

Smart Service AI Assistant – Technical Documentation

Amazon Bedrock • Knowledge Bases • OpenSearch Serverless • Lambda • SES •
Serverless AI Architecture

1. Executive Summary

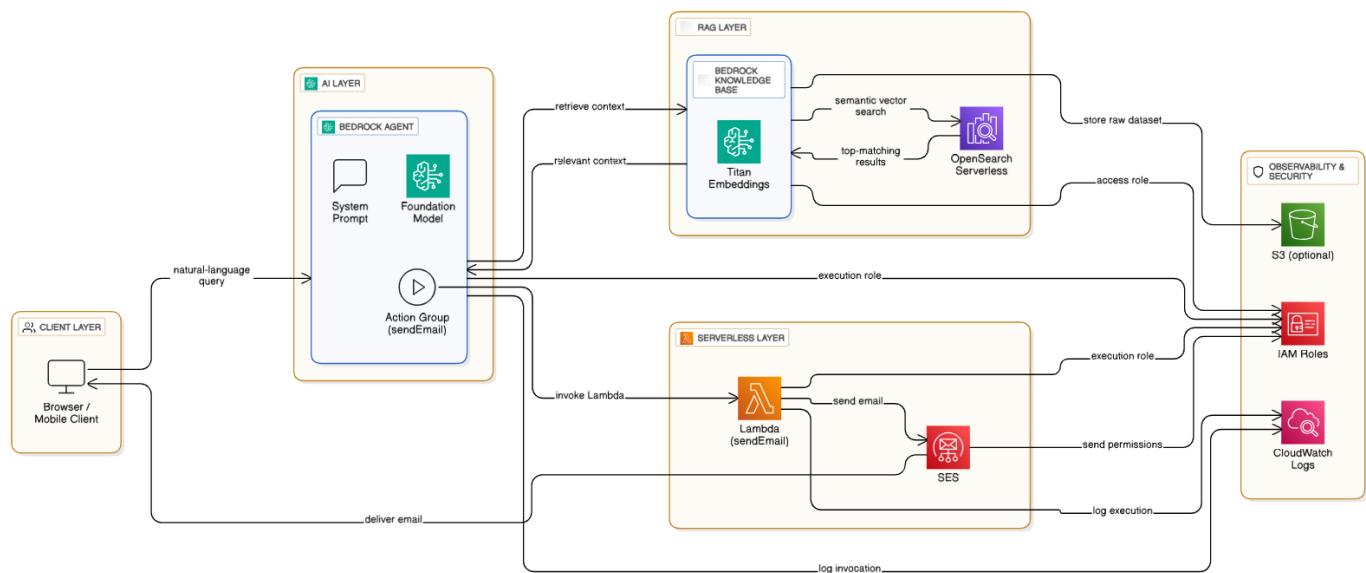
The **Smart Service AI Assistant** is an intelligent, serverless, generative AI system built on **Amazon Bedrock Agents**, **Knowledge Bases**, **OpenSearch Serverless**, **AWS Lambda**, and **Amazon SES**.

The assistant interprets natural language questions, retrieves contextual information using RAG (Retrieval-Augmented Generation), generates conversational responses, and performs real-world actions such as sending emails.

The solution demonstrates how agentic AI, vector retrieval, and serverless compute can be combined to build production-grade intelligent assistants suitable for enterprise workflows, customer support automation, and personalized recommendation systems.

2. Architecture Overview

The solution is designed as a fully managed, serverless AI pipeline:



Core Architecture Components

Component	Purpose
Amazon Bedrock Agent	Interprets user queries, orchestrates tools, and generates human-like responses.
Knowledge Base (KB)	Stores unstructured text (food deals, restaurant offers, shopping/electronics insights). Uses embeddings for semantic search.
Amazon OpenSearch Serverless	Vector index backend that enables fast, relevant retrieval using approximate nearest neighbor (ANN) search.
Lambda Action Group	Custom tool integrated into the agent for executing backend logic (e.g., sending emails).
Amazon SES	Delivers formatted emails triggered by the AI assistant.
Amazon S3	Stores dataset files and KB documents for ingestion.
IAM Roles & Policies	Secures access across all services using least-privilege design.

The architecture is fully serverless, automatically scalable, and cost-optimized.

3. System Workflow

End-to-End Request Flow

1. User Query

The user asks: “Find me food deals and send them to my email.”

2. Bedrock Agent Invocation

The agent parses intent and determines required tools (RAG + email action).

3. RAG Retrieval via Knowledge Base

- Query is embedded using the model.
- Embedding is matched against documents stored in **OpenSearch Serverless**.
- Relevant offers/deals are retrieved.

4. Generative Response

Bedrock synthesizes a clean, conversational answer without citations or technical metadata.

5. Action Group Execution

If the user requests “send it to my email,” the agent triggers the **sendEmail** tool.

6. Lambda Action Group

- Lambda receives structured parameters from the agent.
- The message is formatted into HTML/text.
- SES is invoked to send the email.

7. SES Email Delivery

Email is delivered securely to the user.

8. Final AI Response

The agent confirms delivery and returns the final message.

