Nick Clark · Published April 25, 2026 · PDF

Google Vertex AI Agent Engine (managed runtime for deploying and scaling agents, with sessions/memory) vs an agent-carried, continuity-proofed identity substrate

Google Vertex AI Agent Engine is a managed cloud runtime for deploying, scaling, and operating agents, with server-side sessions and a memory service that persists context across turns. The domain problem it addresses is real: agents need durable state, and building that infrastructure from scratch is hard. This article contrasts that managed-runtime model with an approach built on the Agent-Resident Execution Substrate, disclosed in U.S. Provisional Application No. 64/070,239, in which the agent's identity, cognitive state, and outcome lineage are carried by the agent itself and preserved under continuity proofs across substrate changes.

What Google Vertex AI Agent Engine (managed runtime for deploying and scaling agents, with sessions/memory) Does

Google Vertex AI Agent Engine is a managed runtime within Google Cloud's Vertex AI platform for deploying, hosting, and scaling agents built with common agent frameworks. It handles the operational burden of running an agent in production: containerized deployment, autoscaling, request handling, tracing, and integration with the broader Vertex AI model and tooling ecosystem. It provides a managed Sessions

capability that maintains conversational state across turns within a session, and a Memory Bank capability that extracts, stores, and retrieves longer-lived facts so an agent can recall relevant context in later interactions.

These are genuine strengths. For teams that want to ship an agent without operating their own serving infrastructure, Agent Engine removes a large amount of undifferentiated work. Sessions and Memory Bank give developers a coherent, framework-agnostic way to persist and recall context, which is one of the harder problems in applied agent engineering. The service is well integrated with managed model endpoints, observability, and enterprise controls that many organizations already rely on. For a large class of cloud-hosted agent applications, this is a sound and productive foundation, and nothing here should be read as suggesting otherwise.

This comparison is not a critique of that design. It is a description of a different architectural axis, and of what the disclosed invention structurally provides along that axis.

The Architectural Axis

The axis is where the durable "self" of the agent lives, and what continuity is asserted over.

In a managed-runtime model, the agent is a deployed workload and its durable state is a set of managed cloud resources beside it. Sessions and memory are services the runtime maintains on the agent's behalf: they hold conversation history and extracted facts, keyed to a session or user, in server-side storage under the provider's account and control plane. This is an effective way to give an agent recall. The continuity it offers is continuity of stored context: the same session or memory records can be read back on a later request.

The disclosed substrate addresses a different property. It treats the agent as a persistent, identity-bearing entity whose identity, cognitive state, and append-only lineage are carried by the agent and preserved continuously across the events that would otherwise reset it. The distinction is not "cloud versus local," though the substrate is designed to run on the user's own device. The distinction is what is being made continuous: stored context that a runtime hands back, versus a verifiable agent identity that persists as one entity even as the models, tools, and the runtime underneath it are replaced. These are complementary properties, not competing implementations of the same one.

How the Disclosed Approach Differs

The specification describes a semantic agent that operates as the persistent execution substrate of a computing device. The agent comprises four persistent fields: a persistent identity field, a cognitive state field, an append-only lineage field, and a governance policy field. Inference endpoints, retrieval models, and knowledge ingestion are not the agent; they are managed components subordinate to it, registered in a tool registry and subject to governed lifecycle operations.

The load-bearing mechanism on this axis is the continuity guarantee. The specification states that the agent's identity, cognitive state, and lineage are preserved across lifecycle operations applied to subordinate components, and that these fields are not modified, reset, or interrupted by installing, retraining, replacing, archiving, or removing any inference endpoint. The guarantee explicitly extends to the substrate runtime itself: a substrate-runtime update is executed through a staged procedure, and the agent's persistent identity is carried across runtime versions through a continuity attestation that cryptographically chains the identity over the sequence of runtime versions. The specification describes the practical consequence directly: every managed endpoint currently registered can be replaced, and the runtime itself updated, without modification to the agent, and any external party verifying the agent's continuity sees an unchanged identity across the sequence of subordinate replacements.

