Nick Clark · Published April 25, 2026 · PDF

Ollama alternative: from local model runner to a governed agent-resident substrate

Ollama makes it straightforward to pull, run, and serve open language models on your own hardware, and it does that job well. But running a model locally is not the same as hosting a persistent, governed agent that owns its models as managed assets, accumulates an auditable lineage, and learns your body of work over time. That capability is the subject of the Agent-Resident Execution Substrate, disclosed in U.S. Provisional Application No. 64/070,239.

What Ollama Does

Ollama is an open-source tool for running large language models locally. It packages a model library, a pull-and-run workflow, and a local HTTP server that exposes a chat and completion API on the user's own machine. With a single command a user can download a quantized open-weight model, load it, and start generating text without sending prompts to a hosted service. Ollama supports a range of open models, exposes a Modelfile format for configuring system prompts and parameters, and offers adapter and quantization options that let sizable models run within consumer memory budgets.

Ollama is genuinely good at what it sets out to do. It lowers the barrier to local inference dramatically, it keeps prompt content on the device by default because inference happens locally, and it provides a clean, well-documented API surface that

many downstream tools build on. For a developer who wants a private, no-account, locally hosted inference endpoint, it is an excellent choice, and nothing below is a criticism of that design.

The point of comparison here is narrower and structural. Ollama is, by design, request-routing infrastructure: it loads models, unloads them, and routes requests to a local endpoint. The Agent-Resident Execution Substrate addresses a different layer of the stack, and the contrast is best understood as a difference in scope rather than a defect.

The Architectural Axis

The axis is this: what persistent entity, if any, sits above the models and owns them over time?

In a local model runner, the model registry is the durable object and the running process is stateless with respect to identity. Models are named entries you install, run, and remove. Between requests, the framework maintains configuration and loaded weights, but it does not maintain a continuous identity-bearing entity that accumulates a history of what it did, why, and with what outcome. When a model is replaced with a newer version, nothing structural carries forward except the user's own configuration files. There is no built-in record of which inference produced which result, whether that result was accepted or revised, or how those outcomes should shape future model selection or training.

This is the correct design for a model runner. It is deliberately thin. But it means the runner cannot, on its own, answer questions like: which entity is accountable for this device's inference behavior across its whole lifetime; what is the verifiable history of every model lifecycle operation; and how does the user's accumulated work become part of the model rather than being re-supplied as prompt context on every request. The substrate is built to answer exactly those questions.

How the Disclosed Approach Differs

The Agent-Resident Execution Substrate inverts the relationship between the model and the entity that runs it. Rather than treating the model registry as the durable object, the substrate makes a persistent semantic agent the durable object, and treats inference endpoints as managed assets subordinate to that agent.

Per the specification, the semantic agent comprises a persistent identity field, a cognitive state field, an append-only lineage field, and a governance policy field, and these persist continuously across the lifetime of the device. Inference endpoints live in a managed inference tool registry, each endpoint carrying a model artifact, an interface specification, and an associated governance scope. An agent-to-tool dispatcher routes requests to endpoints, and a tool lifecycle controller performs installation, retraining, replacement, archival, and removal, with every such operation governed by policy and recorded in the lineage field.

Several structural properties follow that a bare model runner does not provide:

A continuity guarantee. The spec describes the agent's identity, cognitive state, and lineage as preserved across all lifecycle operations applied to subordinate components, including replacement of every model artifact in the registry and update of the substrate runtime itself. The agent's identity is not dependent on any specific model artifact. Swapping models does not reset the entity above them.

An append-only, verifiable lineage. Each dispatch, outcome, lifecycle operation, ingestion event, and policy change is recorded as a lineage record chained to its predecessor under a continuity proof, such that the operational history is deterministically reconstructible and any alteration is detectable. This is the accountability layer that a stateless runner leaves to the surrounding tooling.

A personal corpus model. The spec discloses a model in the registry whose parameters are fine-tuned against artifacts the user authors, curates, or designates, so that inference reflects the user's terminology, structural conventions, and prior outputs without retrieval at inference time. As the user keeps authoring, artifacts enter the lineage, a corpus assembly module derives an incremental training set, a parameter-efficient fine-tuning module updates the weights within the local compute envelope, and a governed substitution promotes the updated artifact. The runner runs whatever weights you give it; the substrate closes the loop from your work back into the weights.

Governance and a privacy invariant. Lifecycle operations, dispatch, ingestion, and disclosure are each evaluated against cryptographically signed policy objects. The spec sets out a privacy invariant under which lineage records, model artifacts, training corpora, personal corpus model parameters, and counterparty records are not transmitted off-device except under an explicit disclosure policy object, with each disclosure recorded as an auditable event.