4. Key Features & Capabilities

- Natural Language Understanding using Amazon Bedrock foundation models
 - RAG-powered retrieval using **vector embeddings**
 - Clean, citation-free responses for production use
 - Automatic semantic search for unstructured content (deals, offers, recommendations)
 - Custom **Lambda Action Groups** for business logic
 - Secure, automated email delivery using Amazon SES
 - Fully serverless, scalable architecture
 - Plug-and-play integration for mobile/web applications
-

5. Technologies & AWS Services Used

- **Amazon Bedrock** (Agents, Knowledge Bases, Embeddings)
 - **Amazon OpenSearch Serverless** (Vector Storage & Indexing)
 - **AWS Lambda** (Action Group backend)
 - **Amazon SES** (Transactional email delivery)
 - **Amazon S3** (Document storage for KB ingestion)
 - **IAM** (Fine-grained role-based access control)
 - **Python** (Lambda handler code)
-

6. Troubleshooting & Challenges Resolved

1. Knowledge Base Indexing Failures

- Issue: Incorrect embedding model configuration and OpenSearch vector settings
- Fix: Updated space_type, dimensions, and vector field mappings in the KB configuration

2. Lambda Permission Errors

- Issue: Lambda could not call SES

- Fix: Added ses:SendEmail and ses:SendRawEmail IAM permissions

3. Retrieval Quality Issues

- Issue: KB returned irrelevant or empty results
- Fix: Improved dataset formatting and removed noisy text

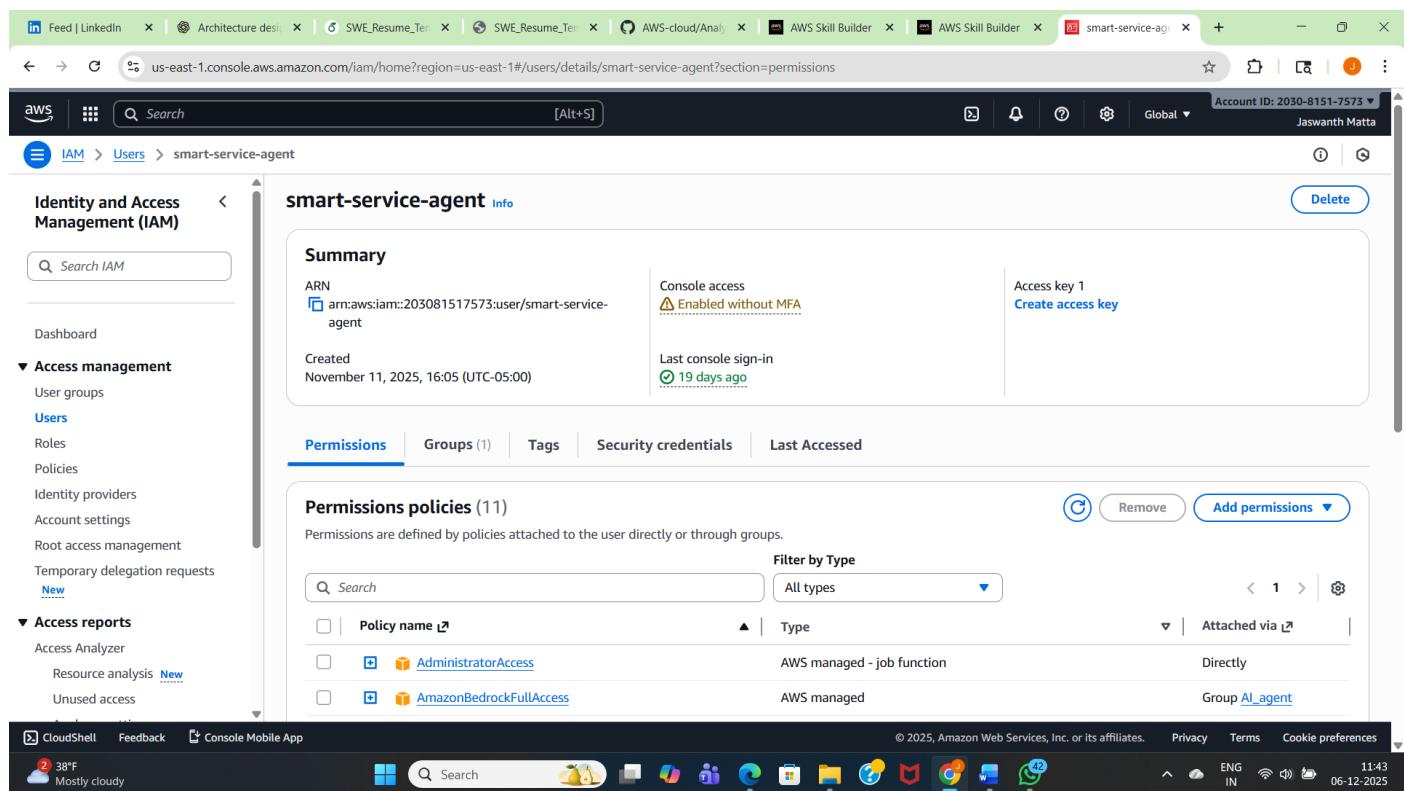
4. Agent Tool Invocation Not Triggering

- Issue: Improper action schema in Bedrock Agent
- Fix: Redefined JSON schema and validated parameter mapping

These represent real engineering debugging scenarios that demonstrate production-level troubleshooting.

7. Implementation Screenshots

This screenshot shows the IAM user **smart-service-agent** configured with the required permissions to support Bedrock Agent operations. The attached policies provide secure access to Bedrock, Lambda, and other necessary AWS services.



The screenshot displays the AWS IAM User Details page for the user 'smart-service-agent'. The user was created on November 11, 2025, at 16:05 UTC-05:00. It shows two attached policies: 'AdministratorAccess' and 'AmazonBedrockFullAccess'. The 'Permissions' tab is selected, showing the attached policies and their details. The 'Summary' section includes information about console access (enabled without MFA) and the last console sign-in (19 days ago). The user has one access key, which is listed under 'Access key 1'.

