Kasadara

AWS Bedrock Agentcore

Architectural Paradigms and Implementation Strategies in AWS Bedrock Agent Core: An Enterprise-Grade Analysis

Kasadara

Executive Summary

This documentation presents a comprehensive implementation of advanced agent capabilities using Amazon Bedrock AgentCore to design, build, and operate secure, scalable, and production-ready AI agents.

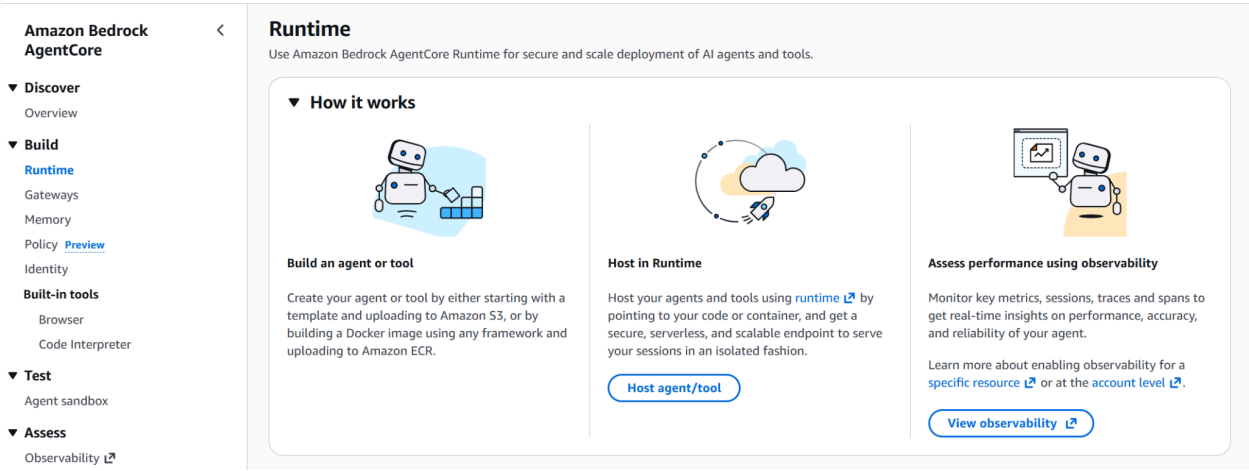
- Agent Core simplifies agent development compared to direct model API usage

1. What is Bedrock Agent Core

Amazon Bedrock AgentCore is an advanced agent orchestration framework that enables AI agents to securely interact with tools and data using fine-grained permissions and built-in governance

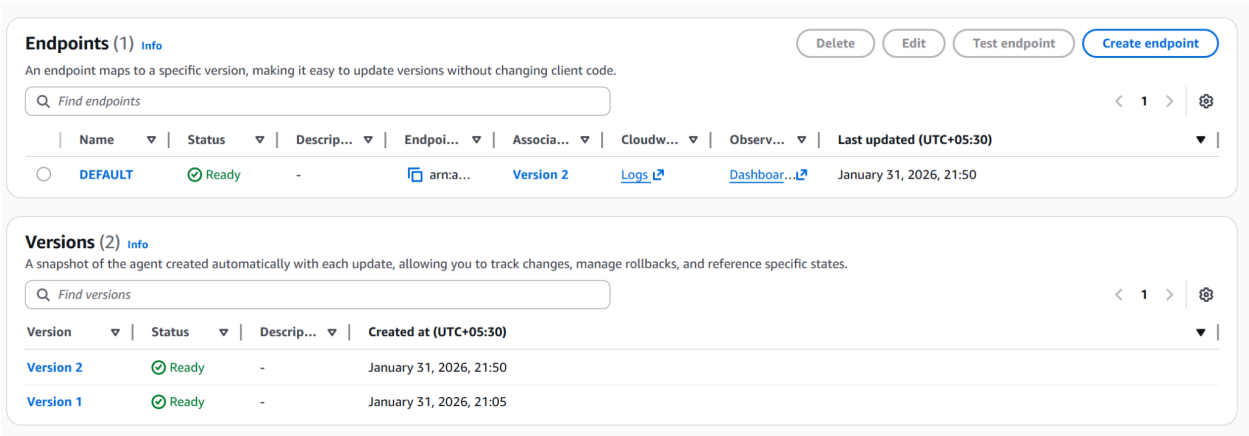
1.1 AgentCore Runtime

Serverless environment that provides complete session isolation and automatically handles Docker packaging, ECR registry management, and scaling.



