

# Build a Banking Chatbot with Amazon Lex and Lambda

a practical chatbot, **BankerBot**, that can help bank's customers check their account balance and transfer money between accounts!

**Amazon Lex** is a service that uses deep learning to enable developers to build conversational interfaces into any application. It provides the same speech recognition, natural language understanding, and text-to-speech capabilities as those used in Amazon Alexa.

**Created a Blank Bot** in Lex with BankerBot as a name and give description for the bot.

The screenshot shows the 'Configure bot settings' page in the Amazon Lex console. The page is divided into two main sections: 'Creation method' and 'Bot configuration'. In the 'Creation method' section, there are three radio button options: 'Create a blank bot' (which is selected), 'Start with an example', and 'Start with transcripts'. The 'Bot configuration' section contains a 'Bot name' field with the value 'BankerBot' and a 'Description' field with the value 'Banker Bot to help the customers check their balance and make transfers.' The 'Description' field is optional and has a maximum character limit of 200.

Under IAM permissions, selected the role with basin Amazon Lex permissions. Amazon Lex needs the permission to call other AWS services on our behalf.

The screenshot shows the 'IAM permissions' page in the Amazon Lex console. The page is divided into two main sections: 'Runtime role' and 'New role'. In the 'Runtime role' section, there are two radio button options: 'Create a role with basic Amazon Lex permissions.' (which is selected) and 'Use an existing role.' A message box states: 'Creating a role takes a few minutes. Don't delete the role or edit the trust or permissions policies in this role until we've finished creating it.' The 'New role' section shows the role name 'AWSServiceRoleForLexV2Bots\_C2ZLW5C4QX'.

- ☐ The Idle Timeout Session – Amazon Lex will only maintain a session for a set length of time. If the user is idle and doesn't add any input for X minutes, their session will end. For now, I've chosen 5 minutes which is default.

Next, given the language and Voice for the bot. For the confidence score keep it as default 0.40

**Add language to bot** [Info](#)

▼ **Language: English (US)**

Select language  
English (US) ▼

Description - optional  
  
Maximum 200 characters.

Voice interaction  
The text-to-speech voice that your bot uses to interact with users.  
Ruth ▼

Voice sample  
Hello, my name is Ruth. Let me know how I can assist you.

Intent classification confidence score threshold  
0.40  
Min: 0.00, max: 1.00.

When using Amazon Lex to build a chatbot, this threshold is like a minimum score for the chatbot to confidently understand what the user is trying to say. Setting this to 0.4 means that your chatbot needs to be at least 40% confident that it understands what the user is asking to be able to give a response. So if a user's input is ambiguous and your chatbot's confidence score is below 0.4, it'll throw an error message.

## Creating intents

- ☐ An intent is what the user is trying to achieve in their conversation with the chatbot. For example, checking a bank account balance; booking a flight; ordering food.  
In Amazon Lex, building the chatbot by defining and categorizing different intents. If we set up different intents, one single chatbot can manage a bunch of requests that are usually related to each other.

After creating the bot, Intent page opened. Entered the Name and Description.

**Intent: NewIntent** [Info](#)

An intent represents an action that fulfills a user's request. Intents can have arguments called slots that represent variable information.

► **Conversation flow** [Info](#)

▼ **Intent details** [Info](#)

Intent name  
WelcomeIntent  
Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Intent and utterance generation description  
Describe the purpose of your intent. This will also be used when generating utterances for your intent.  
Welcoming a user when they say Hello.  
Maximum 200 characters.

ID: NWQLBKCOMB

Under Sample Utterances, provided the sample text in Plain text, the people uses while starting the chat.

**Sample utterances** (4) [Info](#)

What's this?
Generate utterances

Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

To generate utterances, you must have permissions to Amazon Bedrock. Amazon Lex will make calls to Amazon Bedrock. Additional charges may be incurred based on the usage of Amazon Bedrock. [Learn more](#)

Sort by added (ascending)

Preview Plain text

1	Hi	
2	Hello	
3	I need help	
4	Can you help me?	
5		

In the preview,

Preview Plain text

Hi

Hello

I need help

Can you help me?

Under Closing Response, Entered the message of what the bot responds to the user.

**Closing response** [Info](#)

Active

You can define the response when closing the intent.

Response sent to the user after the intent is fulfilled
Message: Hi I'm BB! Yours Banking Bot. How can I help you today?

**Message group** [Info](#)

You can define a text message group to respond using plain text.

Message

► Variations - optional

More response options

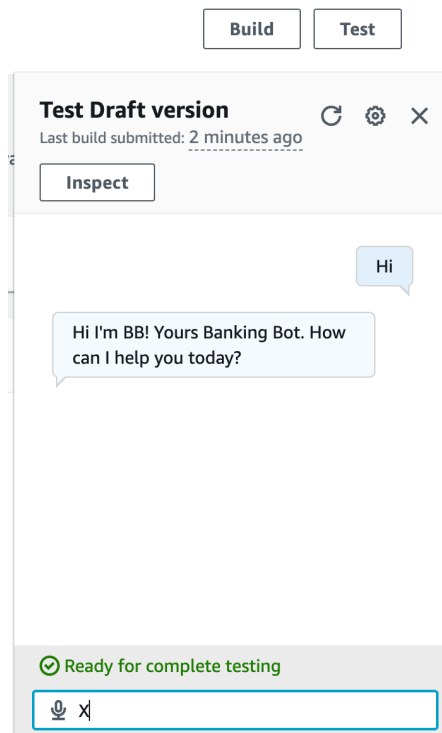
Add custom payloads, SSML, and card groups.