Attached Policies
AdministratorAccess
AmazonBedrockFullAccess

This screenshot shows the Amazon Bedrock Knowledge Bases console. On the left, there's a sidebar with sections like Discover, Test, Infer, and Tune. The main area has a heading 'Knowledge Bases' and a 'How it works' section with three steps: 1. Create a Knowledge Base with (Vector store, Structured data store, Kendra GenAI Index), 2. Test (Standard retrieval, Retrieval with generation), and 3. Integrate (Integrate your Knowledge Base into your application). Below this, a table shows 'Knowledge Bases (1)' with columns for Edit, Delete, Test Knowledge Base, Evaluate, and Create. The status of the single knowledge base is 'Available'. At the bottom, there's a toolbar with CloudShell, Feedback, and Console Mobile App, along with system status icons.

This screenshot shows the Bedrock Knowledge Base successfully created and synced with an OpenSearch Serverless vector database. The Knowledge Base is now in an **Available** state, ready to perform semantic retrieval for RAG-based queries.

This screenshot shows the Amazon Bedrock Knowledge Bases console after creating a new knowledge base named 'knowledge-base-quick-start-myagent'. A green success message indicates that the OpenSearch Serverless vector database is ready and the knowledge base was created successfully. The knowledge base overview table includes fields for Knowledge Base name (knowledge-base-quick-start-myagent), Knowledge Base ID (ABE4KTBA25), Knowledge Base description (empty), Service Role (AmazonBedrockExecutionRoleForKnowledgeBase_ju6bv), Status (Available), and Created date (November 14, 2025, 10:33 (UTC-05:00)). The Data source section shows one source (Synced) with an 'Add' button. The bottom of the screen shows a toolbar with CloudShell, Feedback, and Console Mobile App, along with system status icons.

Screenshot of the AWS Bedrock Knowledge Base overview page for 'knowledge-base-quick-start-myagent'.

Knowledge Base Overview:

- Knowledge Base name:** knowledge-base-quick-start-myagent
- Knowledge Base ID:** ABE4KTBA25
- Status:** Available
- Service Role:** AmazonBedrockExecutionRoleForKnowledgeBase_ju6bv
- Created date:** November 14, 2025, 10:33 (UTC-05:00)

Data source (1): Synced successfully. Data sources contain information returned when querying a Knowledge Base.

	Data so...	Status	Data sour...	Account ID	Source Link	Last sync ...	Last sync ...	Chunking...	Parsing st...	Data
<input checked="" type="checkbox"/>	knowledge... Available	S3	20308151...	s3://myag...	-	-	-	Default	Default	Delete

Tags: 0 tags assigned.

CloudShell: WMT -1.44% **Feedback:** © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences **10:37 14-11-2025**

Screenshot of the AWS Bedrock Knowledge Base overview page for 'knowledge-base-quick-start-myagent' after a sync operation.

Sync completed for data source - 'knowledge-base-myagent-data-source'

knowledge-base-quick-start-myagent

Knowledge Base Overview:

- Knowledge Base name:** knowledge-base-quick-start-myagent
- Knowledge Base ID:** ABE4KTBA25
- Status:** Available
- Service Role:** AmazonBedrockExecutionRoleForKnowledgeBase_ju6bv
- Created date:** November 14, 2025, 10:33 (UTC-05:00)

Data source (1): Synced successfully. Data sources contain information returned when querying a Knowledge Base.

	Data so...	Status	Data sour...	Account ID	Source Link	Last sync ...	Last sync ...	Chunking...	Parsing st...	Data
<input checked="" type="checkbox"/>	knowledge... Available	S3	20308151...	s3://myag...	-	-	-	Default	Default	Delete

CloudShell: WMT -1.44% **Feedback:** © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences **10:38 14-11-2025**

The screenshot shows the Amazon Bedrock Agent builder interface. On the left, a sidebar lists various features: Discover (Overview, Model catalog, API keys), Test (Chat / Text playground, Image / Video playground, Watermark detection, Tokenizer New), Infer (Cross-region inference, Batch inference, Provisioned Throughput, Custom model on-demand New), and Tune (Custom models, Prompt router models, Imported models). The main area is titled "Agent builder" and shows a success message: "Agent agent-1234 was successfully created." Below this, there are tabs for "Manual" (selected), "Assistant", "Test" (highlighted in blue), "Prepare", and "Save". The "Agent details" section includes fields for "Agent name" (agent-1234), "Agent description - optional" (Enter description), "Agent resource role" (Create and use a new service role selected), and a dropdown for "AmazonBedrockExecutionRoleForAgents_VBK4PCMQSXJ". A "Select model" dropdown is also present. To the right, a "Test Agent" panel shows a message input field ("Enter your message here") and a "Run" button. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date and time (14-11-2025).

This screenshot shows the Bedrock Agent being configured with the newly added Knowledge Base, enabling the agent to retrieve context using RAG during conversation. The agent is assigned the required execution role and is ready for preparation and testing.

The screenshot shows the Amazon Bedrock Agent builder interface. The sidebar and main configuration area are identical to the previous screenshot, with the "Test" tab selected. A success message at the top states: "Successfully added Knowledge Base knowledge-base-quick-start-myagent to agent. Prepare the agent to keep its details up to date." The "Agent resource role" section now shows "arn:aws:iam::203081517573:role/service-role/AmazonBedrockEx...". The "Test Agent" panel now includes a "Prepare" button. The bottom of the screen shows a Windows taskbar with various icons and a system tray indicating the date and time (14-11-2025).

Screenshot of the Amazon Bedrock Agent Details page showing the Agent builder interface.