- Secure state-**decoupled runtime** with IAM encryption networking persistent resumable workflows

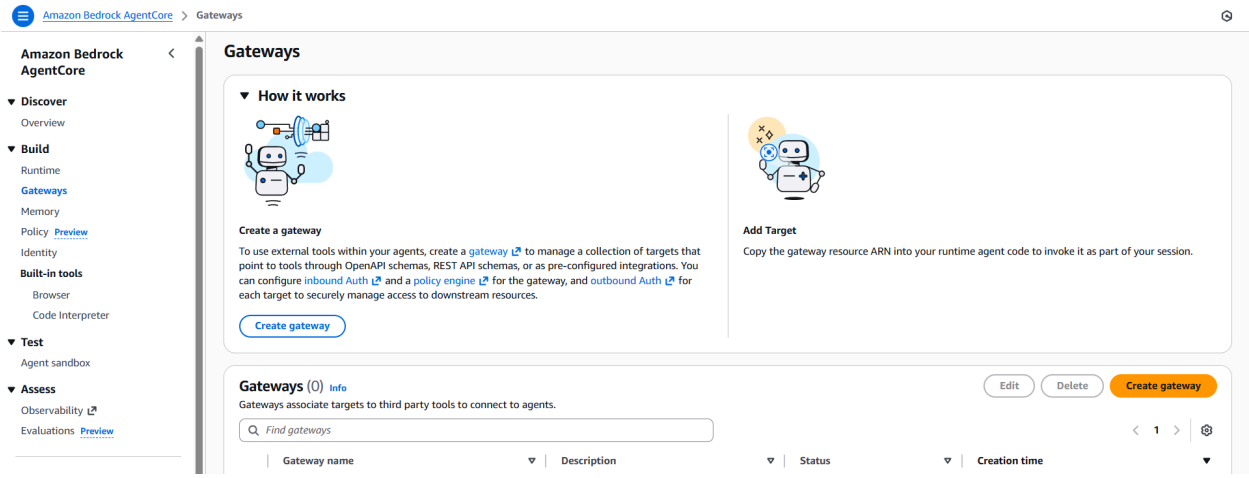
- Built-in observability with traces metrics logs auditing for scalable agents



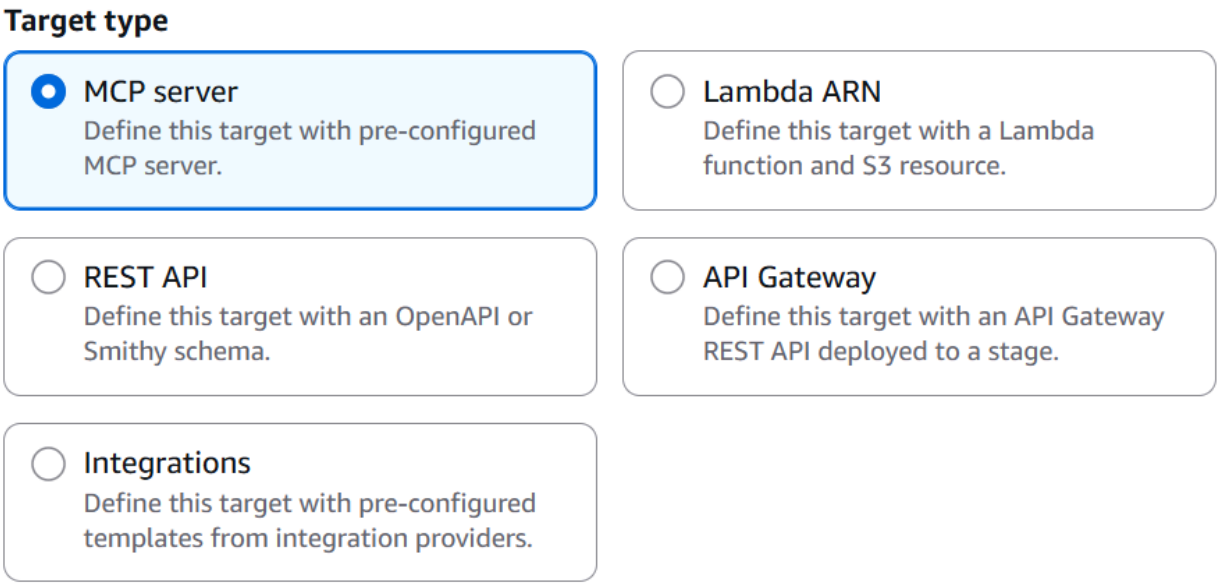
- Endpoints act as fixed access URLs that map to a specific agent version, allowing seamless upgrades without modifying client applications.
- Each version captures agent state, enabling rollbacks, reproducibility, and precise change tracking.