Continuity here is not asserted by a control plane returning stored records; it is verifiable from the lineage. The lineage field is an append-only sequence of records, each carrying a cryptographic reference to its predecessor, such that the agent's complete operational history is deterministically reconstructible and any prior record's alteration produces a detectable continuity break. The identity field can carry a continuity hash chained over that sequence, updated with each appended record and re-derivable from any prior reference value; a discrepancy is detectable and triggers governed escalation. Memory, in this model, is not only recalled context but an auditable, tamper-evident history bound to a single continuous identity.

Two further mechanisms distinguish the approach and both trace to the specification. First, the personal corpus model: the tool registry can hold a model whose weights are fine-tuned against the user's own authored artifacts, so that inference reflects the user's terminology, structural conventions, and prior outputs without retrieval at inference time. Artifacts the user authors enter the lineage; a corpus assembly step derives an admissible training set under policy; a parameter-efficient fine-tuning step produces an updated artifact; and a governed substitution promotes it, with the agent's identity preserved across the swap. The user's body of work is internalized in parameters rather than consulted as external context on each request. Second, the privacy invariant: lineage records, model parameters, training corpora, and counterparty records are, per the specification, not transmitted off the device except under an explicit disclosure policy object, with every off-device disclosure recorded as an auditable lineage event. Federation across a user's devices, where used, exchanges lineage records and governed updates rather than aggregated weights or synchronized files, and the specification distinguishes this from both federated-learning weight averaging and cloud-account device association by producing a federated agent identity verified through cross-device attestations.

Where They Fit Together

These are not mutually exclusive, and for many deployments they compose. A managed runtime like Agent Engine is built to deploy, scale, and operate agents as cloud workloads, with sessions and memory that make server-side recall straightforward and observability that makes production operation manageable. The disclosed substrate is built to make an agent a continuous, identity-bearing, auditable entity whose self survives replacement of its models and its runtime, and to keep the user's accumulated work and history under a local governance boundary.

A reasonable composition uses each for what it is for. An agent could run on a managed runtime for scaling and operational tooling while a substrate-resident personal corpus model and lineage hold the user's internalized body of work and continuity-proofed history on the user's own device, disclosed off-device only under explicit policy. The substrate's application interface admits host applications under policy and dispatches their inference requests, which is a natural seam for such an arrangement. The question is not which product wins; it is whether an application needs managed cloud operation, agent-carried verifiable identity, or both.

Boundary Conditions

An honest account of the disclosed approach names its limits. The substrate is designed for devices with bounded local memory, storage, and compute; the personal corpus model relies on parameter-efficient fine-tuning within a local training window, and its update cadence is bounded by device compute, power, and thermal headroom rather than by elastic cloud capacity. On-device model artifacts are sized to fit local envelopes, which constrains raw model scale relative to large managed endpoints. The continuity and privacy guarantees rest on the integrity of the substrate runtime and, where used, a hardware security element; their strength is only as good as those roots of trust, and the

specification accordingly describes attestation and quarantine mechanisms. Federation adds conflict-resolution and coordination considerations that a single-node deployment does not have.

The subject matter is disclosed in a provisional application and reflects an early-stage filing. The specification describes architecture and embodiments; it is not a benchmark report, and this article makes no performance claims for the disclosed system. Where the specification distinguishes the approach from retrieval-augmented generation, self-training feedback loops, federated learning, and cloud-account synchronization, it does so as architectural framing, not as a measured comparison against any specific product.

Disclosure Scope

The invention described here is disclosed in U.S. Provisional Application No. 64/070,239. The technical statements about the disclosed approach are grounded in that specification; the scope of any resulting protection is defined by claims as they may issue, not by this article. References to Google Vertex AI Agent Engine and to managed agent runtimes are external context provided to locate the disclosed work along a single architectural axis, and reflect that product's publicly described, architecture-level behavior. Nothing here asserts a defect, failure, or deficiency in Google Vertex AI Agent Engine or any other product; the contrast is one of design intent and structural properties, not of quality. Any product feature descriptions are the responsibility of their respective provider and may change over time.

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)

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)

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