



Building Intelligent Customer Service with Agentic AI on Amazon Connect

- ▼ Welcome to AnyCompany Hotels
 - Choose Your Deployment Path
 - Prerequisites Setup (Path B Only)
- ▼ Building the Reservation System
 - Task 1: Get Your Stack Outputs
 - Task 2: Creating Your MCP Server**
 - Task 3: Associating Your MCP Server with Amazon Connect
 - Task 4: Enable Customer Profiles
 - Task 5: Access Amazon Connect Admin Interface
 - Task 6: AI Agent Setup
 - Task 7: Configure AI Prompt
 - Task 8: Create Lex Bot for Connect
 - Task 9: Build Your Flow
 - Task 10: Create Your Customer Profile
 - Task 11: Associate a Phone Number
 - Task 12: Test Your AI Agent
 - Task 13: Experience Agentic Voice
- ▼ Handling Customer Questions
 - Task 1: Upload FAQ Documents
 - Task 2: Add the Retrieve Tool
 - Task 3: Test Knowledge Base Queries
- ▼ Human Escalation
 - Task 1: Configure the Escalate Tool
 - Task 2: Build the Escalation Flow
 - Understanding Step-by-step Guides (Optional)
 - Task 3: Test Escalation
- ▼ Experiment and Explore
 - Experiment 1: Change Your Agent Personality
 - Experiment 2: Custom Voices

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

Content preferences

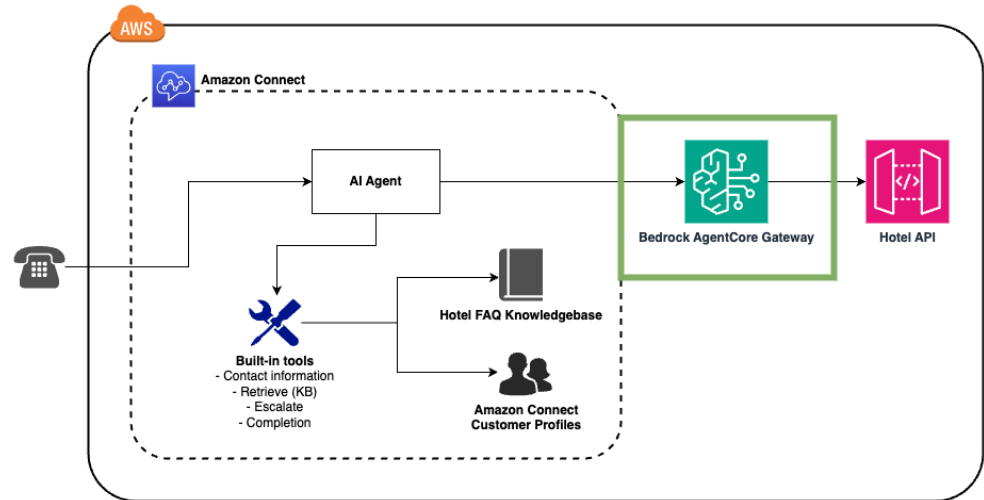
Language

English

[Building Intelligent Customer Service with Agentic AI on Amazon Connect](#) > [Building the Reservation System](#) > T...

Task 2: Creating Your MCP Server

The first task you'll need to do is create an MCP server that acts as the interface between the AI Agent we'll create later and AnyCompany Hotel's API. The API and backend database have been deployed and you create a Bedrock AgentCore Gateway for it.



The API

Your MCP server will provide these core functions to the AI agent based on the AnyCompany's API:

Hotel Search Functions

- `searchHotels` - Search for hotels in a specific city

Customer Service Functions

- `getCustomerReservations` - Retrieve customer's existing reservations

Booking Management Functions

- `createBooking` - Create new hotel reservations
- `modifyBooking` - Update existing reservations
- `cancelBooking` - Cancel reservations

Step 1: Understand Pre-Deployed Hotel API

Your workshop environment includes a hotel reservation API that's ready to use. The CloudFormation templates have been deployed to your account with the backend infrastructure.

Pre-Deployed Resources:

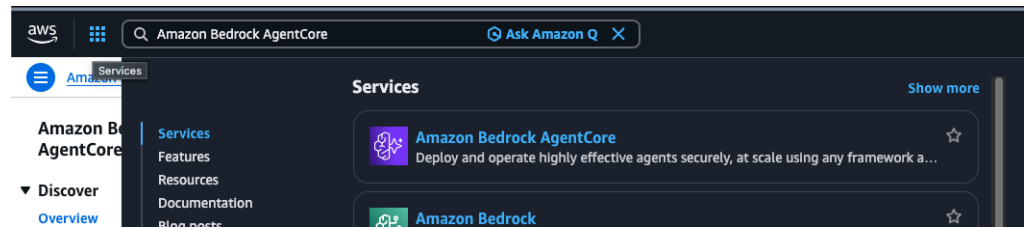
Your workshop account includes:

- Hotel API with Lambda functions (with mock data)
- IAM roles with appropriate permissions (ready to use)
- DynamoDB tables with mock customer and reservation data

- CloudWatch logging for monitoring (active and collecting data)

Step 2: Navigate to AgentCore Gateway

Open the **AWS Console** and search for "Amazon Bedrock AgentCore" and click.



Step 3: Add API Key to AgentCore

The API that has been deployed to your account has created an API key for providing AgentCore Gateway access to your API. We'll need to add that API Key to the identity section of AgentCore before creating our gateway.

1. Click **Identity** in the left pane navigation under Build.
2. Click the **Add OAuth client / API key** button and select **Add API key**
3. **Create API Key details:**
 - **Name:** hotel-api-key
 - **API key:** Paste the **HotelApiKey** value from your workshop-values.txt file.
4. Click **Add** to save.