1.2 AgentCore Gateway

Centralized control layer for agent tool integration, standardizing access to AWS Lambda functions, REST APIs, and OpenAPI endpoints by converting them into agent-ready tools using the **Model Context Protocol (MCP)**

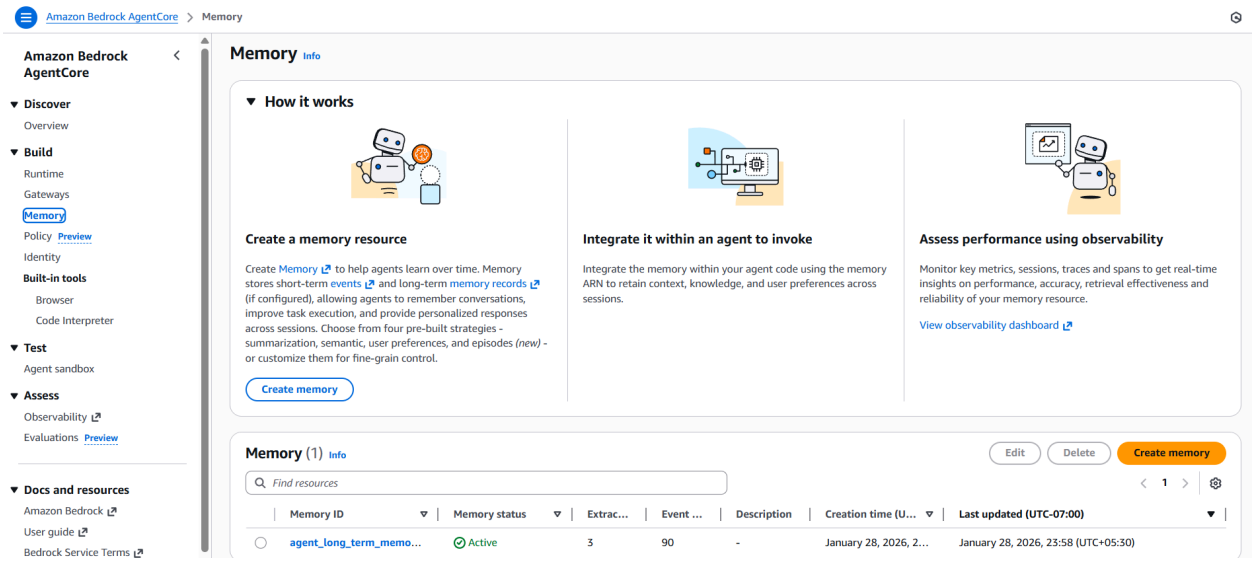


To use external tools within your agents

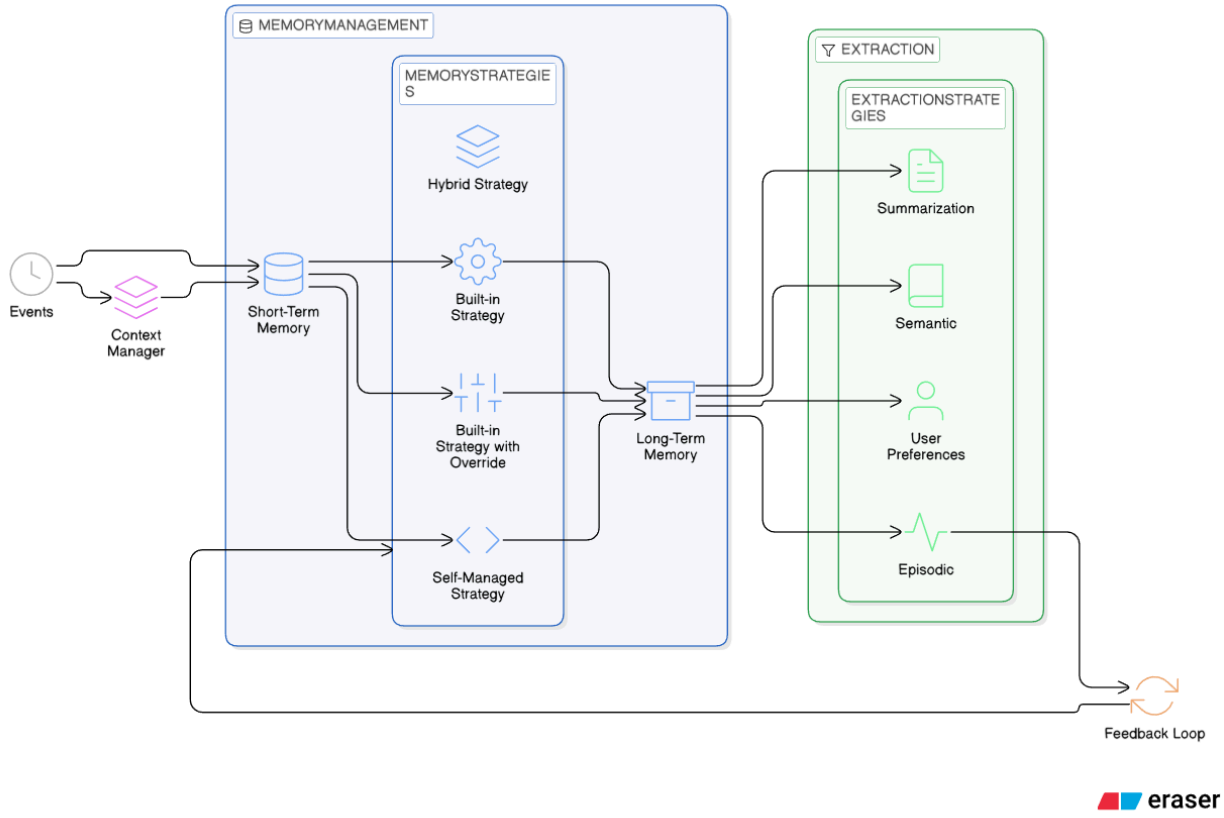


1.3 AgentCore Memory

AgentCore provides a native, fully managed memory system that is deeply embedded into the agent execution lifecycle,



Built-in AgentCore memory strategies



Semantic Memory Strategy

- Stores important facts automatically for reuse across future agent interactions.

Summary Memory Strategy

- Shortens long conversations into compact context for efficient reasoning.

User Preference Strategy

- Remembers user choices to personalize agent behavior without repeated instructions.

