



Diving Deep into Bedrock AgentCore

Diving Deep into Bedrock

• • • • •

Amazon Bedrock AgentCore Fundamentals

► AgentCore Runtime

► AgentCore Gateway

▼ AgentCore Identity

Inbound Auth

▼ Outbound Auth

API Key Credential Provider

User-Delegated Access with Cognito and 3-Legged OAuth flow with GitHub

User-Delegated Access with Cognito and 3-Legged OAuth flow with Google

▼ AgentCore Memory

AgentCore Memory: Short-Term Memory

AgentCore Memory: Long-Term Memory Strategies

▼ AgentCore Tools

AgentCore 1P Tool - AgentCore
Code Interpreter

▼ AgentCore 1P Tool - AgentCore Browser

AgentCore Browser - Nova
Act SDK

AgentCore Browser - Browser-Use

▼ AgentCore Observability

AgentCore Observability for Runtime hosted Agent

Observability for Non-Runtime hosted Agents

AgentCore

AgentCore Documentation

- ▶ **AWS account access**

Workshop catalog in AWS Builder Center [↗](#)

► **Content preferences**

Exit event

Event ends in 19 hours 5 minutes.

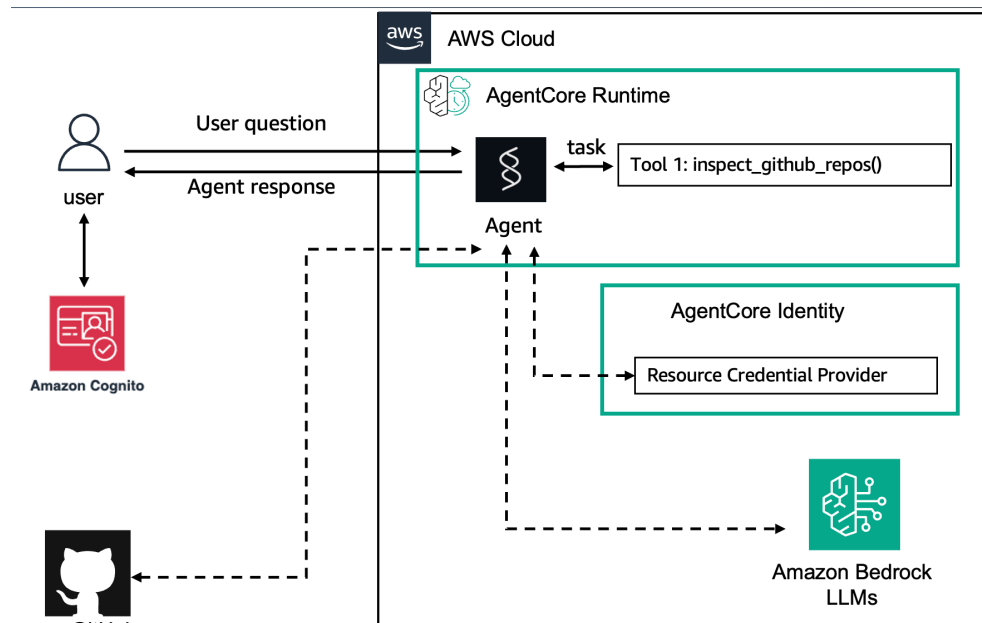
User-Delegated Access with Cognito and 3-Legged OAuth flow with GitHub

© 2008 - 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy policy](#) [Terms of use](#) [Cookie preferences](#)

Introduction

In this Lab, we will focus on implementing Outbound Auth using 3-Legged OAuth (3LO) in AgentCore. We'll demonstrate this OAuth authorization code grant flow by configuring an agent to securely access user's private GitHub repositories through AgentCore's authentication system. You will learn how to set up 3-Legged OAuth to handle user consent and interaction, enabling your agent to access private repositories and GitHub resources that require explicit user permission.

High-Level Architecture



Architecture Flow:

1. **User Initiation:** User makes a request that requires access to a protected resource (private GitHub repositories, user data, etc.)
2. **Authorization Redirect:** The Agent/System redirects the user to the GitHub authorization server
3. **Authorization Code Return:** After successful authentication, GitHub returns an authorization code
4. **Token Exchange:** The application exchanges the authorization code for access and refresh tokens
5. **Resource Access:** Using the access token, the application can now access the user's private GitHub repositories
6. **Response Processing:** Data from GitHub is processed and results are returned to the user

Framework-Agnostic Implementation


AgentCore Outbound Auth works with multiple agentic frameworks

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 03-AgentCore-identity/06-Outbound_Auth_Github/runtime_with_strands_and_egress_github_3lo.ipynb

Self-paced

Here's a notebook that lets you try out the above: [Outbound Auth with GitHub 3LO Example](#) .

Congratulations!

You have successfully implemented Outbound Auth for your AgentCore Runtime agent using user-delegated access with Cognito and 3-legged OAuth flow with GitHub. You learned how to:

- Create and configure resource credential providers
- Use decorators to retrieve credentials securely
- Deploy agents that can access external services
- Understand the security benefits of centralized credential management

Your completed implementation demonstrates how AgentCore Identity simplifies secure external service integration while maintaining the highest security standards.

[Previous](#)[Next](#)