Left Sidebar:

- Amazon Bedrock**
- Discover**: Overview, Model catalog, API keys
- Test**: Chat / Text playground, Image / Video playground, Watermark detection, Tokenizer [New](#)
- Infer**: Cross-region inference, Batch inference, Provisioned Throughput, Custom model on-demand [New](#)
- Tune**: Custom models, Prompt router models, Imported models

Top Bar: Search, [Alt+S], United States (N. Virginia), Account ID: 2030-8151-7573, smart-service-agent

Agent builder Info:

- Manual | Assistant | **Test** | Prepare | Save | Save and exit

Agent details:

- Agent name:** agent-1234 (Valid characters are a-z, A-Z, 0-9, _, (underscore) and - (hyphen). The name can have up to 100 characters.)
- Agent description - optional:** Enter description (The description can have up to 200 characters.)
- Agent resource role:** Create and use a new service role

Test Agent:

Using ODT Chan

Enter your message here

Run

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CloudShell Feedback 45°F Sunny ENG IN 10:47 14-11-2025

Screenshot of the Amazon Bedrock Test Agent interface showing a trace session.

Left Sidebar:

- Amazon Bedrock**
- Discover**: Overview, Model catalog, API keys
- Test**: Chat / Text playground, Image / Video playground, Watermark detection, Tokenizer [New](#)
- Infer**: Cross-region inference, Batch inference, Provisioned Throughput, Custom model on-demand [New](#)
- Tune**: Custom models, Prompt router models, Imported models

Top Bar: Search, [Alt+S], United States (N. Virginia), Account ID: 2030-8151-7573, smart-service-agent

Test Agent:

Using ODT Change

"What are the best restaurants around here?"

Based on the available information, here are some of the best restaurants around here:^[1] For burgers, Five Guys is highly recommended for quality, while Wendy's and McDonald's offer budget-friendly combos.^[2] Taco Bell is popular for spicy food lovers with its \$2 burrito and \$1 menu items.^[3] Chipotle bowls are consistently recommended for healthier meals, especially the double-protein options.^[4] Chick-fil-A is known for clean and reliable chicken sandwiches and is often rated highest for customer service.^[5] For pizza cravings, Domino's is usually the fastest delivery, while Little Caesars is the cheapest for walk-in and carryout.^[6] Starbucks and Dunkin' are the most common coffee shops near most

Trace:

Trace (0) | Session summaries | [Copy all](#)

Pre-Processing Trace:

No trace steps available
Select a response from the left to see the trace here.

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

CloudShell Feedback 45°F Sunny ENG IN 10:50 14-11-2025

This screenshot shows the AWS Lambda function overview for 'smart-service-lambda'. The function was successfully created and is now ready for configuration and testing. Key details include:

- Description:** None
- Last modified:** 3 seconds ago
- Function ARN:** arn:aws:lambda:us-east-1:203081517573:function:smart-service-lambda
- Function URL:** None

The interface includes tabs for Function overview, Code, Test, Monitor, Configuration, Aliases, and Versions. On the right, there's a 'Tutorials' sidebar with a 'Create a simple web app' section and a 'Start tutorial' button.

This screenshot shows the Lambda function **smart-service-lambda**, which processes Bedrock Agent actions to send emails or SMS based on the user request. The function uses boto3 to interact with Amazon SES and SNS, enabling automated message delivery as part of the agent workflow.

This screenshot shows the AWS Lambda function code editor for 'smart-service-lambda'. The code is written in Python (boto3) and handles Bedrock Agent actions to send SMS or Email. The code uses the sns and ses clients from boto3 to interact with Amazon SES and SNS respectively.

```
1 import boto3
2
3 sns = boto3.client("sns")
4 ses = boto3.client("ses")
5
6 def lambda_handler(event, context):
7
8     action = event.get("action")
9     message = event.get("message")
10    phone = event.get("phone")
11    email = event.get("email")
12
13    # Send SMS
14    if action == "sendSMS" and phone and message:
15        sns.publish(PhoneNumber=phone, Message=message)
16        return {"status": "SMS sent"}
17
18    # Send Email
19    if action == "sendEmail" and email and message:
20        ses.send_email(
21            Source=email,
22            Destination={"ToAddresses": [email]},
23            Message={
24                "Subject": {"Data": "Deals for you"},
25                "Body": {"Text": {"Data": message}}
26            },
27        )
```

The screenshot shows the AWS Lambda console interface. A green success message at the top states: "The test event 'testSendSMS' was successfully saved." The main area displays the code source for a Python function named `lambda_function.py`:

```

def lambda_handler(event, context):
    ...

```

The status is listed as "Succeeded" and the test event name is "testSendSMS". The response is a JSON object with a single key "status": "SMS sent". Function logs show a successful execution with a duration of 226.48 ms and memory usage of 128 MB. The request ID is 5d28fabb-34eb-44c4-af75-ee1fc51d585.

On the right side, there's a sidebar titled "Tutorials" with a section for "Create a simple web app". It includes a list of steps: "Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage" and "Invoke your function through its function URL". There are "Learn more" and "Start tutorial" buttons.

The screenshot shows the AWS IAM console interface. A green success message at the top states: "Policies have been successfully attached to role." The main area displays a table of attached policies:

Policy name	Type	Attached entities
AmazonBedrockFullAccess	AWS managed	3
AmazonSESFullAccess	AWS managed	1
AmazonSNSSFullAccess	AWS managed	2
AWSLambdaBasicExecutionRole-7c3568fa...	Customer managed	1

Below the table, there are sections for "Permissions boundary (not set)" and "Generate policy based on CloudTrail events". A note says: "You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy." A "Generate policy" button is available, and a note states: "No requests to generate a policy in the past 7 days."

