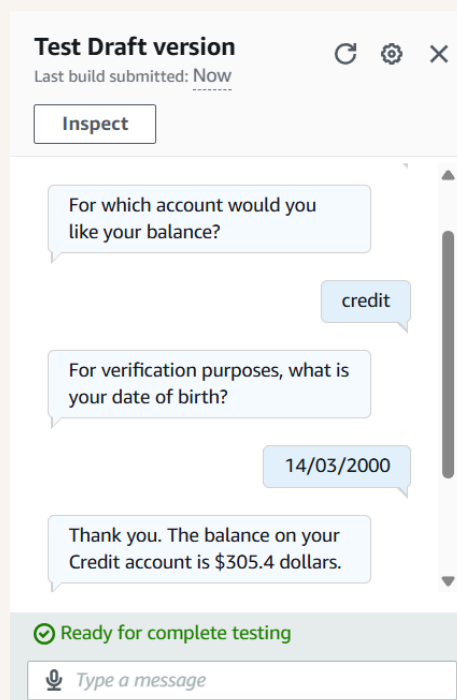




# Connect Amazon Lex with Lambda



Ashan Sooriyarachchi





# Introducing Today's Project!

## What is Amazon Lex?

Amazon Lex is useful for building scalable, AI-powered chatbots that support both text and voice interactions. It simplifies development with pre-built integrations, seamlessly connects with AWS services like Lambda and handles high use.

## How I used Amazon Lex in this project

I used Amazon Lex in today's project to build a chatbot that interacts with users and retrieves a simulated bank balance using AWS Lambda.

## One thing I didn't expect in this project was...

It was how seamlessly Amazon Lex integrates with AWS Lambda. I thought setting up the connection and handling responses would be more complex, but the Lambda function panel in TestBotAlias made it surprisingly straightforward.

## This project took me...

This project took me around two hours to complete.



# AWS Lambda Functions

AWS Lambda is a serverless computing service that automatically runs code in response to events, scaling dynamically when needed, from a few requests per day to thousands per second.

In this project, I created a Lambda function to simulate retrieving a user's bank balance. When a user asks about their account balance, Amazon Lex invokes the Lambda function, which generates a random number to represent the balance.

The screenshot shows the AWS Lambda console for the function 'BankingBotEnglish'. The left sidebar contains the 'EXPLORER' view with 'BANKINGBOTENGLISH' and 'lambda\_function.py' listed. Below it are 'DEPLOY' buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)', and a 'TEST EVENTS [NONE SELECTED]' section with a '+ Create new test event' button. The main area displays the Python code for 'lambda\_function.py'.

```
1 import json
2 import random
3 import decimal
4
5 def random_num():
6     return(decimal.Decimal(random.randrange(1000, 50000))/100)
7
8 def get_slots(intent_request):
9     return intent_request['sessionState']['intent']['slots']
10
11 def get_slot(intent_request, slotName):
12     slots = get_slots(intent_request)
13     if slots is not None and slotName in slots and slots[slotName] is not None:
14         return slots[slotName]['value']['interpretedValue']
15     else:
16         return None
17
18 def get_session_attributes(intent_request):
19     sessionState = intent_request['sessionState']
20     if 'sessionAttributes' in sessionState:
21         return sessionState['sessionAttributes']
22     return {}
23
24 def elicit_intent(intent_request, sessionState):
25     return {
26         'sessionState': sessionState,
27         'intent': get_slots(intent_request),
28         'slot': get_slot(intent_request, 'slotName'),
29         'randomNum': random_num()
30     }
```

Two notification messages are visible at the bottom right: 'Successfully updated the function BankingBotEnglish.' and 'Amazon Q Developer processes data across US Regions. See here for ...'.



# Chatbot Alias

An alias in Amazon Lex is a pointer to a specific version of the bot. When integrating Lex with AWS services or applications, external resources connect to an alias instead of a fixed bot version.

TestBotAlias is the default alias in Amazon Lex used for testing and development. It serves as a playground version of your bot, allowing you to validate functionality and troubleshoot issues before deploying updates to a live environment.

To connect Lambda with my BankerBot, I visited my bot's TestBotAlias and accessed the Lambda function panel. From there, I selected my BankingBotEnglish Lambda function.

The screenshot shows the Amazon Lex console interface for configuring a chatbot alias. At the top, a breadcrumb trail reads: Lex > Bots > Bot: BankerBot > Aliases > Alias: TestBotAlias > Alias language support: English (US). Below this, the page title is 'Alias language support: English (US)'. A section titled '▼ Lambda function - optional' contains a sub-header: 'This Lambda function is invoked for initialization, validation, and fulfillment.' Below this, there are two dropdown menus: 'Source' with 'BankingBotEnglish' selected, and 'Lambda function version or alias' with '\$LATEST' selected. A link 'Learn more about Lambda' with an external icon is located below the second dropdown. At the bottom right of the configuration panel are 'Cancel' and 'Save' buttons.

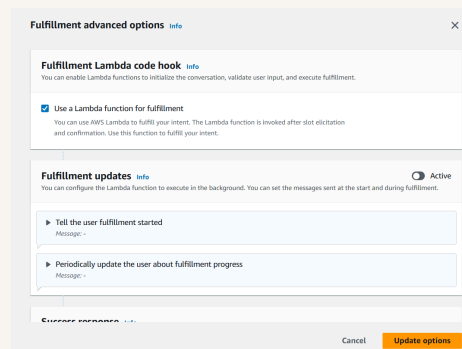


# Code Hooks

A code hook is a way to connect the chatbot to a custom Lambda function for handling specific tasks. It enables the bot to perform complex actions, such as retrieving data from a database or making decisions based on past interactions.

Even though I already connected my Lambda function with my chatbot's alias, I had to use code hooks to allow the chatbot to execute custom logic during conversations.

I could find code hooks in the Advanced Options of the Fulfillment section of my chatbot's intent configuration.





# The final result!

I've set up my chatbot to trigger Lambda and return a random dollar figure when a user asks about their account balance. Lex invokes the Lambda function, which generates a random balance and sends it back, which then delivers the response to the user

