediated disclosure, context accumulation attacks, client-side orchestration bypass, and audit failure. Client-side orchestration is fundamentally unsuitable for enterprise multitenancy because it distributes trust to every

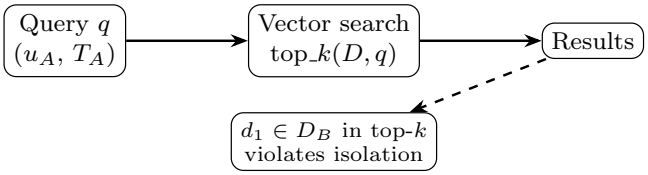

Figure 1: Similarity-only retrieval: top-k from shared corpus D can include documents from another tenant ($d_1 \in D_B$), violating isolation.

client, enabling fabricated tool results, unauthorized context injection, and manipulation of conversation state.

4 Multitenant Agentic AI Challenges and Solutions

Standard agentic AI deployments exhibit fundamental incompatibilities with enterprise multitenancy. We analyze these failure modes and propose a layered isolation architecture that addresses them.

4.1 Similarity-authorization gap in retrieval

Every major RAG framework retrieves documents by semantic similarity using dense retrievers such as DPR [8] and ANN indexing [6]. Similarity is orthogonal to authorization: a document from tenant B may be the most semantically similar result for tenant A's query, yet returning it violates isolation.

Formally, for tenants T_A and T_B sharing corpus $D = D_A \cup D_B$, the isolation invariant requires that only documents with $\text{tenant}(d) = \text{tenant}(u_A)$ be returned. For any similarity threshold θ, there exist queries q such that $\arg\max_{d \in D} \text{sim}(q, d) \in D_B$. Thus similarity-based retrieval cannot enforce tenant isolation without an authorization predicate: $\mathcal{R}(q, u) = \{d \in D : \text{sim}(q, d) > \theta \wedge P(u, d) = \text{permit}\}$.

4.2 Tool-mediated disclosure and client-side orchestration risks

Agentic systems executing tool sequences [12, 14] face unique multitenant threats: **Tool-mediated disclosure:** Agents invoke tools with agent credentials rather than end-user authorization, potentially accessing unauthorized data across tenant boundaries. **Context accumulation:** Multi-turn conversations persist context without per-turn policy re-validation, enabling cross-tenant data leakage. **Client-side bypass:** When orchestration runs client-side, malicious clients can skip authorization checks, manipulate tool invocation, or extract unauthorized data.

These failures stem from distributing security-critical logic outside the trust boundary, consistent with production ML system fragility patterns [4, 13].

Failure mode	Enforcement point
Cross-tenant retrieval leakage	Layer 2 retrieval gating (ABAC + metadata filters)
Context accumulation	Tenant-scoped state storage and per-turn authorization
Tool-mediated disclosure	Server-side tool execution with authorization propagation
Client-side bypass	Server-side orchestration (reduced TCB)
Audit failure	Server-side telemetry and tracing

Table 1: Mapping from failure modes to architectural enforcement points.

4.3 Layered isolation architecture

We propose a three-layer solution: policy-aware ingestion, retrieval-time gating, and shared inference.

Layer 1: Policy-aware ingestion. Tenant metadata must be attached at document ingestion time. An ingestion function $\mathcal{I}(d, t) \to D_t$ tags document d with tenant t's attributes so that every chunk inherits ownership metadata.

Layer 2: Retrieval gating. Two-tier enforcement: (1) resource-level authorization before search, and (2) chunk-level metadata filtering after retrieval. This composes similarity search with mandatory authorization predicates, separating semantic ranking from access control.

Layer 3: Shared inference. The LLM layer is shared across tenants, with isolation enforced at input construction. Since layers 1–2 ensure only authorized content enters prompts, inference can be safely shared, reducing cost from $O(N \cdot M)$ to $O(M)$ for N tenants and M model endpoints.

4.4 Server-side orchestration as enforcement layer

Server-side orchestration centralizes retrieval, tool execution, and state management within the trust boundary, eliminating client-side bypass risks. This design aligns with tool-using agent formulations [7, 14] while enabling centralized policy enforcement, audit logging, and state isolation.

Table 1 maps failure modes to enforcement points.

5 Reference Implementation: Llama Stack

Llama Stack [1] provides an open-source, vendor-neutral implementation of the proposed architecture. Unlike client-side orchestration libraries, Llama Stack executes the complete agentic control loop on the server, creating natural enforcement points for multitenancy and security [10].

The framework implements a provider abstraction enabling hybrid deployments: inference providers (vLLM [9], Ollama, OpenAI), vector stores (Chroma, pgvector, Elasticsearch), and tool runtimes (file_search, web_search, MCP). Together

with open models such as gpt-oss [11], this provides a complete open-source alternative to proprietary agentic AI platforms.