Cloud-burst forwarding. Where local endpoints lack capability or capacity, the spec describes a cloud-burst forwarding subsystem that selectively forwards a request to a remote endpoint only after passing capability, capacity, disclosure, and cost tests, with the forwarded payload treated as an off-device disclosure event under the privacy invariant. Off-device inference is the governed exception, not the default.

Where They Fit Together

These are complementary, not mutually exclusive. A local model runner and the substrate answer different questions: one is "how do I run this model here," the other is "what persistent, governed entity owns and improves the models over the device's lifetime."

In practice a substrate implementation can sit above a local runtime. The spec is explicit that endpoints are defined by a model artifact plus an interface specification and dispatch protocol, and that base models may be frozen while a personal corpus model specializes their output. A local runner that exposes open-weight models behind a clean local API is a reasonable way to host those artifacts; the substrate contributes the persistent agent identity, the lineage-driven retraining pipeline, the governance policy evaluation, and the privacy invariant around them. If your only need is to run a model locally, the runner alone is sufficient. If you need an accountable, self-improving, policy-governed agent, the substrate is the layer you add.

Boundary Conditions

An honest accounting of limits matters here. The substrate is disclosed in a provisional application; it describes an architecture and its embodiments, not a shipped consumer product, and provisional status reflects an early filing stage rather than an issued or examined claim set. The personal corpus model relies on well-understood parameter-efficient fine-tuning techniques such as low-rank adaptation, and continuous on-device retraining is bounded by the device's real memory, compute, power, and thermal budgets. On modest hardware the retraining window may be long and the corpus model's benefit incremental. The continuity, lineage, and governance guarantees are properties of a correct implementation of the disclosed mechanisms; they are engineering commitments, and their strength depends on the enforcement mechanisms actually built, including any hardware-anchored attestation.

Ollama, for its part, does not claim to be an agent platform, a governance layer, or a personalization pipeline, and it should not be judged as though it failed to be one. It is a focused, well-executed local model runner, and much of what the substrate composes could be built atop a runtime like it.

Disclosure Scope

The mechanisms attributed to the disclosed approach in this article trace to U.S. Provisional Application No. 64/070,239, which discloses the Agent-Resident Execution Substrate, including the persistent semantic agent and its identity, cognitive state, lineage, and governance policy fields; the managed inference tool registry and tool lifecycle controller; the lineage-driven retraining pipeline and personal corpus model; the governance policy framework and privacy invariant; and the cloud-burst forwarding subsystem. The description of Ollama and the surrounding market framing is provided as external context to locate the invention on a single architectural axis; it is not part of the filing and is not a claim of the application. Nothing here asserts that Ollama is defective, infringing, or deficient for its intended purpose; Ollama is a capable local model runner, and the comparison is limited to the structural difference between running models locally and hosting a persistent, governed agent that owns and improves them.

Agent-Resident Execution Substrate (/agent-resident-execution-substrate)

All 40 steps → (/inventive-steps)

Persistent execution environment carried by the agent, not the host — identity, state, and lineage across power cycles, devices, and upgrades.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- Agent-Resident Execution Substrate, Articles (/articles/agent-resident-execution-substrate)

SECONDARY TECHNICAL

- Persistent Semantic Agent (/articles/agent-resident-execution-substrate/persistent-semantic-agent)
- Managed Inference Tool Registry (/articles/agent-resident-execution-substrate/managed-inference-tool-registry)

- Agent-to-Tool Dispatcher (/articles/agent-resident-execution-substrate/agent-to-tool-dispatcher)
- Lineage-Derived Training Signal (/articles/agent-resident-execution-substrate/lineage-derived-training-signal)
- Identity Preservation Across Upgrades (/articles/agent-resident-execution-substrate/identity-preservation-across-upgrades)
- Cognitive State-Conditioned Dispatch (/articles/agent-resident-execution-substrate/cognitive-state-conditioned-dispatch)
- Governed Tool Lifecycle (/articles/agent-resident-execution-substrate/governed-tool-lifecycle)
- Continuity-Proof Lineage (/articles/agent-resident-execution-substrate/continuity-proof-lineage)
- Substrate Runtime Continuity (/articles/agent-resident-execution-substrate/substrate-runtime-continuity)
- Personal Corpus Model Training (/articles/agent-resident-execution-substrate/personal-corpus-model-training)
- Heterogeneous Inference Endpoints (/articles/agent-resident-execution-substrate/heterogeneous-inference-endpoints)
- Atomic Lifecycle Substitution (/articles/agent-resident-execution-substrate/atomic-lifecycle-substitution)
- Integrity Signal Feedback (/articles/agent-resident-execution-substrate/integrity-signal-feedback)
- Hardware-Bound Identity (/articles/agent-resident-execution-substrate/hardware-bound-identity)
- Cognitive State Append-Only Invariant (/articles/agent-resident-execution-substrate/cognitive-state-append-only-invariant)
- Counterparty Identity Records (/articles/agent-resident-execution-substrate/counterparty-identity-records)
- Privacy Egress-Controlled Disclosure (/articles/agent-resident-execution-substrate/privacy-egress-controlled-disclosure)
- Federated Cross-Device Agent Identity (/articles/agent-resident-execution-substrate/federated-cross-device-agent-identity)

