Lab 3: Scale with Gateway and Identity

Introduction

In Lab 2, we personalized our agent, creating a more streamlined customer experience using AgentCore Memory. Now that memory is in place, we turn our attention to the next key element of production-ready agents: access to powerful tools and scaling their impact by exposing our web search tool as an MCP tool using AgentCore Gateway. In this lab we will also use existing tools for checking warranty and getting the customer profile. This will emulate the real world scenario where an agentic use case reuses tools created by other teams and/or applications. In this lab we will also see how AgentCore Gateway integrates with AgentCore Identity to provide secure connections via inbound and outbound authentication.

Why is this Important

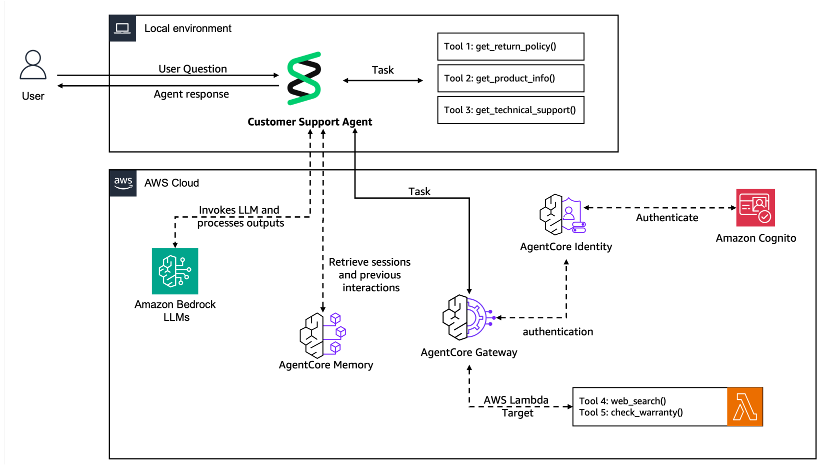
Great agents need tools that take full advantage of proprietary and third-party APIs and data, which lets those agents get things done for both internal and external customers. However, building, securing, and scaling agent tools is difficult, becoming a significant blocker for customers moving from agent prototypes to real agent business value in production.

In this lab, we show you how AgentCore Gateway helps tackle this problem of giving your agents easy access to enterprise APIs and data. We'll take the agent from Lab 2 and demonstrate how to move a local tool so that now many agents can use it, and how to easily expose an existing API as MCP with secure acess to plug into any agent framework.

Why Use AgentCore Gateway?

AgentCore Gateway serves as a connectivity layer that enables AI agents to discover, authenticate, and invoke real-world tools through a unified Model Context Protocol (MCP) endpoint. This is tools.

Architecture diagram



- Provides a fully-managed MCP server solution without infrastructure management
- Enables integration of existing APIs and Lambda functions
- Offers uniform interface across diverse tools
- Ensures secure authentication and authorization
- Supports semantic tool discovery and selection

What you'll build :

Tool Centralization & Reusability:

- Migrate web search from local tool to centralized AgentCore Gateway
- Integrate existing enterprise Lambda functions (warranty check, customer profile)
- Create a shared tool infrastructure that multiple agent types can access

Enterprise-Grade Security:

- Implemente JWT-based authentication with Cognito integration
- Configure secure inbound authorization for gateway access
- Establishe identity-based access control for tool usage

Scalable Architecture Foundation:

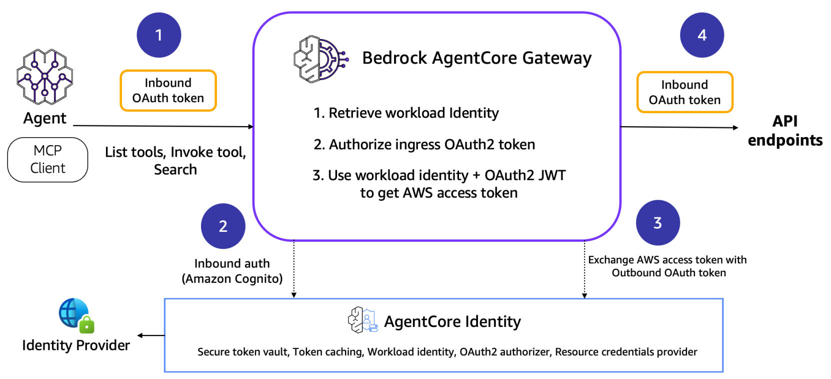
- Build reusable tools that serve multiple use cases (customer support, sales, returns processing)
- Eliminate code duplication across different agents
- Create centralized management for tool updates and maintenance

Authentication Flow

Gateway implements dual authentication:

- Inbound Auth: User validation
- Outbound Auth: Secure backend connections

Authorization in Bedrock AgentCore Gateway


1
Inbound OAuth token
Agent
MCP Client
List tools, Invoke tool, Search
Bedrock AgentCore Gateway
1. Retrieve workload Identity
2. Authorize ingress OAuth2 token
3. Use workload identity + OAuth2 JWT to get AWS access token
4
Inbound OAuth token
API endpoints
2
Inbound auth (Amazon Cognito)
3
Exchange AWS access token with Outbound OAuth token
Identity Provider
AgentCore Identity
Secure token vault, Token caching, Workload identity, OAuth2 authorizer, Resource credentials provider

Steps to follow

Step 1 : Expose Lambda functions as MCP endpoints

AgentCore Gateway populates the Lambda context with the name of the tool to invoke, while the parameters passed to the tool are provided in the Lambda event.

Step 2: Convert your web search tool to MCP

We can MCP-ify any tools which we think we'll use for multiple Agents. One of these tools might be a web search tool like we built in Lab1

Step 3 : Create your AgentCore Gateway

Now let us create the AgentCore Gateway to expose the Lambda function as MCP-compatible endpoint. To validate the callers authorized to invoke our tools we need to configure the Inbound Auth.

Inbound Auth works using OAuth authorization, the standard for MCP servers. Your client would receive an access token which is used at runtime.

▶ **Code Snippet**

Step 4 : Add the Lambda function Target and our new MCP-based tools

Using `prerequisite/lambda/api_spec.json` we will define the tools that your gateway will host. Also integrate the authentication token from Cognito into an MCPClient from Strands SDK to create an MCP Server

▶ **Code Snippet**

Step 5 : Test the agent with MCP tool access to existing APIs

Test your agent with the following example prompts :

- "I have a Gaming Console Pro device , I want to check my warranty status, warranty serial number is MNO33333333."
- "What are the warranty support guidelines?"

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-03-agentcore-gateway.ipynb`

Self-paced

For the complete implementation and examples: Create your AgentCore Gateway

Congratulations on completing Lab 3 !!

Ready to Continue? Lab 4: Deploy the Agent to Production with observability → Deploy to Production using AgentCore Runtime with observability

Previous Next