1.3 AgentCore Policy

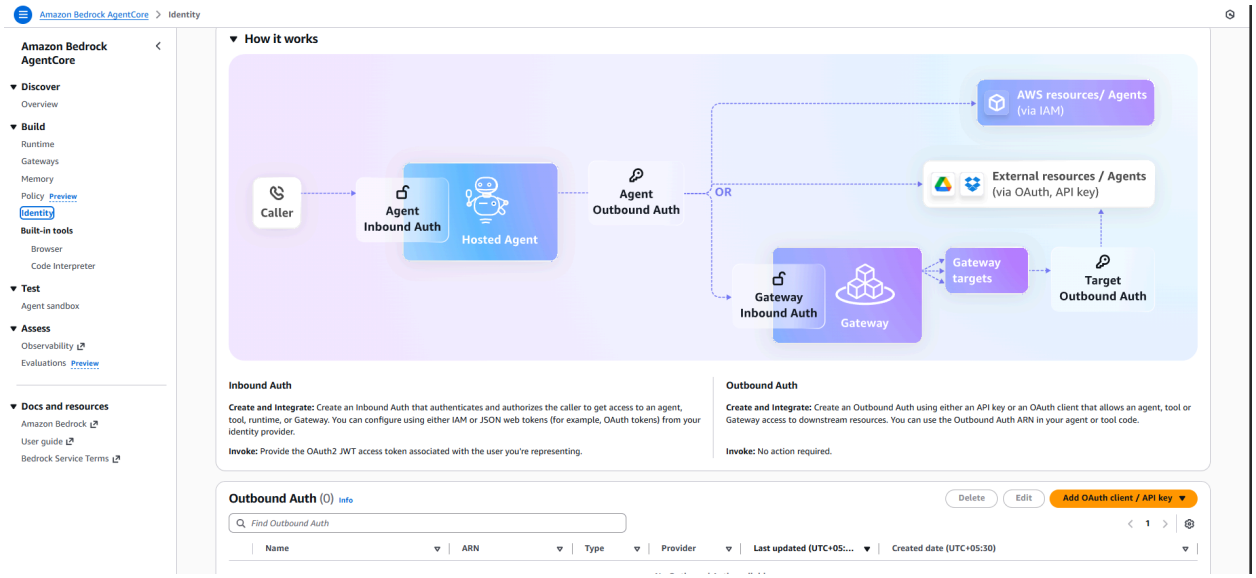
AgentCore policies are mandatory and enforced at runtime, meaning an agent cannot bypass or override them—even if the model attempts unsafe or unauthorized actions.

- Uses **Cedar language** to define natural language boundaries for secure access.

```
permit (
    principal is AgentCore::OAuthUser,
    action == AgentCore::Action::"InsuranceAPI__file_claim",
    resource ==

AgentCore::Gateway::"arn:aws:bedrock-agentcore:us-west-2:012345678901:gateway/insurance"
)
when
{
    context.input has claimType &&
    (context.input.claimType == "health" ||
     context.input.claimType == "property" ||
     context.input.claimType == "auto")
};
```

1.4 Agent Identity:



AgentCore Identity defines who an agent is and what it is allowed to do.

Each agent is given its own secure identity, just like a user or service account. This identity controls which tools, APIs, and AWS resources the agent can access.

AgentCore uses IAM-based permissions, the agent can only perform actions that are explicitly allowed. Even if the model tries to do something unsafe, the action is blocked by policy.

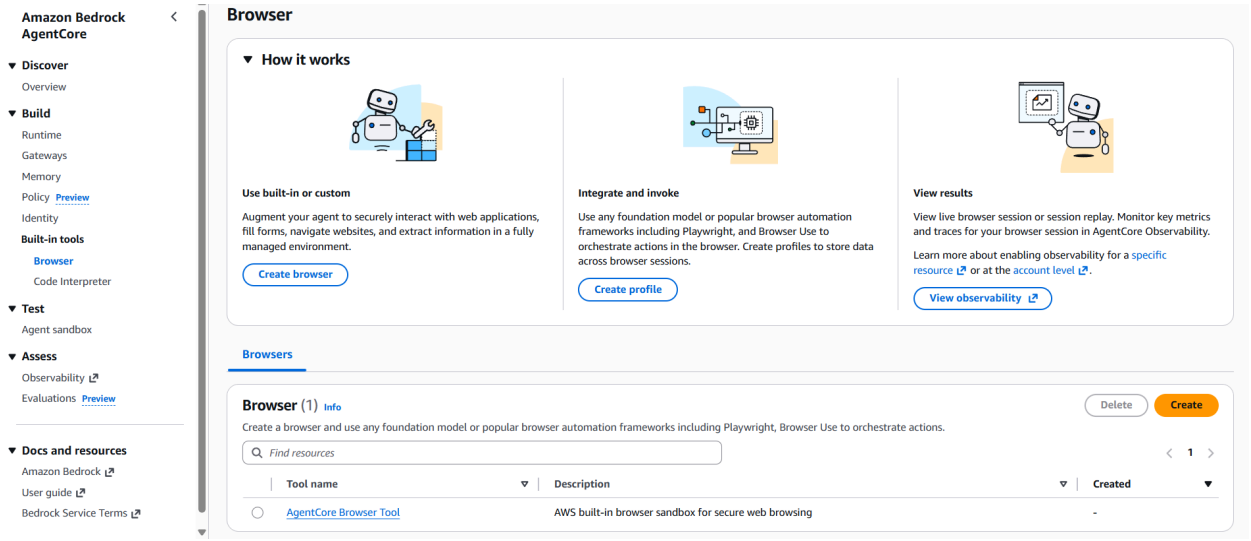
Every action taken by the agent is logged and traceable, making it easy to audit, monitor, and meet security or compliance requirements.