Set values
Next step in conversation

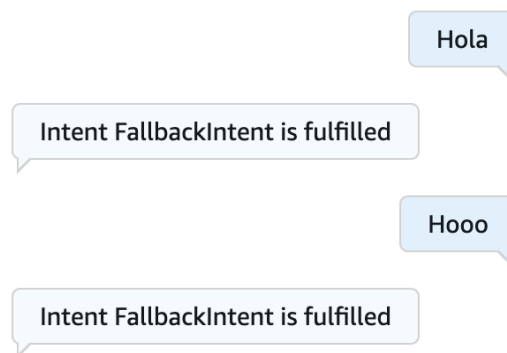
- End conversation

Add conditional branching

Now Saved Intent and Built the English(US) and then Tested.



Try other than the mentioned utterances.



the last two failed, resulting in an **Intent FallbackIntent is fulfilled** response - meaning Amazon Lex doesn't quite recognize the utterance. Instead of text, also tried with voice.

## Managing FallbackIntent

If the chatbot has a confidence score **below** 40% for all the intents defined (in our case, it's just the WelcomeIntent for now), the FallbackIntent is triggered. It is a custom error message that chatbot will use to tell the user it doesn't understand their input.

In the FallbackIntent Closing response changed the message **Intent FallbackIntent is fulfilled** to **Sorry, I am having trouble understanding. Can you describe what you would like to do in few words? I can help you find your account balance, transfer funds and make a payment.**

[Back to intents list \(2\)](#)

Search

Sort by last updated

WelcomeIntent

FallbackIntent **Unsaved**

### Closing response **Info**

You can define the response when closing the intent.

**Active**

#### ▼ Response sent to the user after the intent is fulfilled

Message: Sorry, I am having trouble understanding. Can you describe what you would like to do in few words? I can help you find your account balance, transfer funds and make a payment.

#### ▼ Message group **Info**

You can define a text message group to respond using plain text.

##### Message

Sorry, I am having trouble understanding. Can you describe what you would like to do in few words? I can help you

##### ► Variations - optional

**More response options**

Add custom payloads, SSML, and card groups.

##### ► Set values

-

##### Next step in conversation

End conversation

**+** Add conditional branching

Under Variations - optional : Provided few Variations.

→ Hmm, Could you try rephrasing that? I can help you find your account balances, transfer funds and make a payment.

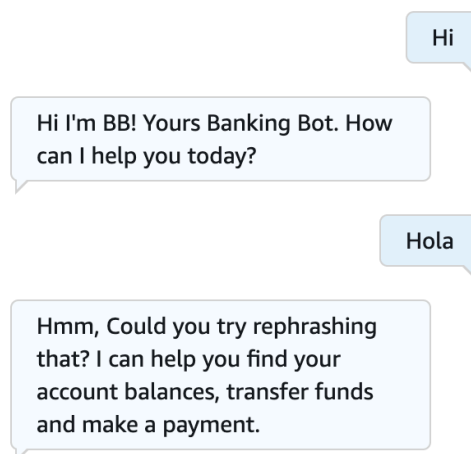
→ I didn't quite get that. Please let me know if you want to know your account balances, or to transfer funds and make a payment.

#### ▼ Variations - optional

Hmm, Could you try rephrasing that? I can help you find your account balances, transfer funds and make a payi

I didn't quite get that. Please let me know if you want to know your account balances, or to transfer funds and r

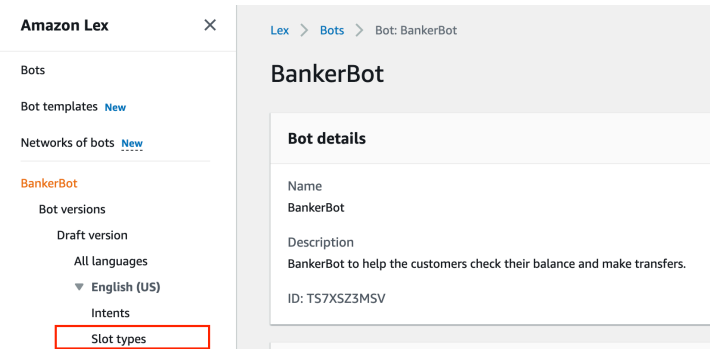
Saved the intent and Tested it.



## Building Chatbot with Custom Slots

After successful creation of BankerBot, now it's time for letting the customers know their account balance with their date of birth as verification.

### Slot Types



☐ **Slots** are pieces of information that a chatbot needs to complete a user's request.

For example, if the intent is to book a table at a restaurant, the chatbot needs specific details like: restaurant name, date, time, number of people. Amazon Lex provides many ready-to-use slot types for common information, like dates and times, but we can also create our own custom slot types to fit our specific needs!