General | YouTube (189) | AWS Multi | Cloud | AWS | AWS CloudFormation | Create | Inbox | Works | smart | Add a... | smart | Text | PI | X

203081517573-fkt6azg4.us-east-1.console.aws.amazon.com/sms-voice/home?region=us-east-1#/numbers?action=request-originator

aws Search [Alt+S] United States (N. Virginia) Account ID: 2030-8151-7573 smart-service-agent

AWS End User Messaging > SMS > Phone numbers > Request originator

Step 1 Select country
Step 2 Define use case
Step 3 Select originator type
Step 4 Review and request

Review and request Info

Step 1: Select country Edit

Country of origination

Country: United States

Step 2: Define use case Edit

Messaging use case

Estimated monthly message volume Less than 5000	Company Headquarters Local	Two-way SMS messaging No	International sending No
----------------------------------------------------	-------------------------------	-----------------------------	-----------------------------

Step 3: Select originator type Edit

Number type

Product Long code	Registration	Number capabilities Voice	Cost estimate View pricing details
----------------------	--------------	------------------------------	-------------------------------------------------------

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 47°F Mostly cloudy ENG IN 12:31 14-11-2025

General | YouTube (189) | AWS Multi | Cloud | AWS | AWS CloudFormation | Create | Inbox | Works | smart | Add a... | smart | Text | PI | X

203081517573-fkt6azg4.us-east-1.console.aws.amazon.com/sms-voice/home?region=us-east-1#/numbers?org-identity=phone-adda3308c0cd439695d9652f99368a5f

aws Search [Alt+S] United States (N. Virginia) Account ID: 2030-8151-7573 smart-service-agent

AWS End User Messaging > SMS > Phone numbers > +12065576782

AWS End User Messaging

SMS

- Overview
- Dashboard
- Shortcuts

Configurations

- Phone pools
- Phone numbers**
- Sender IDs
- Registrations
- Configuration sets
- Opt-out lists

Protect

- Protect configurations New

Social messaging

Push notifications

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 47°F Mostly cloudy ENG IN 12:42 14-11-2025

+12065576782 Info

Success
New long code +12065576782 was successfully added to your account.

Release phone number **Add to phone pool**

Number details Info

Capability Voice	Phone number ID phone-adda3308c0cd439695d9652f99368a5f	Status Pending
Phone number type Long code	Amazon Resource Name (ARN) arn:aws:sms-voice:us-east-1:203081517573:phone-number/phone-adda3308c0cd439695d9652f99368a5f	Creation date November 14, 2025 12:41 PM (UTC-05:00)
Message type Transactional	Phone pool	Monthly fee \$1.00
Country United States		

Two-way SMS Info **Edit settings**

Enabling two-way SMS allows you to receive incoming text messages from your end recipients.

General Case Cloud AWS AWS CloudFormation Lambda Functions smart-service-lambda smart-service-agent

203081517573-fkt6azg4.us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/smart-service-lambda?newFunction=true&tab=code

aws Search [Alt+S] United States (N. Virginia) Account ID: 2030-8151-7573 smart-service-agent

Lambda > Functions > smart-service-lambda

Successfully updated the function smart-service-lambda.

The test event "testEmail" was successfully saved.

EXPLORER SMART-SERVICE-LAMBDA lambda_function.py

lambda_function.py

```
1 import hotos
```

PROBLEMS OUTPUT CODE REFERENCE LOG TERMINAL Execution Results

Status: Succeeded Test Event Name: testSendsSMS

Response:

```
{ "status": "Invalid request or missing parameters" }
```

Function Logs:

```
START RequestId: f650a642-2d7a-4a8b-bbff-255a5ef2596c Version: $LATEST
END RequestId: f650a642-2d7a-4a8b-bbff-255a5ef2596c
REPORT RequestId: f650a642-2d7a-4a8b-bbff-255a5ef2596c Duration: 1.87 ms Billed Duration: 460 ms Memory Size: 128 MB
Max Memory Used: 84 MB Init Duration: 458.13 ms

Request ID: f650a642-2d7a-4a8b-bbff-255a5ef2596c
```

TEST EVENTS (SELECTED: T...)

+ Create new test... Private saved ev... testEmail

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

47°F Mostly cloudy 13:21 14-11-2025

General Case Cloud Amazon AWS AWS CloudFormation Lambda Functions smart-service-lambda smart-service-agent

203081517573-fkt6azg4.us-east-1.console.aws.amazon.com/ses/home?region=us-east-1#/identities/jaswanthmatta344%40gmail.com?tabId=authentication

aws Search [Alt+S] United States (N. Virginia) Account ID: 2030-8151-7573 smart-service-agent

Amazon SES > Configuration: Identities > jaswanthmatta344@gmail.com

Mail Manager Journaling Streamline email compliance and reduce costs with SES Mail Manager's email journaling solution. Learn more ↗ Notifications 0 0 0 0 2 0

jaswanthmatta344@gmail.com

Summary Identity status Verification pending Amazon Resource Name (ARN) arn:aws:ses:us-east-1:203081517573:identity/jaswanthmatta344@gmail.com AWS Region US East (N. Virginia)

Recommendations Recommendations are available only for verified domain identities.

