Nick Clark · Published April 25, 2026 · PDF

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?

AWS Bedrock AgentCore provides managed runtime, memory, identity, and gateway services for deploying agents at scale on AWS infrastructure. It is a capable, production-grade way to run agent workloads without operating your own serving stack. The architectural axis this article addresses is different: an agent whose identity, cognitive state, and outcome lineage are hardware-bound and resident on the device, with inference endpoints held as governed subordinate assets. That property is supplied by the Agent-Resident Execution Substrate, disclosed in U.S. Provisional Application No. 64/070,239. This article describes AWS Bedrock AgentCore accurately and the disclosed approach as a dated public disclosure tied to that filing.

What AWS Bedrock AgentCore (runtime, memory, identity, and gateway services for deploying agents at scale) Does

AWS Bedrock AgentCore is Amazon's set of managed services for building, deploying, and operating AI agents at scale on AWS. Rather than a single monolithic product, it is a family of complementary capabilities. AgentCore Runtime provides a serverless, session-isolated execution environment for running agent code, framework-agnostic, so teams can bring agents written with a variety of orchestration libraries and models. AgentCore Memory offers managed short-term and long-term memory so an agent can carry context within a session and retain durable knowledge across sessions without teams building their own storage tier. AgentCore Identity handles agent authentication and secure, scoped access to tools and downstream resources, including delegated access on behalf of a user. AgentCore Gateway turns existing APIs and services into agent-callable tools, and a browser and code-interpreter toolset extends what agents can act on. Observability features surface traces and operational telemetry.

This is a mature, well-engineered offering, and it does what it targets well. It removes a large amount of undifferentiated operational work: scaling, session isolation, credential brokering, tool exposure, and memory persistence are handled by the platform so teams can focus on agent behavior. It integrates cleanly with the broader AWS ecosystem and with Amazon Bedrock's model catalog, and it lets organizations move agent prototypes toward production quickly. For teams whose agents live in the cloud and whose priority is scale, integration, and operational simplicity, it is a strong choice. The comparison in this article is not about its quality or execution; it concerns a single architectural axis.

The Architectural Axis

The axis is where the agent's durable identity, cognitive state, and history of outcomes physically reside, and what relationship the agent has to the inference models it uses.

In a managed cloud-services model, the agent is composed at deploy time from separately provisioned services: a runtime that executes agent code in isolated sessions, a memory service that persists context, an identity service that brokers credentials, and a gateway that exposes tools. Each service is a well-defined, independently managed component of the platform. Memory is durable context and knowledge that the agent reads and writes; identity is an authentication and authorization construct for the workload and its access to resources. The models the agent calls are shared, service-hosted endpoints invoked per request. This composition is exactly what makes the platform scalable and easy to operate.

The disclosed substrate treats the same concerns as properties of a single persistent entity rather than as a composition of separately provisioned services. In the spec's framing, a semantic agent operates as the persistent execution substrate of a device, and inference endpoints, ingestion modules, and related computational assets are maintained as governed managed components subordinate to that agent. The agent bears a persistent identity field, a cognitive state field, a lineage field, and a governance policy field, and these are preserved continuously across the lifetime of the device and across lifecycle operations applied to the subordinate components. This is a structural difference in what the agent is, not a claim that one deployment model is superior for all workloads. It is framed here as a difference along one axis, not as a defect in the managed-services approach.

How the Disclosed Approach Differs

Three spec-grounded mechanisms mark the difference on this axis.

First, identity is hardware-bound and continuity-preserving rather than a workload credential. The spec discloses a persistent identity field that may be cryptographically bound to a hardware security element of the substrate device, a secure enclave, trusted platform module, hardware security module, or embedded secure element, with private key material that is not extractable from the element. The binding is established at first

instantiation through a key-derivation operation incorporating hardware-attested values and is re-verifiable at any subsequent substrate event. A hardware-bound identity is not transferable to a device lacking access to the bound element except under a governed migration operation attested by both origin and destination devices, and the substrate can quarantine agent execution if the hardware element fails attestation. The agent's identity is not dependent on any specific model artifact and is preserved across replacement, retraining, or removal of any subordinate model.

Second, models are subordinate governed assets, not shared endpoints the agent merely calls. The spec discloses a managed inference tool registry in which each endpoint comprises a model artifact, an interface specification, and an associated governance scope, and a tool lifecycle controller that performs installation, retraining, replacement, archival, and removal under a governed state machine. Each lifecycle operation is evaluated against the governance policy field and recorded in the lineage field as a deterministic event, and the continuity guarantee holds the agent's identity, cognitive state, and lineage unchanged across arbitrary replacement of every registered endpoint, including update of the substrate runtime itself under a continuity attestation. The agent owns and governs the lifecycle of its tools; it does not simply consume them.