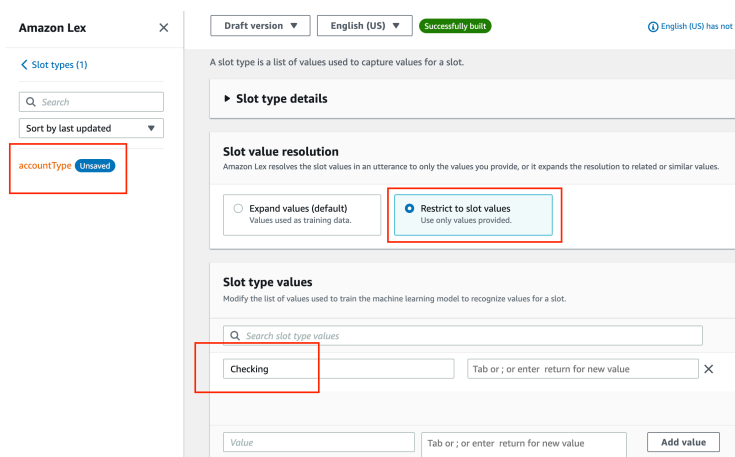
After Choosing **Add slot type**, From the dropdown, chosen **Add blank slot type**.

Then Entered `accountType` for the **Slot type name**. Finally, Chosen **Add** which will bring up a large **Slot types** editor panel. In the **Slot value resolution** panel, chosen **Restrict to slot values**.

☐ Selecting **Restrict to slot values** makes sure that only the values that you specify will count as a valid `accountType`! Otherwise, Amazon Lex will use machine learning to accept other values that it sees users constantly entering.

Different use cases will require different settings, but our BankerBot will only offer customers 3 types of accounts – we don't want Amazon Lex to recognize any alternatives.

In `accountType`, **Slot value resolution** was **restricted to slot values**. and **Value** was given as **Checking**.



Added few more Slot Type Values and their synonyms. And then saved the slot type.

**Slot type values**  
Modify the list of values used to train the machine learning model to recognize values for a slot.

Q Search slot type values

Checking	Tab or ; or enter return for new value X
Savings	Tab or ; or enter return for new value X
Credit	Tab or ; or enter return for new value X
	credit card X visa X mastercard X
	amex X american express X

Value Tab or ; or enter return for new value Add value

Maximum 140 characters. Valid characters: A-Z, a-z, 0-9, @, #, \$

☐ Use slot values as custom vocabulary [Info](#)

Created a new intent CheckBalance

**Add empty intent** X

Create a custom intent for your bot.

Intent name

CheckBalance

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Cancel Add

▼ Intent details [Info](#)

Intent name

CheckBalance

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Intent and utterance generation description

Describe the purpose of your intent. This will also be used when generating utterances for your intent.

Intent to check the balance in specified account type.

Maximum 200 characters.

Provided few utterances, the customer may ask to check their balances.

Sample utterances (7) Info

What's this? [Generate utterances](#)

Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

To generate utterances, you must have permissions to Amazon Bedrock. Amazon Lex will make calls to Amazon Bedrock. Additional charges may be incurred based on the usage of Amazon Bedrock. [Learn more](#)

Q Filter

Sort by added (ascending)

Preview

Plain text

1	What's the balance in my account?
2	Check my account balance.
3	What's the balance in my {accountType} account?
4	How much do I have in my {accountType} ?
5	I want to check the balance.
6	Can you help me with account balance?
7	Balance in {accountType}
8	

Created a slot, accountType for different account types like checking, savings, etc.

Add slot

A slot is used to capture information from the user to fulfill the intent.

☒ Required for this intent

The bot will prompt for this slot during the conversation if a value is not provided by the user.

Name

Slot type

accountType

accountType

Prompts

For which account would you like your balance?

Cancel

Add

▼ Slots (1) - optional Info

Add slot

Information that a bot needs to fulfill the intent. The bot prompts for slots required for intent fulfillment, in priority order below.

Q Filter

Prompt for slot: accountType

Message: For which account would you like your balan...

Slot type

accountType

Created another slot, dateOfBirth for Verifying the customer.

Add slot

A slot is used to capture information from the user to fulfill the intent.

☒ Required for this intent

The bot will prompt for this slot during the conversation if a value is not provided by the user.

Name

Slot type

dateOfBirth

AMAZON.Date

Prompts

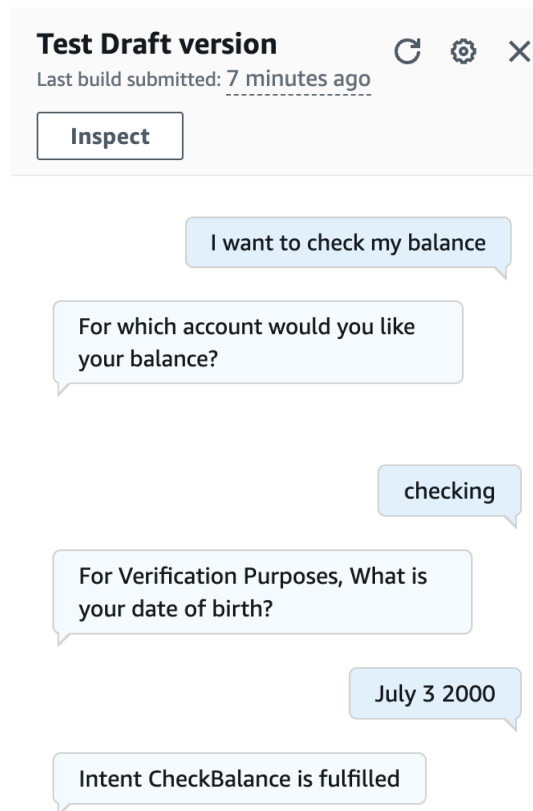
For Verification Purposes, What is your date of birth?

Cancel

Add



After building and tested the BankerBot with the Utterances and it worked.