Enable Virtual Deliverability Manager to generate recommendations automatically

Authentication Notifications Authorization Configuration set Tenants Tags

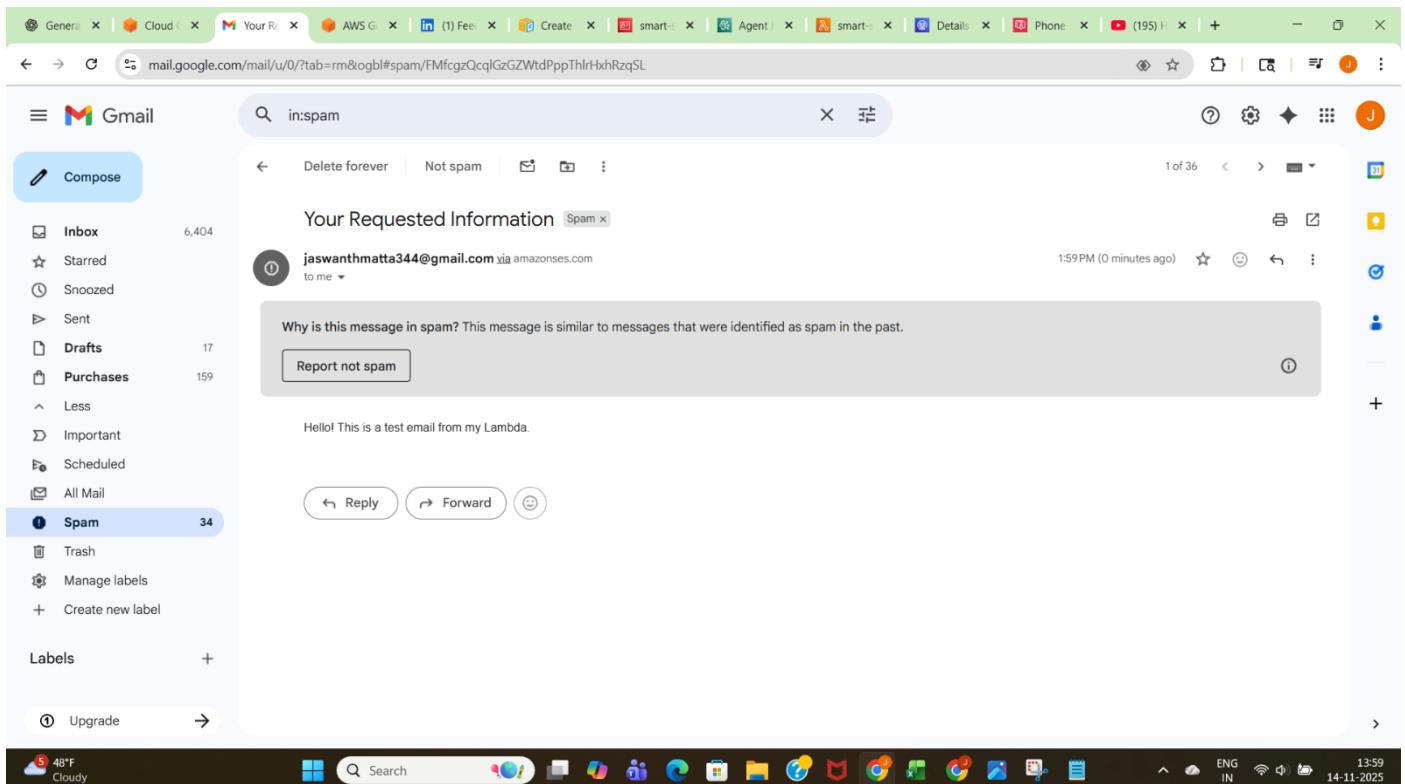
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

