Lab 2: Enhance your Agent with Memory

Overview

Picture this: A valued customer contacts your support team about an issue with their recent order. They explain their preferences, share their frustration, and work with your agent to resolve the problem. Three weeks later, they contact support again with a related question. But now they have to repeat everything - their preferences, their history, their context - because your agent has no memory of previous interactions.

This is the reality for most AI agents today. Every conversation starts from zero, creating:

- **Frustrated customers** who must repeat their information repeatedly
- **Inefficient support** that cannot build on previous interactions
- **Lost opportunities** to provide personalized, proactive service
- **Poor customer satisfaction** due to impersonal, generic responses

Amazon Bedrock AgentCore Memory addresses this limitation by providing a managed service that enables AI agents to maintain context over time, remember important facts, and deliver consistent, personalized experiences.

AgentCore Memory operates on two levels:

- **Short-Term Memory**: Immediate conversation context and session-based information that provides continuity within a single interaction or closely related sessions.
- **Long-Term Memory**: Persistent information extracted and stored across multiple conversations, including facts, preferences, and summaries that enable personalized experiences over time.

What we are building

Transform Your Prototype into a Customer-Obsessed Agent

In this lab, you'll upgrade your Lab 1 prototype to deliver **exceptional customer experiences** through intelligent memory

aware assistant that:

Customer Experience Transformations:

- **"Welcome back, Sarah!"** - Instantly recognizes returning customers
- **"I remember you prefer email updates"** - Recalls individual preferences automatically
- **"Following up on your laptop issue from last month"** - Connects related conversations seamlessly
- **"Based on your purchase history, here's what I recommend"** - Provides personalized suggestions

Technical Implementation:

By the end of this lab, your agent will automatically:

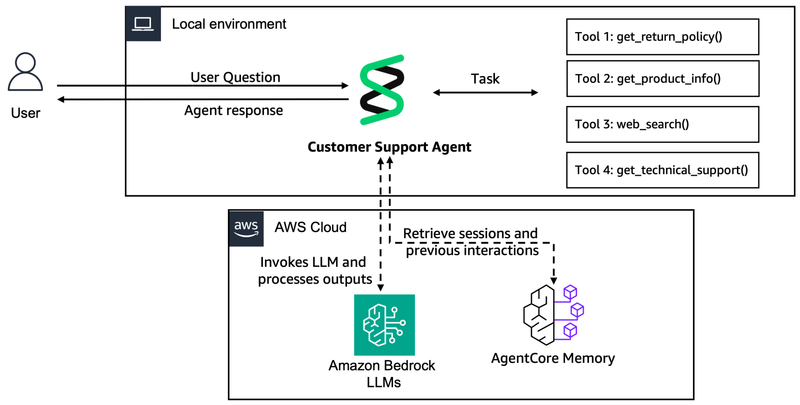
- **Remember every customer interaction** using AgentCore Memory hooks
- **Extract customer preferences** and store them persistently
- **Retrieve relevant context** when customers return
- **Personalize responses** based on historical patterns

The result: Your customers will experience truly personalized, context-aware support that builds relationships over time.

Memory Integration: This lab enhances your Lab 1 agent with persistent memory capabilities. Your existing tools remain unchanged while gaining contextual awareness.

Architecture diagram

The architecture adds AgentCore Memory components to your existing agent:


Local environment
User
User Question
Agent response
Customer Support Agent
Task
Tool 1: get_return_policy()
Tool 2: get_product_info()
Tool 3: web_search()
Tool 4: get_technical_support()
AWS Cloud
Invokes LLM and processes outputs
Retrieve sessions and previous interactions
Amazon Bedrock LLMs
AgentCore Memory

Steps to follow

⚠ **Note:** The code snippet in the following steps are shown for reference purposes only. The complete implementation is provided in the accompanying Jupyter notebook for this lab section.

After loading the necessary libraries, our first step is to create AgentCore memory resources:

Step 1: Create AgentCore Memory Resources

Our first step (**This is a one-time setup**) is to create a managed agentcore memory resource with multiple strategies (USER_PREFERENCE and SEMANTIC) to store comprehensive customer context, enabling persistent recall across conversations and balanced behavioral/factual insights.