## Connecting Chatbot with Lambda

AWS Lambda is a service that lets you run code without provisioning or managing servers. Lambda runs the code only when needed and scales automatically, from a few requests per day to thousands per second – all we need to do is supply our code in one of the languages that Lambda supports.

The screenshot shows the AWS Lambda 'Create function' page. The 'Author from scratch' option is selected. The function name is 'BankingBotEnglish' and the runtime is 'Python 3.12'.

**Create function** [Info](#)

Choose one of the following options to create your function.

- ☒ **Author from scratch**  
Start with a simple Hello World example.
- ☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.
- ☐ **Container image**  
Select a container image to deploy for your function.

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  
☒ **x86\_64**  
☐ arm64

After creating the lambda function, under the code source, given the below code and deployed it to give answers.

```
import json
```

```

import random

import decimal

def random_num():

    return(decimal.Decimal(random.randrange(1000, 50000))/100)

def get_slots(intent_request):

    return intent_request['sessionState']['intent']['slots']

def get_slot(intent_request, slotName):

    slots = get_slots(intent_request)

    if slots is not None and slotName in slots and slots[slotName] is not None:

        return slots[slotName]['value']['interpretedValue']

    else:

        return None

def get_session_attributes(intent_request):

    sessionState = intent_request['sessionState']

    if 'sessionAttributes' in sessionState:

        return sessionState['sessionAttributes']

    return {}

def elicit_intent(intent_request, session_attributes, message):

    return {

        'sessionState': {

            'dialogAction': {

                'type': 'ElicitIntent'

            },

            'sessionAttributes': session_attributes

        },

        'messages': [ message ] if message != None else None,

        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes' in intent_request else None

    }

def close(intent_request, session_attributes, fulfillment_state, message):

    intent_request['sessionState']['intent']['state'] = fulfillment_state

    return {

        'sessionState': {

            'sessionAttributes': session_attributes,

            'dialogAction': {

                'type': 'Close'

            },

            'intent': intent_request['sessionState']['intent']

        },

        'messages': [message],

        'sessionId': intent_request['sessionId'],

        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes' in intent_request else None

    }

def CheckBalance(intent_request):

```

```

        session_attributes = get_session_attributes(intent_request)

        slots = get_slots(intent_request)

        account = get_slot(intent_request, 'accountType')

        #The account balance in this case is a random number

        #Here is where you could query a system to get this information

        balance = str(random_num())

        text = "Thank you. The balance on your "+account+" account is $" +balance+" dollars."

        message = {

            'contentType': 'PlainText',

            'content': text

        }

        fulfillment_state = "Fulfilled"

        return close(intent_request, session_attributes, fulfillment_state, message)
def FollowupCheckBalance(intent_request):

    session_attributes = get_session_attributes(intent_request)

    slots = get_slots(intent_request)

    account = get_slot(intent_request, 'accountType')

    #The account balance in this case is a random number

    #Here is where you could query a system to get this information

    balance = str(random_num())

    text = "Thank you. The balance on your "+account+" account is $" +balance+" dollars."

    message = {

        'contentType': 'PlainText',

        'content': text

    }

    fulfillment_state = "Fulfilled"

    return close(intent_request, session_attributes, fulfillment_state, message)
def dispatch(intent_request):

    intent_name = intent_request['sessionState']['intent']['name']

    response = None

    # Dispatch to your bot's intent handlers

    if intent_name == 'CheckBalance':

        return CheckBalance(intent_request)

    elif intent_name == 'FollowupCheckBalance':

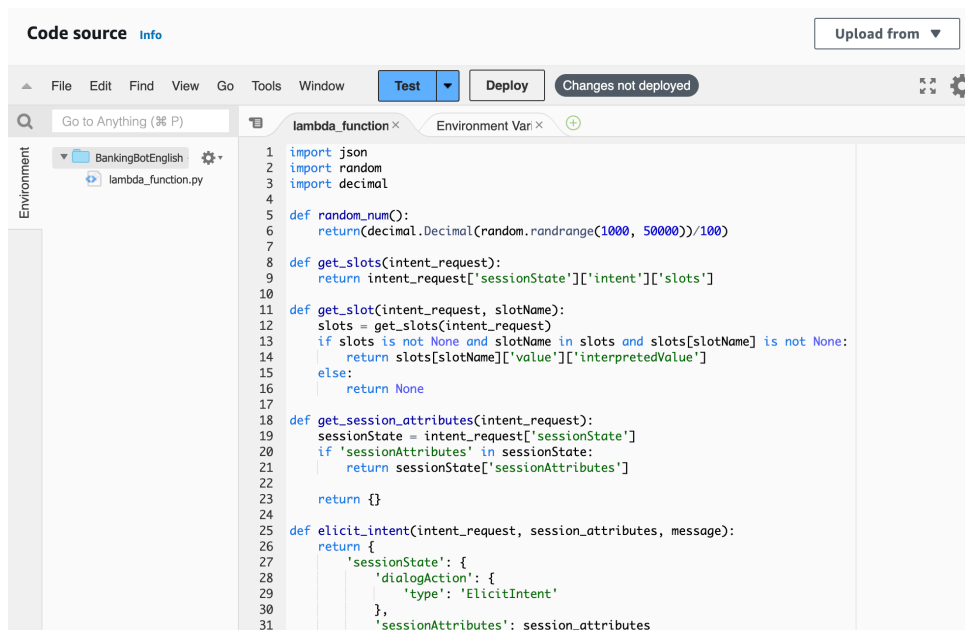
        return FollowupCheckBalance(intent_request)

    raise Exception('Intent with name ' + intent_name + ' not supported')
def lambda_handler(event, context):

    response = dispatch(event)

    return response