1.4 Agent Built-in tools

Browser Tool

Ready-to-use, managed capabilities that can be integrated into any agent without additional infrastructure.

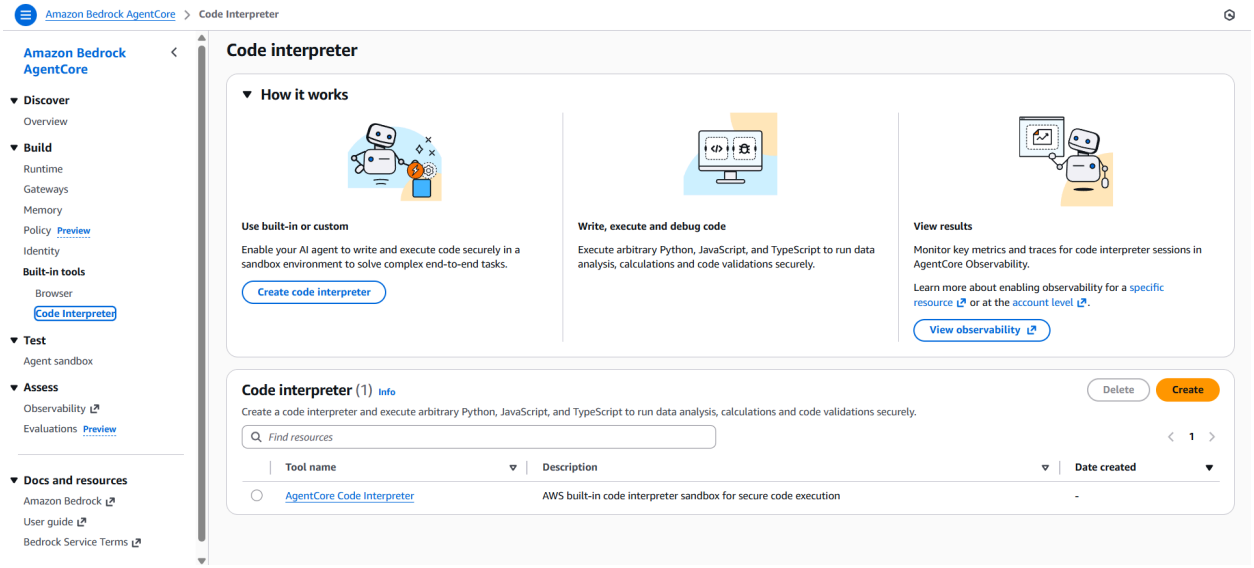
- Safely browses web information with controlled, auditable, and restricted agent access.



- Agents securely browse web sources, validate information, gather external data, supporting accurate, auditable decision making.
- Agents analyze data, perform calculations, generate insights, combining browsing and computation for autonomous reasoning workflows.

Code Interpreter Tool

- Enables agents to safely run code inside a sandboxed environment
- Perform calculations, analyze data, transform inputs, and validate logic in real time.