Memory Strategies Configuration

Amazon Bedrock AgentCore Memory provides multiple long-term memory strategies. We create a memory resource combining:

Strategy Type	Purpose	Namespace Pattern
USER_PREFERENCE	Customer preferences and behaviors	`support/customer/{actorId`

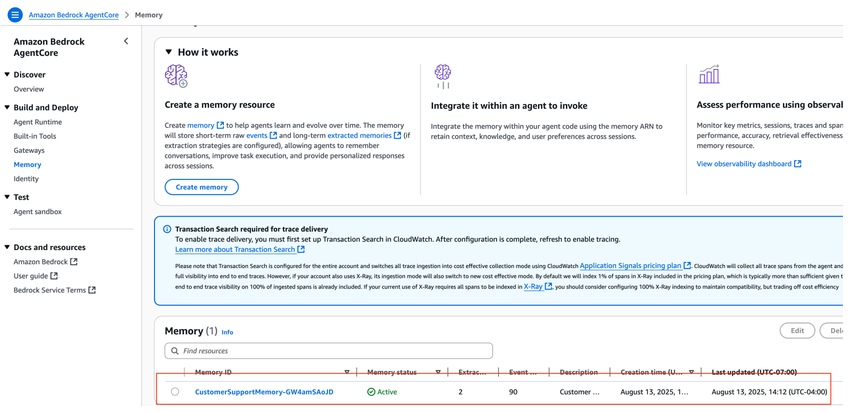
Strategy Type	Purpose	Namespace Pattern
`SEMANTIC`	Factual information and context	`support/customer/{actorId`

AgentCore Memory uses namespaces to logically group long-term memory messages. Every time a new long-term memory is extracted using this memory strategy, it is saved under the namespace you set. We use the follwing namespaces using the `actorId` to group messaging of the same customer together:

- `support/customer/{actorId}/preferences`: for the user preference memory strategy
- `support/customer/{actorId}/semantic`: for the semantic memory strategy

▶ **Code Snippet**

This picture illustrates the successful creation of a memory resource in AgentCore within AWS Console:



Create a memory resource

Create memory to help agents learn and evolve over time. The memory will store short-term raw events and long-term extracted memories (if extraction strategies are configured), allowing agents to remember conversations, improve task execution, and provide personalized responses across sessions.

Integrate it within an agent to invoke

Integrate the memory within your agent code using the memory ARN to retain context, knowledge, and user preferences across sessions.

Assess performance using observal

Monitor key metrics, sessions, traces and span performance, accuracy, retrieval effectiveness memory resource.

Transaction Search required for trace delivery
To enable trace delivery, you must first set up Transaction Search in CloudWatch. After configuration is complete, refresh to enable tracing.
Learn more about Transaction Search

Please note that Transaction Search is configured for the entire account and switches all trace ingestion into cost effective collection mode using CloudWatch Application Signals pricing plan. CloudWatch will collect all trace spans from the agent and full visibility into end to end traces. However, if your account also uses X-Ray, its ingestion mode will also switch to new cost effective mode. By default we will index 1% of spans in X-Ray included in the pricing plan, which is typically more than sufficient given t end to end trace visibility on 100% of ingested spans is already included. If your current use of X-Ray requires all spans to be indexed in X-Ray, you should consider configuring 100% X-Ray indexing to maintain compatibility, but trading off cost efficiency

Memory (1) Info

Find resources

Memory ID	Memory status	Extrac...	Event ...	Description	Creation time (U...	Last updated (UTC-07:00)
CustomerSupportMemory-GW4amSAoJD	✓ Active	2	90	Customer ...	August 13, 2025, 1...	August 13, 2025, 14:12 (UTC-04:00)

Note: The creation of AgentCore Memory resources can take couple of minutes. Please proceed to next steps after you confirm that memory got created successfully.

Step 2: Configure Memory Strategies for Customer Personalization

This is a one-time setup that defines how your agent will automatically learn about and remember your customers. Once configured, AgentCore Memory will seamlessly extract and organize customer insights from every interaction.

The Power of Automatic Personalization

Your memory resource uses two intelligent strategies that work together to create personalized customer experiences:

Strategy	Customer Benefit	What It Does
USER_PREFERENCE	**"I remember you prefer..."**	Automatically learns customer preferences, communication styles, and behavioral patterns
SEMANTIC	**"Regarding your previous issue..."**	Captures factual information, order details, and conversation context

This configuration enables your agent to automatically personalize every interaction by understanding what each customer wants, how they prefer to communicate, and what their history looks like - all without any manual effort from your team.