```



```
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
23     return {}
24
25 def elicit_intent(intent_request, session_attributes, message):
26     return {
27         'sessionState': {
28             'dialogAction': {
29                 'type': 'ElicitIntent'
30             },
31             'sessionAttributes': session_attributes
```

## Now **Connected AWS Lambda with Amazon Lex**

Under the deployment in Lex, Chosen Aliases

### BankerBot

#### Bot versions

Draft version

All languages

▼ English (US)

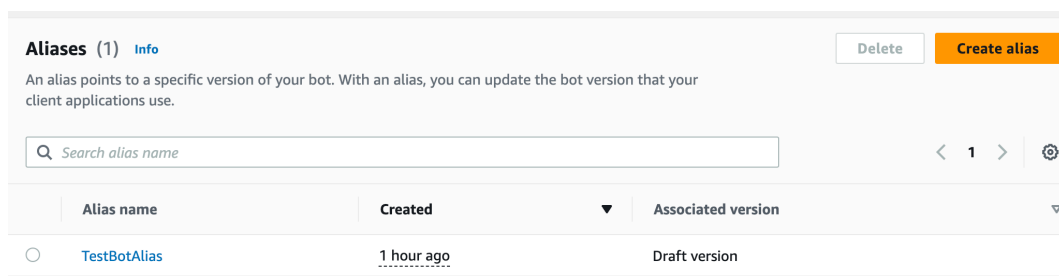
Intents

Slot types

▼ Deployment

Aliases

Channel integrations



Alias name	Created	Associated version
TestBotAlias	1 hour ago	Draft version

- ❑ An **alias in Amazon Lex** acts as a pointer for a specific version of the bot. So when connecting Lex with other AWS services or custom applications, those external resources will connect to an alias, which will point to the specific version of the bot that we want to use. Now, instead of always updating apps to connect to the newest version of the bot, we can just update the alias to point to that new version. All the apps will automatically start using the updated bot without needing any changes on our end – this saves developers a TON of time and reduces the risk of errors!

Now, Chosen TestBotAlias

- ☐ TestBotAlias is a default version of your bot that's made for testing or development. This is the playground version of your bot that you'll use to make sure everything works smoothly before rolling out changes!

Under Languages chosen English, and selected BankingBotEnglish as Lambda Function Source.

- ☐ Using the **\$LATEST** version means we're directing our alias to always use the most up to date version of this Lambda function. This setup is a time saver that lets us immediately test any changes in our function.

**Languages (1)** [Info](#) Test Manage languages in alias

Select the languages you want to enable in the alias. Only languages that are built can be enabled.

< 1 > [Settings](#)

Language	Status
<input checked="" type="radio"/> English (US)	Successfully built

### Alias language support: English (US)

#### ▼ Lambda function - optional

The Lambda function is invoked for initialization, validation, and fulfillment.

Source

BankingBotEnglish

Lambda function version or alias

\$LATEST

[Learn more about Lambda](#) [🔗](#)

The Lambda function is now ready to work on the BankingBot intents, but we still have to tell Amazon Lex *which* intent will actually use the Lambda function.

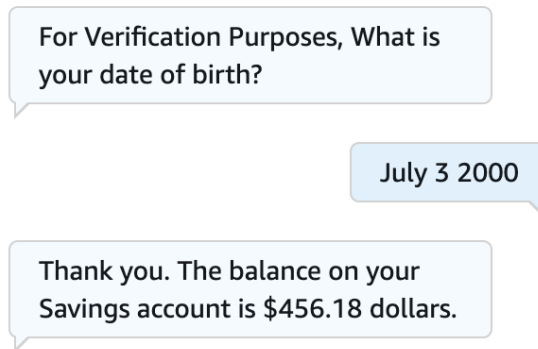
## Connecting CheckBalance intent with Lambda function

Under the CheckBalance intent, in the Fulfilment Panel, on Expansion the **On successful fulfilment** bubble. → Chosen **Advanced options** And under the **Fulfilment Lambda code hook** panel, checked the checkbox next to **Use a Lambda function for fulfilment**.

- ☐ In **Amazon Lex**, **fulfilment** means completing the intent. With our BankingBot, after a user tells our bot: The account they want to check, and Their birthday for verification. The bot has all the information it needs and moves to fulfilment. This is where it will use the Lambda function to get the account balance and pass it back to the user.
- ☐ **Code hooks** help us connect our chatbot to custom Lambda functions for doing specific tasks during a conversation. They're used to handle more complex actions that the basic chatbot setup can't do on its own, like checking data from a database or making decisions based on past conversations.

Essentially, code hooks make our chatbot smarter and more useful by allowing it to perform these extra steps seamlessly during chats.

After saving the Function, built the bot and tested it and now it responds with the account balance.



Now it's time save the user info.

## Saving User Info with the Chatbot

- In your **CheckBalance** intent page, scroll down to the **Contexts** panel.
- Under the **Output contexts** drop-down, choose **New context tag**.

☐ **Context tags** in Amazon Lex are used to store and check for specific information across different parts of a conversation. They help save the user from having to repeat certain information. There are two types of context tags in Amazon Lex:

**Output context tag:** This tells the chatbot to remember certain details after an intent is finished, so other parts of the conversation can use this stored information later. For example, the account type from BalanceCheck could be saved and reused.