Step 4: Create Your MCP Server

1. Click **"Gateways"** in the left hand navigation pane
2. Click **Create gateway** button
3. **Enter the Gateway details:**
 - **Gateway Name:** anycompany-hotels-mcp-server
 - **Description (in Additional configurations):** MCP server for AnyCompany Hotels reservation system

 A screenshot of the 'Create gateway' form in the AWS Console. The form is divided into several sections:

- Gateway details:** Includes a text input for 'Gateway name' with the value 'anycompany-hotels-mcp-server'.
- Additional configurations - optional:**
 - Gateway description:** A text input with the value 'MCP server for AnyCompany Hotels reservation system'.
 - Instruction:** A text input with the value 'Enter instruction'.
 - Enable semantic search:** A checkbox that is unchecked.
 - Exception level debug:** A checkbox that is checked.
 - Request interceptor Lambda ARN:** A text input with the value 'Enter Lambda ARN'.
 - Response interceptor Lambda ARN:** A text input with the value 'Enter Lambda ARN'.

4. **Inbound auth** - This controls how Amazon Connect authenticates when calling your gateway.
 - Below in the Inbound Auth configuration section select **Use JSON Web Tokens (JWT)**

- Select Use existing identity provider configurations and in the **Discovery URL** field, paste the **AgentCoreGatewayDiscoveryUrl** value from your `workshop-values.txt` file.
- **Check** Allowed audiences and put placeholder `placeholder` as text in the audiences box. *Don't worry—we'll come back to this shortly.*
- **Uncheck** the Allowed clients checkbox

▼ Inbound Auth configurations Info

Inbound Auth is the gatekeeper that controls who can access this hosted resource. Define how access is granted to this gateway.

Inbound Auth type

- ☐ Use IAM permissions
This gateway will perform authentication and authorization using AWS Signature Version 4 (SigV4). No further configurations are necessary.
- ☒ Use JSON Web Tokens (JWT)
This gateway will perform authentication and authorization using Identity. Control access to the AgentCore Gateway resource by using a JWT (like OAuth) based authorization server.
- ☐ No authorization
Opt out of managed identity for inbound requests. This gateway will be publicly accessible with no access control.

JWT schema configuration

☐ Quick create configurations with Cognito - recommended
Inbound Auth configurations will be created on your behalf with Cognito as the identity provider.

☒ Use existing identity provider configurations
Bring existing Inbound Auth configurations from any identity provider to enable OAuth 2.0.

Discovery URL

Enter the Discovery URL from your auth provider. This link lets us automatically fetch login, token, and verification settings.

`https://amazon-connect-123456789012-0e5b7acf49c1.my.connect.aws/well-known/openid-configuration`

This URL can be typically found in your auth provider's dashboard or documentation

JWT Authorization Configuration

☒ Allowed audiences
This is used to validate that the audiences specified for the OAuth token matches or are a subset of the audiences specified in the AgentCore Gateway.

Audiences

`placeholder`

[Add audience](#)

☐ Allowed clients
This is used to validate that the public identifier of the client, as specified in the authorization token, is allowed to access the AgentCore Gateway.

5. **Permissions** Choose to create a new service role (should be selected by default)

Configure API Target:

Now you'll configure your MCP server to connect to the Hotel API. This step establishes the secure connection between AgentCore Gateway and your backend services.

1. Basic Target Configuration:

- **Target Name:** anycompany-hotels-api `placeholder`
- **Description:** AnyCompany Hotels reservation management API `placeholder`
- **Target Type:** Select REST API

2. Schema Configuration:

- **API Schema Type:** Select OpenAPI schema
- **Schema Source:** Choose Define with an S3 resource
- **S3 URI:** Paste the **HotelApiOpenApiSpecUrl** value from your `workshop-values.txt` file. (See the [Hotel API OpenAPI Schema](#) reference if you'd like to explore the API specification.)

3. Authentication Configuration:

Since your Hotel API requires an API key for authentication, you'll set authentication to the API key you created previously:

Outbound Auth configurations: The outbound auth defines how Bedrock AgentCore Gateway authenticates to our API.

- **Authentication Type:** Select API Key
- **API Key:** Select the API key you created earlier in the dropdown.

4. Click **Create gateway!**

Step 5: Update the Inbound Audience

Earlier we added `placeholder` as our inbound identity allowed audience. Now that the gateway is created we'll replace it with the actual Gateway ID.

1. **Copy the Gateway ID** from the Gateway details section.

- Inbound Auth configurations

info

Inbound Auth is the gatekeeper that controls who can access this hosted resource. Define how access is granted to this gateway.

Inbound Auth type

Use JSON Web Tokens (JWT)

JWT schema configuration

Use existing identity provider configurations

Discovery URL

Enter the Discovery URL from your auth provider. This link lets us automatically fetch login, token, and verification settings.

https://amazon-connect-123456789012-0e5b7acf49c1.my.connect.aws/.well-known/openid-configuration

This URL can be typically found in your auth provider's dashboard or documentation

JWT Authorization Configuration

☒ Allowed audiences

This is used to validate that the audiences specified for the OAuth token matches or are a subset of the audiences specified in the AgentCore Gateway.

Audiences

anycompany-hotels-mcp-server-xxxxxxxxxxxxx

Add audience

☐ Allowed clients

This is used to validate that the public identifier of the client, as specified in the authorization token, is allowed to access the AgentCore Gateway.

Excellent! You've successfully created AgentCore Gateway MCP server. Your server is now ready to connect to Amazon Connect and provide intelligent hotel reservation capabilities to your AI agent.

Ready to make the connection? Let's associate your MCP server with Amazon Connect.