47°F Mostly cloudy 13:30 14-11-2025

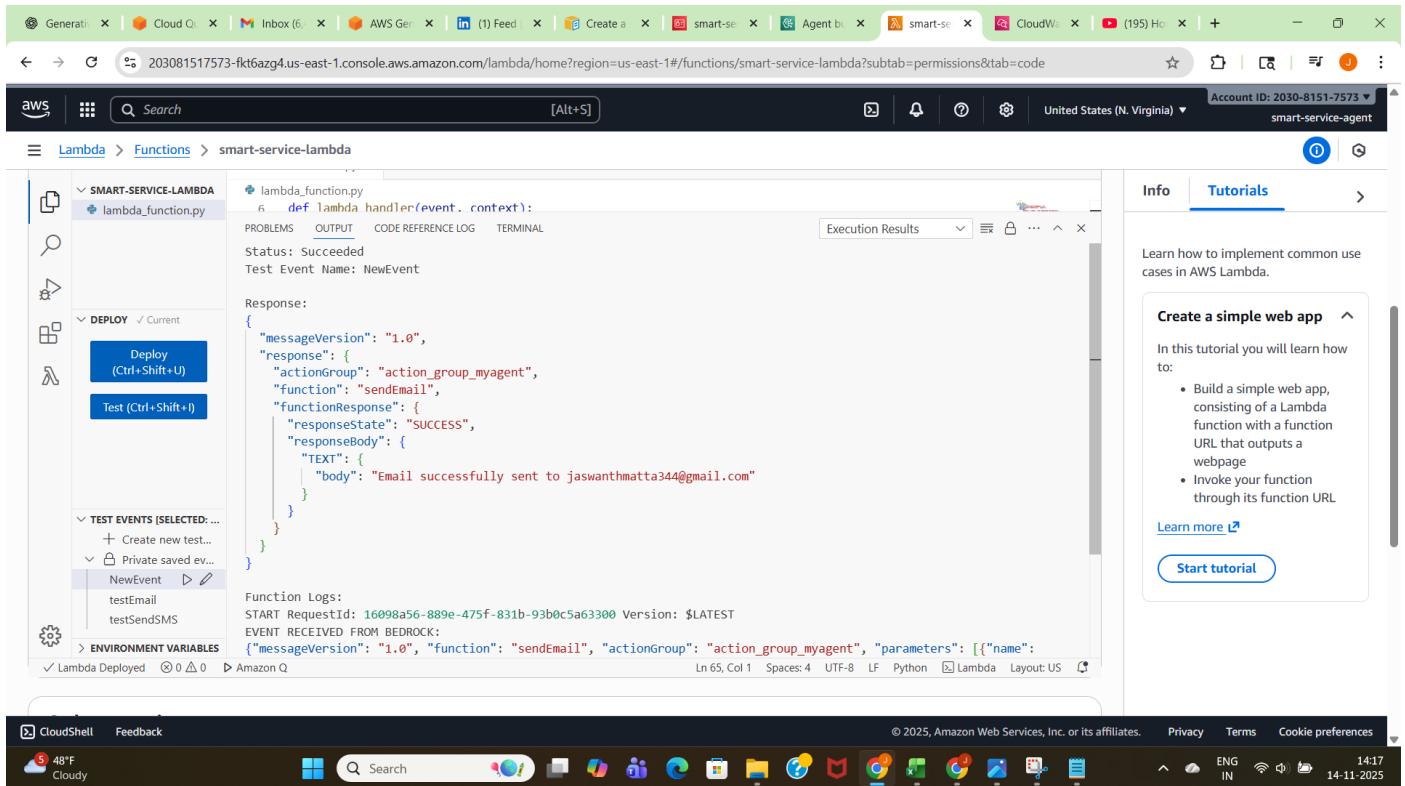
The screenshot shows the Amazon SES Identity configuration page. On the left, a sidebar lists navigation options: Get set up, Account dashboard, Reputation metrics, SMTP settings, What's new, Configuration (with Identities selected), WorkMail (Overview and Get set up), and Virtual Deliverability Manager. The main content area displays the identity details for **jaswanthmatta344@gmail.com**. It includes a summary section with Identity status (Verified), Amazon Resource Name (ARN) (arn:aws:ses:us-east-1:203081517573:identity/jaswanthmatta344@gmail.com), and AWS Region (US East (N. Virginia)). Below this is a Recommendations section stating "Recommendations are available only for verified domain identities." A blue button at the bottom right says "Enable Virtual Deliverability Manager to generate recommendations automatically". The top right shows account information (Account ID: 2030-8151-7573) and a "Send test email" button.

The screenshot shows the Gmail inbox. The left sidebar includes a Compose button, an **Inbox** folder (6,404 messages), and other sections for Starred, Snoozed, Sent, Drafts (17 messages), Purchases (159 messages), and More. Labels are listed below. The main area shows an email from **complaints@email-abuse.amazonaws.com** with the subject **FW:SEs**. The message body reads:
Hello,
This is an example of a complaint email response.
Thank you,
-Amazon SES Mailbox Simulator
----- Forwarded message -----
From: [jaswanthmatta344@gmail.com](#)
To: [complaint@simulator.amazonaws.com](#)
Cc:
Bcc:
Date: Fri, 14 Nov 2025 18:34:36 +0000
Subject: SEs

The screenshot shows the AWS Lambda console interface for the 'smart-service-lambda' function. The top navigation bar includes tabs for General, CloudShell, AWS Lambda, and more. The main title is 'smart-service-lambda'. On the left, there's a sidebar with 'Function overview' (selected), 'Code', 'Test', 'Monitor', 'Configuration' (selected), 'Aliases', and 'Versions'. The 'Configuration' tab is active, showing 'General configuration' and 'Execution role'. The main content area displays the function details: 'smart-service-lambda' (ARN: arn:aws:lambda:us-east-1:203081517573:function:smart-service-lambda), 'Description' (empty), 'Last modified' (33 minutes ago), 'Function ARN' (arn:aws:lambda:us-east-1:203081517573:function:smart-service-lambda), and 'Function URL' (empty). Buttons for 'Throttle', 'Copy ARN', and 'Actions' are visible. A green banner at the top says 'Your changes have been saved.' To the right, a 'Tutorials' section titled 'Create a simple web app' provides instructions and links to learn how to implement common use cases in AWS Lambda.



This screenshot shows the successful execution of the Lambda function triggered by the Bedrock Agent's **sendEmail** action, confirming that the email workflow is functioning correctly. The response output and logs indicate that the message was delivered to the specified email address without errors.



The screenshot shows a Gmail inbox with a search bar at the top containing "in:spam". There are 38 messages in total, with 1 being displayed. The message is from "jaswanthmatta344@gmail.com via amazonaws.com" and is labeled as spam. The subject is "Your Requested Information". The message body contains a link to "Report not spam" and a test message: "Hello! This is a test email from Bedrock-style event." Below the message are standard reply, forward, and delete buttons.

The screenshot shows the Amazon Bedrock Agent builder interface. On the left, there are navigation menus for Infer, Tune, and Build. The main area is titled "Agent builder" and shows two green success notifications: "Agent: agent-1234 was successfully updated. Prepare the agent to keep its details up to date." and "Agent: agent-1234 was successfully prepared." Below this, there are tabs for "Manual", "Assistant", "Test", "Prepare", "Save", and "Save and exit". The "Agent details" section includes fields for "Agent name" (set to "agent-1234") and "Agent description - optional" (with a placeholder "Enter description"). The "Agent resource role" section has two options: "Create and use a new service role" (unchecked) and "Use an existing service role" (checked), with a dropdown menu showing "arn:aws:iam::203081517573:role/service-role/AmazonBedrockEx...". To the right, there is a "Test Agent" panel with a message input field and a "Run" button, and a sidebar with a message history about burger deals.