**Input context tag:** This checks if specific details are already available before an intent activates. For example, FollowupCheckBalance will check if this conversation already has the user's date of birth saved somewhere, so it won't need to ask for that information again.

**Add new context tag** ×

Context tag name

Expires after

turns, or  seconds

## Creating the FollowupCheckBalance intent

×

**Add empty intent**  
Create a custom intent for your bot.

Intent name

FollowupCheckBalance

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Cancel

Add

▼ Intent details [Info](#)

Intent name

FollowupCheckBalance

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Intent and utterance generation description

Describe the purpose of your intent. This will also be used when generating utterances for your intent.

Intent to allow a follow-up balance check request without authentication.

Maximum 200 characters.

Added Input Context and Utterances for the Intent.

▼ Contexts - optional [Info](#)

Input contexts

Choose contexts

contextCheckBalance

×

How about my {accountType} account?

what about {accountType} ?

and in {accountType} ?

Added 2 slots – **accountType** ; **dateOfBirth** ;

×

**Add slot**  
A slot is used to capture information from the user to fulfill the intent.

☒ Required for this intent  
The bot will prompt for this slot during the conversation if a value is not provided by the user.

Name

accountType

Slot type

accountType

▼

Prompts

For which account would you like your balance?

Cancel

Add

▼ Slots (2) - optional

Info

Information that a bot needs to fulfill the intent. The bot prompts for slots required for intent fulfillment, in priority order below.

Q Filter

⋮

▶ Prompt for slot: accountType

Message: For which account would you like your balan...

Slot type

accountType

×

⋮

▶ Prompt for slot: dateOfBirth

Message: For Verification Purposes, what is your date ...

⚡ Slot type

AMAZON.Date

×

- In **FollowupCheckBalance** intent page, expanded the **dateOfBirth** slot.
- In **Advanced options** -> Under **Default values** panel (This panel lets us create default values for the intent's slots) Entered *#contextCheckBalance.dateOfBirth* , added default value and updated the slot.

This tells Amazon Lex that the input context `contextCheckBalance` should have the value of `dateOfBirth` in `CheckBalance`.

Name

dateOfBirth

Prompts

For Verification Purp

You can use the advanced

Advanced options

▼ Default values - *optional*

#contextCheckBalance.dateOfBirth

In the Fulfillment section, Advanced Options, Checked the checkbox of **Fulfillment Lambda code hook**

Now the BankerBot, can answer the account balance for multiple account types by verifying



only once.

## Creating the new TransferFunds intent

▼ Intent details [Info](#)

Intent name

TransferFunds

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, \_

Intent and utterance generation description

Describe the purpose of your intent. This will also be used when generating utterances for your intent.

Intent to help user transfer funds between bank accounts

Maximum 200 characters.

### Sample Utterances

- Can I make a transfer?
- I want to transfer funds
- I would like to transfer {transferAmount} from {sourceAccountType} to {targetAccountType}
- Can I transfer {transferAmount} to {targetAccountType} ?
- Please help me with a transfer
- Need to make a transfer

Added 3 Slots - **sourceAccountType ; targetAccountType ; transferAmount**

▼ Slots (3) - optional [Info](#)

Information that a bot needs to fulfill the intent. The bot prompts for slots required for intent fulfillment, in priority order below.

Add slot

Q Filter

▶ Prompt for slot: sourceAccountType

Message: Which account would you like to transfer fro...

Slot type

accountType

×

▶ Prompt for slot: targetAccountType

Message: Which account are you transferring to?

Slot type

accountType

×

▶ Prompt for slot: transferAmount

Message: How much money would you like to transfer?

Slot type

AMAZON.Number

×

- ☐ Now setting up Confirmation Prompts - **Confirmation prompts** typically repeat back information for the user to confirm. e.g. "Are you sure you want to do x?" If the user confirms the intent, the bot fulfills the intent. If the user declines, then the bot responds with a decline response that you set up.

**Confirmation**
[Info](#)

☒ Active

Prompts help to clarify whether the user wants to fulfill the intent or cancel it.

**Prompts to confirm the intent**

**Responses sent when the user declines the intent**

Message: Got it. So we are transferring {transferAmount} from {sourceAccountType} to {targetAccountType}. Can I go ahead

Message: The transfer has been cancelled.

**Confirmation prompt**  
What will the bot say to prompt the user to confirm this intent.

Got it. So we are transferring {transferAmount} from {sourceAccountType} to {targetAccountType}. Can I go ahead

**Decline response**  
What will the bot say if the user says NO to the confirmation prompt.

The transfer has been cancelled.

**Advanced options**

Configure confirmation prompts and decline responses.

Added a closing response after the transfer been made.

**Closing response**
[Info](#)

☒ Active

You can define the response when closing the intent.

**Response sent to the user after the intent is fulfilled**

**Message group**
[Info](#)

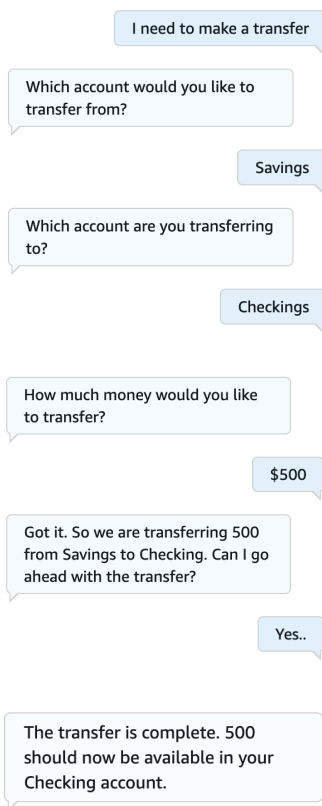
Message: The transfer is complete. {transferAmount} should now be available in your {targetAccountType} account.