Features of AgentCore Agent

Human-in-the-Loop Native Support

- Tool execution can be paused for explicit human approval, then resumed without losing agent state.

Built-In Observability & Auditing

- Tool calls are automatically logged with execution context, outcomes, and policy decisions for compliance and debugging.

Kasadara

What's the difference between using Agent Core versus directly calling the Bedrock API?

AgentCore manages orchestration, retries, state, and tools automatically within a unified framework.Direct APIs require manual coordination, custom error handling, and explicit state management. As a result, AgentCore reduces development effort, complexity, and long-term maintenance.

Orchestration

- **Agent Core**: Automatically manages multi-step workflows, tool chaining, and decision flow
- **Direct API**: You manually coordinate every step and tool call

Retry & Reliability

- **Agent Core**: Built-in retries, exponential backoff, and timeout handling
- **Direct API**: Custom retry loops and error handling must be written and maintained

State Management

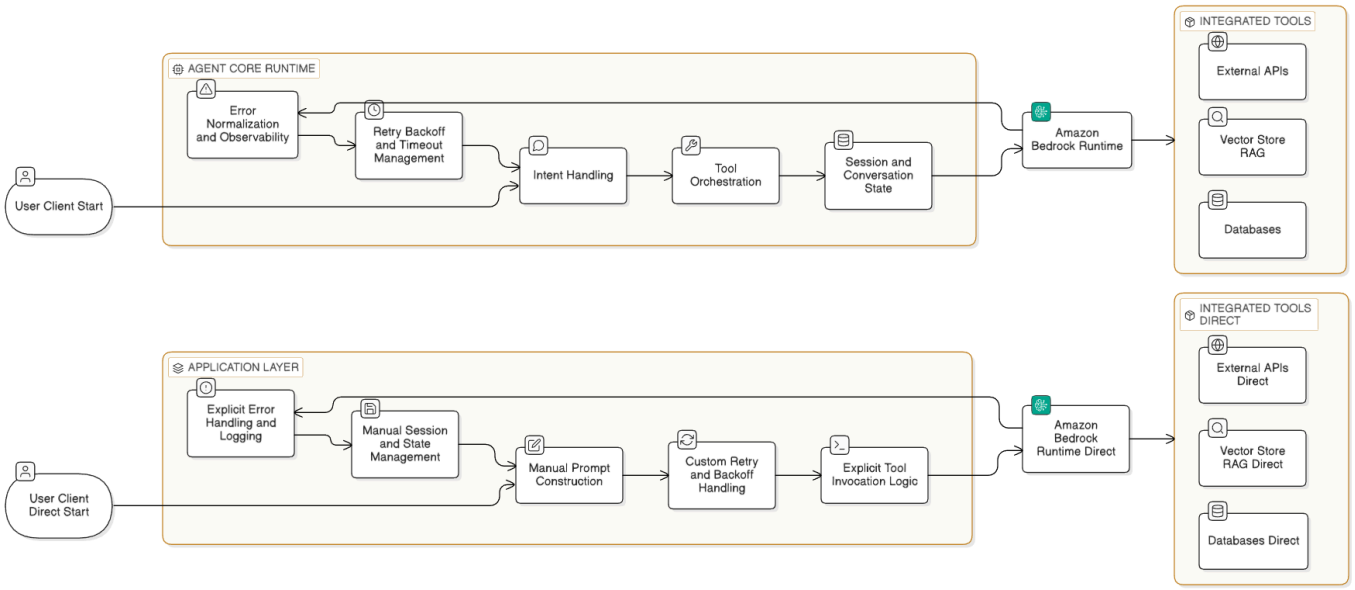
- **Agent Core**: Session and conversation context handled automatically
- **Direct API**: Manual tracking of conversation history and memory

Tool Integration

- **Agent Core**: Declarative tool registration and automatic invocation
- **Direct API**: Manual tool schema definition, invocation logic, and response parsing

Development & Maintenance Effort

- **Agent Core**: Faster development with significantly less code and lower maintenance
- **Direct API**: More boilerplate, higher complexity, and increased maintenance cost