The screenshot shows the AWS Bedrock Agent builder interface. On the left, a sidebar navigation includes sections for Chat / Text playground, Image / Video playground, Watermark detection, Tokenizer New, Infer (Cross-region inference, Batch inference, Provisioned Throughput, Custom model on-demand New), Tune (Custom models, Prompt router models, Imported models, Marketplace model deployment), and Build (Agents, Flows, Knowledge Bases, Automated Reasoning, Guardrails, Prompt Management). The main area displays two green success messages: "Agent agent-1234 was successfully updated. Prepare the agent to keep its details up to date." and "Agent agent-1234 was successfully prepared." Below these messages is the "Agent builder" section with tabs for Manual, Assistant, Test (selected), Prepare, Save, and Save and exit. The "Agent details" form contains fields for Agent name (agent-1234), Agent description - optional (Enter description), and Agent resource role (radio buttons for Create and use a new service role and Use an existing service role, with the latter selected and a dropdown menu showing arn:aws:iam::203081517573:role/service-role/AmazonBedrockEx...). To the right, the "Test Agent" section shows a message about chicken sandwich deals from McDonald's and Burger King, with a "Show trace" button below it.

This screenshot shows the email successfully delivered by Amazon SES, containing AI-generated deal recommendations retrieved through the Bedrock Knowledge Base. Although Gmail placed it in the spam folder, the content confirms that the end-to-end workflow—from query to retrieval to email delivery—is working correctly

The screenshot shows a Gmail inbox with the search term "in:spam". The left sidebar lists various inbox categories: Inbox (6,404), Starred, Snoozed, Sent, Drafts (17), Purchases (159), Less, Important, Scheduled, All Mail, Spam (34, highlighted), Trash, Manage labels, and Create new label. The main pane displays an email from "jaswanthmatta344@gmail.com via amazones.com" with the subject "Your Requested Information". The email content discusses why it was identified as spam and provides a list of burger deals under \$10. The bottom of the email includes standard Gmail interaction buttons: Reply, Forward, and a smiley face icon. The status bar at the bottom shows the date and time as 14-11-2025 and 14:23.

The screenshot shows the AWS Lambda Agent builder interface. On the left, a sidebar lists various configuration sections: Chat / Text playground, Image / Video playground, Watermark detection, Tokenizer New, Infer (Cross-region inference, Batch inference, Provisioned Throughput, Custom model on-demand), Tune (Custom models, Prompt router models, Imported models, Marketplace model deployment), and Build (Agents, Flows, Knowledge Bases, Automated Reasoning, Guardrails, Prompt Management). The main area displays the Agent Details page, specifically the Agent builder section. It includes settings for enabling or disabling the agent, a Knowledge Bases section listing one entry ('knowledge-base-quick-start-myagent' enabled), a Guardrail details section, and an Orchestration strategy section defining Default orchestration type, Default orchestration, and Default post-processing. A Test Agent panel on the right shows a conversation with the agent, responding to a user query about clothing deals.

The screenshot shows a Gmail inbox search results for 'in:spam'. The search bar indicates the results are filtered by spam. The first result is an email from 'jaswanthmatta344@gmail.com via amazonoses.com' titled 'Your Smart Service AI Assistant Update'. The email body contains a message from the AI assistant suggesting clothing deals. The Gmail interface includes a sidebar with navigation links like Compose, Inbox (6,404), Starred, Snoozed, Sent, Drafts (17), Purchases (159), Less, Important, Scheduled, All Mail, and Spam (34). The bottom of the screen shows a taskbar with various application icons and system status indicators.

The screenshot shows the AWS Bedrock Agent builder interface. On the left, there's a sidebar with sections for Infer, Tune, and Build. The main content area is titled "Action group details". It has three main sections: "Description - optional" (with a text input field and a note about character limits), "Action group type" (with two radio button options: "Define with function details" (selected) and "Define with API schemas"), and "Action group invocation" (with a note about specifying a Lambda function). To the right, there's a "Test Agent" window showing a conversation between a user ("i need only pizza") and an AI agent ("For pizza under \$15, Domino's offers a carryout special: large 1-topping pizza for \$7.99. Their mix & match deal lets you pick any 2 items for \$6.99 each (bread twists, pasta, sandwiches, pizzas, etc.). Little Caesars hot-n-ready pizza sometimes sells for \$5.55 depending on region. It is usually available without waiting."). There's also a text input field and a "Run" button at the bottom of the test agent window.

8.What I Learned

Through this project, I gained hands-on experience with:

Cloud Engineering

- Designing fully serverless architectures
- Securing service-to-service communication with IAM
- Leveraging managed services for scale and cost efficiency

Generative AI & RAG

- Building Bedrock Agents with custom tools
- Designing Knowledge Bases for semantic retrieval

- Using embeddings and vector search for unstructured data

Serverless Patterns

- Integrating Lambda with AI workflows
- Event-driven execution through agent action groups
- Using SES for automated notifications

This project significantly strengthened my skills in building **real-world AI systems** on AWS.

9. Conclusion

- The Smart Service AI Assistant demonstrates how **Generative AI, RAG, and AWS serverless services** can be combined to deliver real, actionable intelligence. It retrieves unstructured content, synthesizes it into natural responses, and performs real-world actions all without managing any infrastructure.
- This architecture can be extended to enterprise customer support, e-commerce applications, marketing automation, hospitality, or any domain requiring intelligent and automated interactions.