You can define a text message group to respond using plain text.

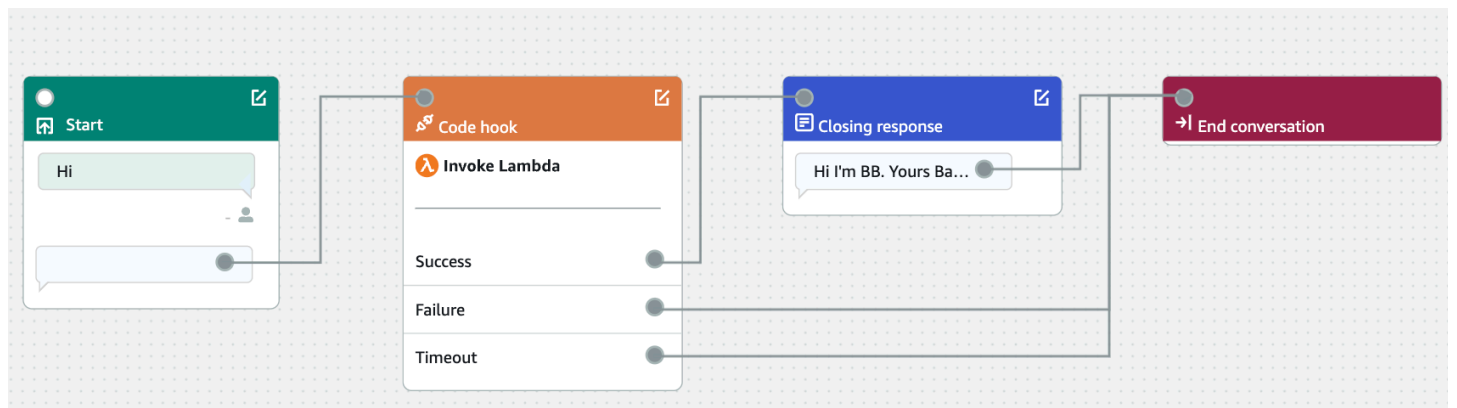
**Message**

The transfer is complete. {transferAmount} should now be available in your {targetAccountType} account.

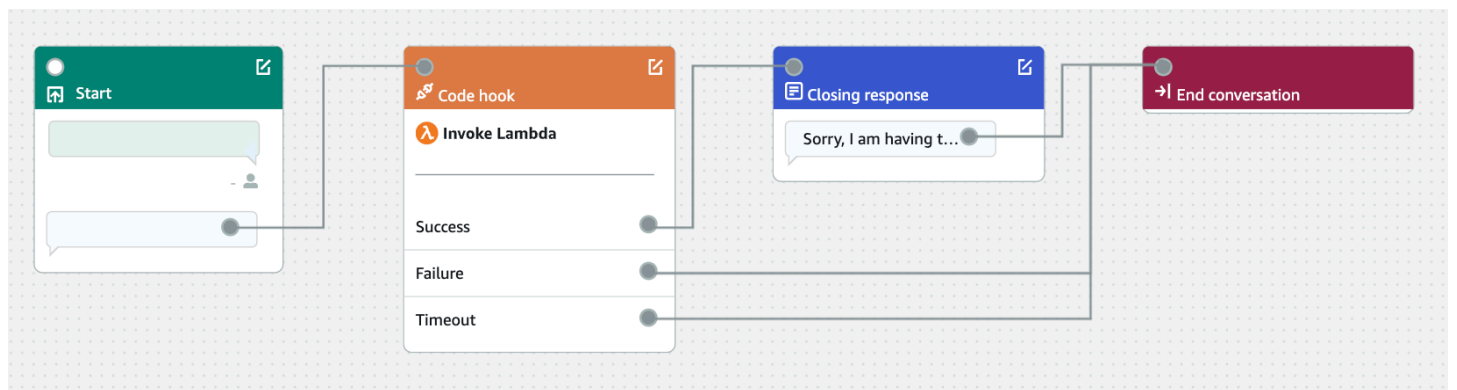
Saved Intent and Built the BankerBot and Tested it for Making Transfers.