5.1 Multitenant deployment patterns

The Llama Stack Kubernetes Operator [2] supports multiple deployment topologies: **Shared server:** Single instance with logical isolation via ABAC policies and metadata-gated retrieval. **Per-tenant instances:** Separate deployments with namespace isolation and shared model backends. **Hybrid:** Mixed deployment balancing cost efficiency with stronger isolation for high-security tenants.

Authorization is enforced at API routes, routing resolution, and tool execution, with tenant-scoped state storage and per-turn policy checks preventing cross-tenant leakage.

6 Kubernetes Deployment

The Llama Stack Kubernetes Operator [2] automates deployment on shared cluster infrastructure [5]. The operator uses a `LlamaStackDistribution` custom resource to declaratively manage Llama Stack server deployments, supporting multiple topologies: shared instances with logical isolation, per-tenant deployments with namespace isolation, and hybrid approaches balancing cost with security requirements. The operator provides OpenAI-compatible endpoints and optional `NetworkPolicy` resources for defense-in-depth, layering network-level segmentation on top of application-level ABAC.

7 Analysis and Discussion

The layered isolation architecture addresses the failure modes identified above. Cross-tenant retrieval leakage is mitigated by composing similarity retrieval with mandatory policy predicates [6, 8]. Tool-mediated disclosure is reduced through centralized, policy-gated execution aligned with tool-using agent patterns [12, 14]. Context accumulation and state leakage are prevented by tenant-scoped storage and per-turn authorization [4, 13].

Performance depends on the inference backend [9, 15]; predicate pushdown into vector stores is desirable for scale but post-retrieval gating still enforces security invariants.

Limitations: ABAC policies can become complex at scale; metadata filtering timing varies by backend; model prior knowledge is orthogonal to RAG isolation; client-side function tools execute outside the server trust boundary.

8 Conclusion

Agentic AI systems introduce multitenancy, access control, and compliance challenges that exceed the assumptions in standard client-orchestrated patterns. We formalized the insufficiency of similarity-based retrieval for tenant isolation, grounded in dense retrieval and ANN vector search [6, 8], and analyzed failure modes of existing RAG and agentic frameworks. We proposed a layered isolation architecture combining policy-aware ingestion, retrieval-time gating, and shared inference, and argued that server-side orchestration is the unifying enforcement layer that centralizes retrieval, tool execution, and state management and reduces the trusted computing base. These design choices also reduce operational risk by turning implicit assumptions into enforced invariants, consistent with lessons from production ML systems engineering [4, 13].

Llama Stack [1] demonstrates that vendor-neutral, open-source tooling can support enterprise-grade multitenancy through provider abstraction, policy enforcement, and a server-side Responses API implementation [10]. Together with open models such as gpt-oss [11] and efficient serving via vLLM [9], this provides a complete open-source alternative to proprietary agentic AI platforms. The Llama Stack Kubernetes Operator [2] enables reproducible deployments on shared cluster infrastructure [5]. The patterns presented here—server-side orchestration, policy-aware retrieval, defense in depth, and operator-driven infrastructure—provide a foundation for secure, scalable agentic AI deployments that maintain enterprise control over autonomous agent behavior.

References

[1] [n. d.]. Llama Stack. GitHub repository. https://github.com/llamastack/ Accessed: 2026-01-28.

[2] [n. d.]. Llama Stack Kubernetes Operator. GitHub repository. https://github.com/llamastack/llama-stack-k8s-operator Accessed: 2026-01-28.

[3] [n. d.]. MLflow: A Machine Learning Lifecycle Platform. Open-source project. https://mlflow.org/ Accessed: 2026-02-23.

[4] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 291–300. doi:10.1109/ICSE-SEIP.2019.00042

[5] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *Commun. ACM* (2016). https://cacm.acm.org/practice/borg-omega-and-kubernetes/

[6] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-Scale Similarity Search with GPUs. *arXiv preprint arXiv:1702.08734* (2017). https://arxiv.org/abs/1702.08734

[7] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. 2022. MRKL Systems: A Modular, Neuro-Symbolic Architecture that Combines Large Language Models, External Knowledge Sources and Discrete Reasoning. *arXiv preprint arXiv:2205.00445* (2022). https://arxiv.org/abs/2205.00445

[8] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[9] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP)*. 611–626. doi:10.1145/3600006.3613165

[10] OpenAI. [n. d.]. Responses API Reference. Online documentation. https://platform.openai.com/docs/api-reference/responses Accessed: 2026-02-16.

[11] OpenAI. 2025. gpt-oss-120b & gpt-oss-20b Model Card. arXiv:2508.10925 [cs.CL] https://arxiv.org/abs/2508.10925

[12] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas

Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2302.04761

[13] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems*. 2503–2511. https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757b546b22a-Abstract.html

[14] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2210.03629

[15] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 521–538. https://www.usenix.org/conference/osdi22/presentation/yu