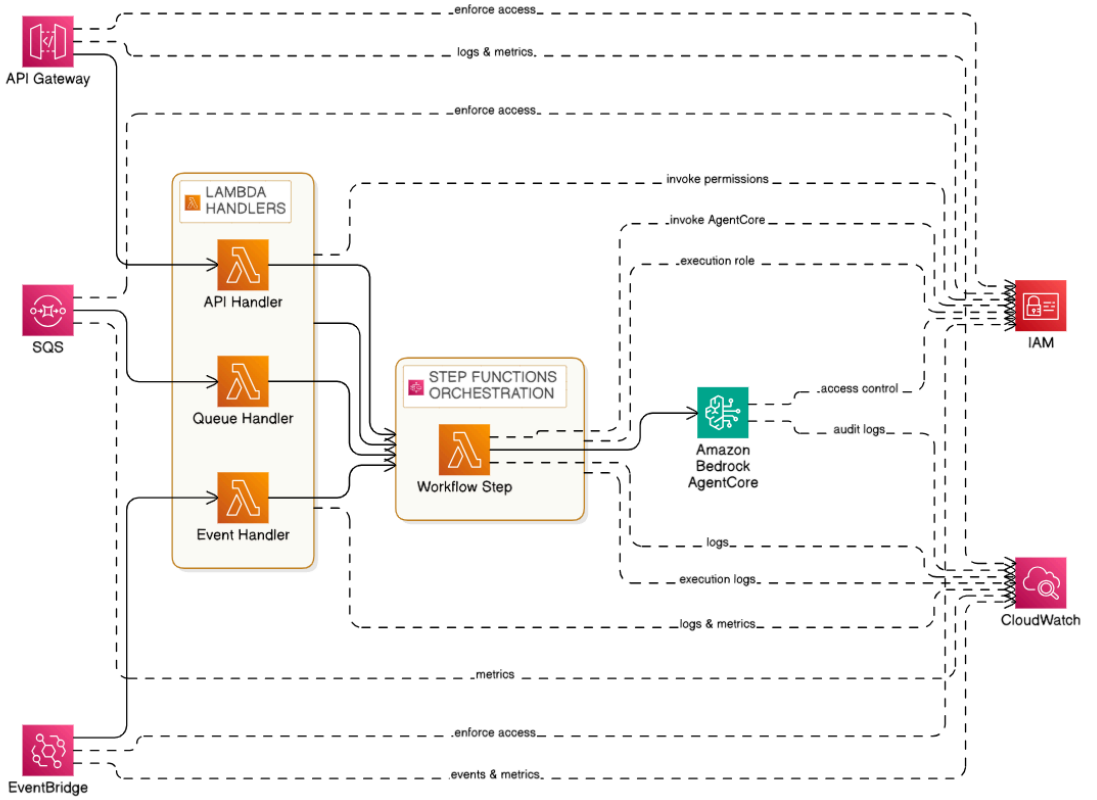


How can Amazon Bedrock AgentCore be integrated with AWS services such as Lambda, Step Functions, API Gateway, EventBridge, or SQS to build serverless, event-driven architectures?

AgentCore agents are versioned, exposed via **secure endpoints**, and invoked by **Lambda with IAM permissions**, enabling governed, scalable, event-driven orchestration.

- AgentCore exposes versioned, secure agent endpoints invoked by Lambda, enabling scalable event-driven execution through API Gateway, EventBridge, and SQS.

- Step Functions orchestrate multi-step agent workflows with retries, approvals, and observability, while IAM enforces strict access control and governance.


enforce access
logs & metrics
API Gateway
enforce access
LAMBDA HANDLERS
invoke permissions
invoke AgentCore
execution role
API Handler
SQS
IAM
Queue Handler
STEP FUNCTIONS ORCHESTRATION
access control
audit logs
Amazon Bedrock AgentCore
Workflow Step
Event Handler
logs
execution logs
CloudWatch
logs & metrics
metrics
enforce access
EventBridge
events & metrics

eraser