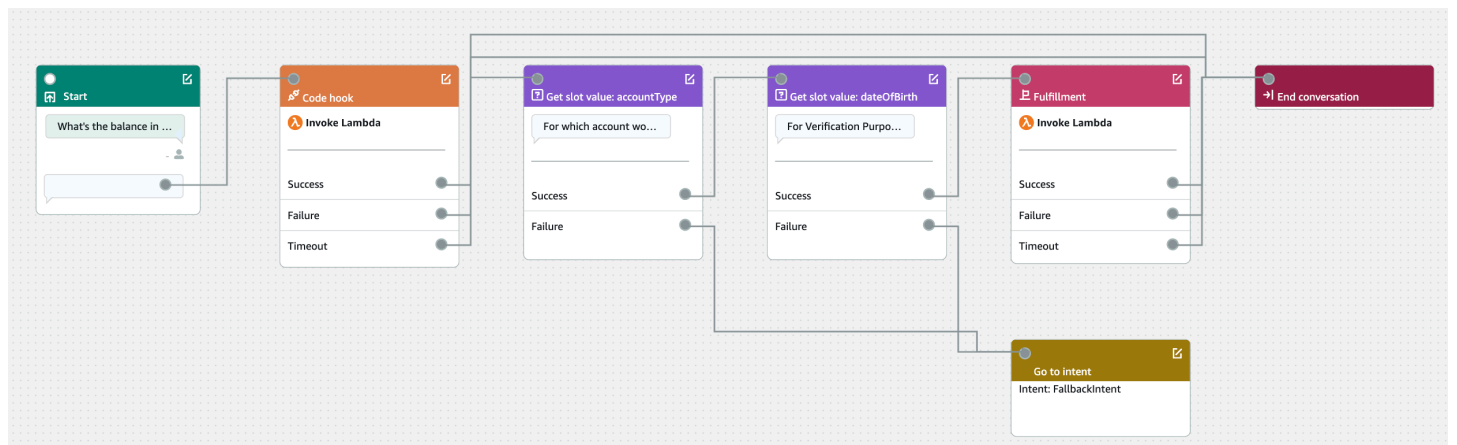
# PROJECT



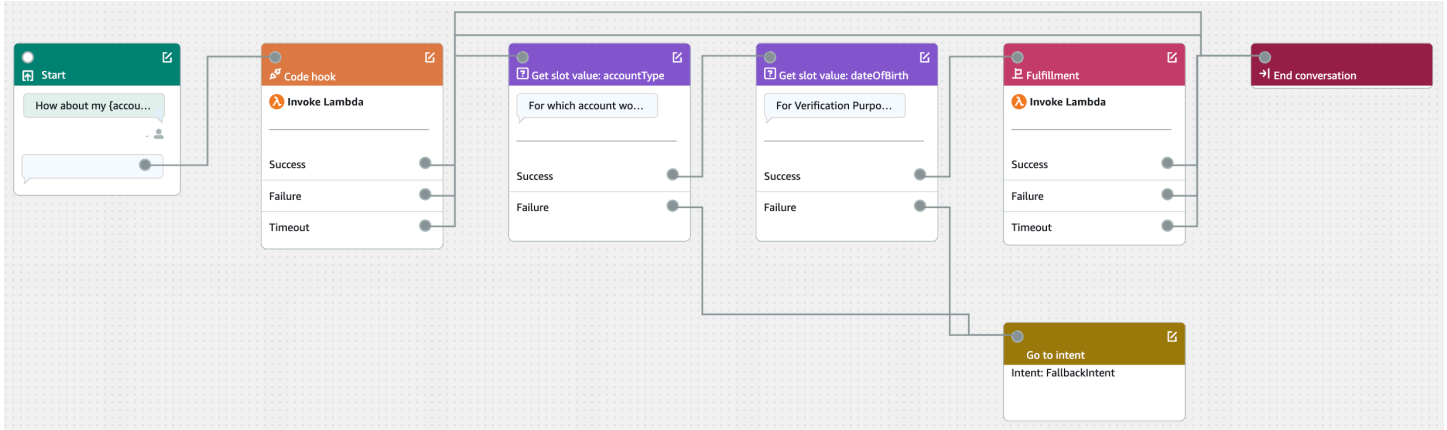
## FALLBACK INTENT



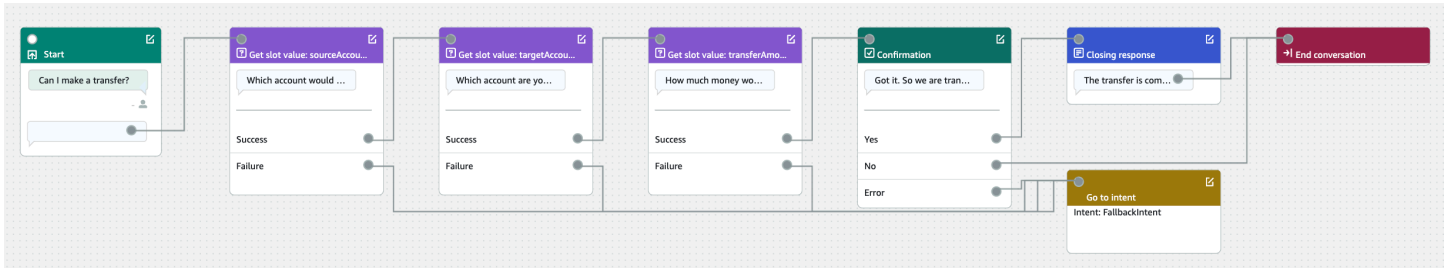
## CHECK BALANCE INTENT



## FOLLOWUP CHECK BALANCE INTENT



## TRANSFER FUNDS INTENT



## Final BankerBot Results

Hi

Hi I'm BB. Yours Banking Bot. How can I help you today?

I want to know my account balance

For which account would you like your balance?

Savings

For Verification Purposes, What is your date of birth?

July 3 2000

Thank you. The balance on your Savings account is \$143.74 dollars.

and in Checkings?

Thank you. The balance on your Checking account is \$297.66 dollars.

Make a Transfer to Savings

Which account would you like to transfer from?

Checkings

How much money would you like to transfer?

\$150

Got it. So we are transferring 150 from Checking to Savings. Can I go ahead with the transfer?

Yes

The transfer is complete. 150 should now be available in your Savings account.

Thank you