Step 3: Seed Customer History to Demonstrate Memory Intelligence

What we're doing from a customer perspective: We're simulating a returning customer named "customer_001" who has had previous interactions with your support team. This demonstrates how AgentCore Memory automatically transforms individual conversations into rich, persistent customer insights.

These insights are then stored as long-term memories that persist across weeks, months, and years of customer interactions.

What you'll see: We'll load previous customer interactions and watch AgentCore Memory automatically turn them into long-term customer insights - no manual processing required!

See It in Action: This step proves that AgentCore Memory's short-term to long-term memory transformation really works! The notebook shows you exactly how conversations get automatically turned into permanent customer memories. You'll see the memories before and after the transformation, so you know it's actually happening.

▶ **Code Snippet**

Step 4: Enable Automatic Customer Context with Memory Hooks

Customer Experience Goal: Now that we've demonstrated how AgentCore Memory transforms conversations into customer insights, we want to leverage this power in our live agent. Each time a customer interacts with our agent, we'll automatically:

1. **Personalize the conversation** based on their previous interactions and preferences
2. **Add new interactions to memory** to continuously improve future personalization

What our hooks integration does:

- **Before responding to customers**: Automatically retrieve relevant customer context and preferences to personalize the conversation
- **After responding to customers**: Automatically save the new interaction to AgentCore Memory to enrich the customer's profile

This creates a seamless experience where customer relationships build naturally over time without any manual effort.

Hook System Architecture

Strands Agents provides a powerful hook system that enables components to react to or modify agent behavior through strongly-typed event callbacks.

The hook system ensures memory operations happen automatically without manual intervention, creating a seamless experience where customer context is preserved across conversations.

To create the hooks we will extend the `HookProvider` class:

▶ **Code Snippet**

For detailed information about hooks, please refer to the Strands documentation.

Framework Flexibility: While this lab demonstrates memory integration using Strands Agents hooks, other frameworks like CrewAI, LangGraph, and LlamaIndex provide similar hook mechanisms for integrating AgentCore Memory. The customer benefits and memory strategies remain the same regardless of your chosen framework.

Step 5: Update Your Support Agent to be Personalized

We'll upgrade our Lab 1 customer support agent by integrating memory hooks through the `CustomerSupportMemoryHooks` class, transforming it into a context-aware assistant that combines tools, foundation model, and memory capabilities.

▶ **Code Snippet**

Step 6: Experience the Power of Personalized Customer Support

See the transformation in action! Execute the provided Python notebook to witness how your agent now delivers exceptional customer experiences through the power of AgentCore Memory and intelligent personalization.

What you'll experience:

- **"Welcome back, Sarah! I see you prefer email notifications."** - Instant customer recognition
- **"Regarding your previous laptop order..."** - Seamless context continuation
- **"Based on your history, I recommend..."** - Personalized suggestions
- **Automatic memory updates** - Every interaction improves future personalization

This isn't just about technical capabilities - it's about delivering the kind of customer experience that builds loyalty and drives business value.

Congratulations - You've Built a Customer-Obsessed AI Agent!

What you've accomplished: You've transformed your basic prototype into an intelligent, relationship-building assistant that delivers truly personalized customer experiences. Your agent now:

Customer Experience Improvements:

- **Remembers every customer interaction** - No more repetitive conversations
- **Learns customer preferences automatically** - Personalizes responses without manual effort
- **Maintains context across time** - Builds relationships over weeks and months
- **Delivers exceptional service** - Creates the kind of experience customers remember

Business Impact:

Your agent is now positioned to drive real business value through improved customer satisfaction, reduced support costs, and increased customer loyalty.

What's Next: In Lab 3, we'll add enterprise tools and APIs to your agent, enabling it to access real customer data, update orders, and integrate with your existing business systems - taking personalization to the next level.

Try it out

At an AWS event

If you are following the workshop via workshop studio, now go to JupyterLab in SageMaker Studio. In the JupyterLab UI navigate to `lab-02-agentcore-memory.ipynb`

Self paced

For the complete working implementation and examples: Add Memory to Your Agent.

Resources

- Amazon Bedrock AgentCore Memory Documentation
- Strands Agents Hooks Documentation
- AgentCore Memory Strategies Guide

Ready to Continue? Lab 3: AgentCore Gateway → Enhance your customer support agent with external tool integration and expand its capabilities beyond the built-in tools.

Previous Next