APPLICATIONS · GENERAL

- Personal AI Agents That Survive Device Loss: One Continuous Identity and a Private Corpus Across Every Device (/articles/agent-resident-execution-substrate/personal-cross-device-agents)
- Enterprise Agent Fleets: Stable Agent Identity and Governed Tool Access Across Model Upgrades and Infrastructure Migration (/articles/agent-resident-execution-substrate/enterprise-agent-fleets)
- Audit-Grade Agent Identity for Regulated Finance and Healthcare: Continuity-Proof Lineage Across the Agent Lifecycle (/articles/agent-resident-execution-substrate/regulated-industry-agents)

- Edge and On-Device Agents: Hardware-Bound Identity Across Heterogeneous Inference Endpoints (/articles/agent-resident-execution-substrate/edge-and-on-device-agents)
- Agent-to-Agent Commerce With Counterparty Identity Records and Egress-Controlled Disclosure (/articles/agent-resident-execution-substrate/agent-to-agent-commerce)
- Governed Tool Lifecycles for Managed Inference-Provider Ecosystems: A Substrate Approach to Owning, Routing, and Retiring AI Tools (/articles/agent-resident-execution-substrate/managed-tool-ecosystems)
- Proving Unbroken Continuity in Long-Lived Autonomous Systems Across Substrate Migration and Atomic Model Substitution (/articles/agent-resident-execution-substrate/long-lived-autonomous-systems)
- Personal-Model Personalization: A User's Own Corpus-Internalized Model on the Agent-Resident Execution Substrate (/articles/agent-resident-execution-substrate/personal-model-personalization)
- On-Device Agent Identity for Robots and Autonomous Vehicles: An Auditable Substrate for Embodied Physical-World Agents (/articles/agent-resident-execution-substrate/embodied-physical-world-agents)

APPLICATIONS · SPECIFIC

- LangGraph Platform (LangChain) vs an agent-resident execution substrate: orchestration-graph state versus a portable, hardware-anchored agent runtime (/articles/agent-resident-execution-substrate/langgraph-platform)
- OpenAI AgentKit and the Assistants/Responses API vs agent-carried, hardware-anchored identity with governed tool lifecycle (/articles/agent-resident-execution-substrate/openai-agentkit)
- Microsoft Copilot Studio vs an agent-resident execution substrate: platform-hosted agent authoring versus portable, device-resident agent identity and continuity (/articles/agent-resident-execution-substrate/microsoft-copilot-studio)
- Google Vertex AI Agent Engine (managed runtime for deploying and scaling agents, with sessions/memory) vs an agent-carried, continuity-proofed identity substrate (/articles/agent-resident-execution-substrate/google-vertex-agent-engine)
- AWS Bedrock AgentCore (runtime, memory, identity, and gateway services for deploying agents at scale) vs an agent-resident execution substrate: where does the agent identity actually live? (/articles/agent-resident-execution-substrate/aws-bedrock-agentcore)
- Letta (formerly MemGPT) vs an append-only cognitive-state substrate: what a memory-management framework does not provide (/articles/agent-resident-execution-substrate/letta-memgpt)
- Cognition's Devin, an autonomous AI software-engineering agent vs a portable, continuity-proofed agent-resident runtime (/articles/agent-resident-execution-substrate/cognition-devin)

- Cloudflare Agents (Durable Objects) vs an agent-resident execution substrate: portable hardware-bound identity and continuity-proof lineage (/articles/agent-resident-execution-substrate/cloudflare-agents)
- **Ollama alternative: from local model runner to a governed agent-resident substrate (/articles/agent-resident-execution-substrate/ollama)**
- Apple Intelligence (on-device foundation models, Private Cloud Compute) vs a persistent agent-resident substrate: who owns identity, lineage, and the model? (/articles/agent-resident-execution-substrate/apple-intelligence)

Agent-Resident Execution Substrate overview → (/agent-resident-execution-substrate)