Third, memory in the disclosed approach is an append-only lineage field under a continuity proof, not durable context storage. Each lineage record encodes an operational event, dispatched inference requests and their outcomes, integrity-signal feedback, lifecycle operations, ingestion events, counterparty encounters, and is structured so the complete operational history is deterministically reconstructible and each record is verifiable against its predecessor through cryptographic chaining. No prior record can be modified or deleted without producing a detectable continuity break. That lineage is not only a record: it is a training signal. The spec discloses a personal corpus model whose parameters are fine-tuned against a corpus derived from the user's own artifacts and from downstream-outcome references (acceptance, revision, execution success or failure), so the user's accumulated body of work is internalized in the model's weights rather than retrieved as context at inference time.

The spec notes this outcome-derived signal is structurally distinct from training on a model's own prior outputs and is intended to mitigate distributional collapse across successive retraining events.

Where They Fit Together

These are largely complementary, and the honest framing is compose rather than compete. AWS Bedrock AgentCore is built for running agent workloads at scale in the cloud: elastic session isolation, managed memory, credential brokering, and turning existing APIs into agent-callable tools. The disclosed substrate is built for a persistent, device-resident agent whose identity and history stay on the device under governance, with inference tools it owns and retrains locally.

A natural composition places the substrate on the device and treats a managed cloud runtime as one destination among the substrate's governed capabilities. The spec discloses a cloud-burst forwarding subsystem that selectively forwards inference requests to a remote inference endpoint when local endpoints lack capability or capacity, governed by a cloud-burst policy specifying admissible remote endpoints, disclosure scopes, encryption requirements, and cost limits, with each forwarding event recorded as a disclosure event under the privacy invariant. In that arrangement, a cloud runtime and gateway of the kind AgentCore provides can serve the heavy, elastic, or specialized inference that a bounded local device cannot, while the device-resident agent retains its identity, lineage, and governance authority. AgentCore does the scale-out work it is good at; the substrate keeps the durable identity and the audit-grade history local. Each is for a different job.

Boundary Conditions

The disclosed subject matter is early-stage. U.S. Provisional Application No. 64/070,239 is a provisional filing; it establishes a priority date for the disclosed subject matter and is not an issued patent, and its ultimate claim scope is not yet determined.

The architecture is disclosed across embodiments; the descriptions here are illustrative rather than a finished commercial system, and the spec itself states that its embodiments are illustrative and not exhaustive.

There are real engineering constraints on the device-resident side. The substrate is designed for a bounded local memory, storage, and compute envelope, so local model artifacts are of bounded size and are often quantized, and local fine-tuning is parameter-efficient (for example low-rank adaptation) and confined to a policy-declared training window. Workloads that need the largest frontier models, very high throughput, or elastic burst capacity are precisely where a managed cloud service is well suited, and the substrate's own cloud-burst mechanism acknowledges this by forwarding such requests off-device under policy. Hardware-bound identity depends on a functioning hardware security element and imposes a governed migration process across device refresh. None of these is a criticism of AWS Bedrock AgentCore; they are the ordinary trade-offs of running an agent on a device rather than on elastic cloud infrastructure, and they define where each approach fits.

Disclosure Scope

The invention and all statements about what it does, its mechanisms, its hardware-bound identity, its managed inference tool registry and governed lifecycle operations, its append-only lineage field and continuity proofs, its personal corpus model training, its privacy invariant, and its cloud-burst forwarding, are disclosed in U.S. Provisional Application No. 64/070,239. This article is a public disclosure of that subject matter as of the filing date, intended to be enabling to a person of ordinary skill in the art and reasonably broad across the disclosed embodiments.

References to AWS Bedrock AgentCore and its components (AgentCore Runtime, Memory, Identity, Gateway, and related tooling), to Amazon Bedrock, and to Amazon Web Services are provided solely as external market and architectural context to situate the disclosed approach. Those products and services are the property of their respective

owners, are described here in a general, architecture-level manner, and are not claimed as part of U.S. Provisional Application No. 64/070,239. Any comparison is limited to the single architectural axis of where the agent's durable identity, cognitive state, and outcome lineage reside and the agent's ownership relationship to its inference tools; it is not an assertion of any defect, limitation, or shortcoming in AWS Bedrock AgentCore, nor a statement about its overall quality, roadmap, security, or commercial standing. Nothing in this article should be read as legal, freedom-to-operate, or investment advice.

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)