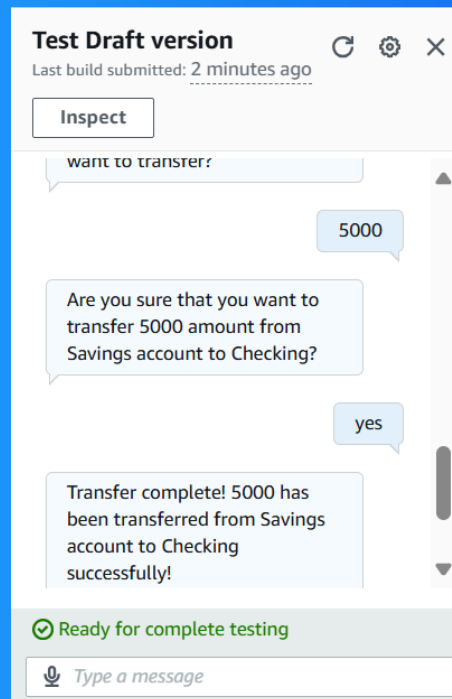


# Build a Chatbot with Multiple Slots



Unmilan Mukherjee



# Introducing Today's Project!

## What is Amazon Lex?

AWS Lex is a service that allows us to create a chatbot using robust configurations. Lex is particularly useful as it can be connected to any other AWS service very seamlessly and easily to enhance their experience.

## How I used Amazon Lex in this project

I used Amazon Lex to create an intent to transfer funds between two accounts, use the visual interface to create a bot and also use CloudFormation to automate the setup of a AWS Lex service.

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

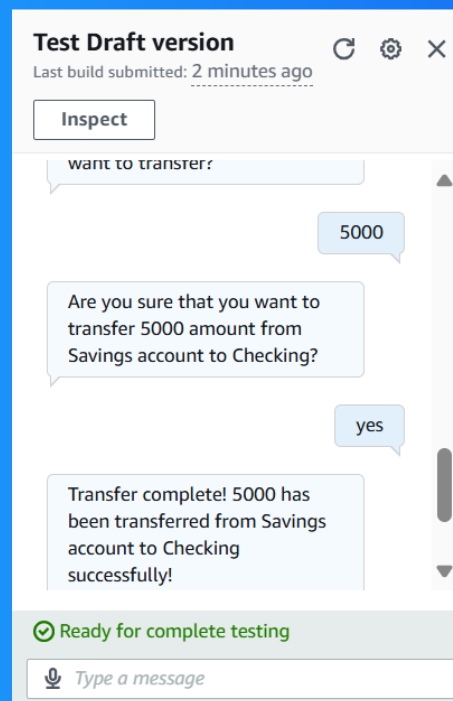
I did not expect how easy it would be to automate the creation of a Amazon Lex bot by using AWS CloudFormation and a simple yaml file with all the configurations mentioned within it.

## This project took me...

This project took me roughly 80 minutes

# TransferFunds

An intent I created for my chatbot was TransferFunds, which allows the user to transfer some amount from one account to another.



# Using multiple slots

For this intent, I had to use the same slot type twice. This is because we wanted two account types which lie within the same: savings, current or credit card types. Hence we can use the same slot type twice with no issues.

I also learnt how to create confirmation prompts, which are basically confirmation messages that the chatbot will ask the user before fulfilling an intent. If the user confirms then it goes ahead, otherwise it will cancel the intent fulfillment.

**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: Are you sure that you want to transfer {tran...	Message: Transfer cancelled.

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

Are you sure that you want to transfer {transferAmount} amount from {sourceAccountType} account to {targetAccc

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

Transfer cancelled.

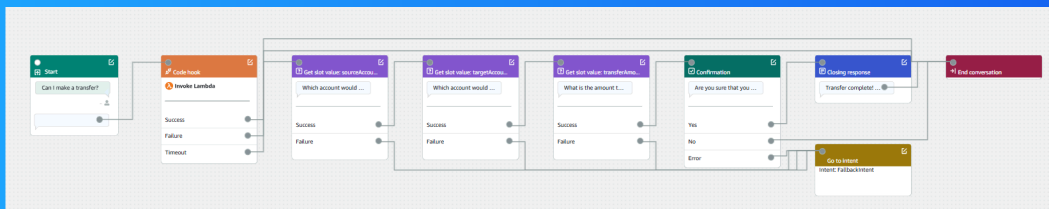
Advanced options

Configure confirmation prompts and decline responses.

# Exploring Lex features

Lex also has a special conversation flow feature that allows us to have a graphical look at every single logical flow of conversation that our bot could go into. It can show us everything that our bot can say. We can edit directly as well from here!

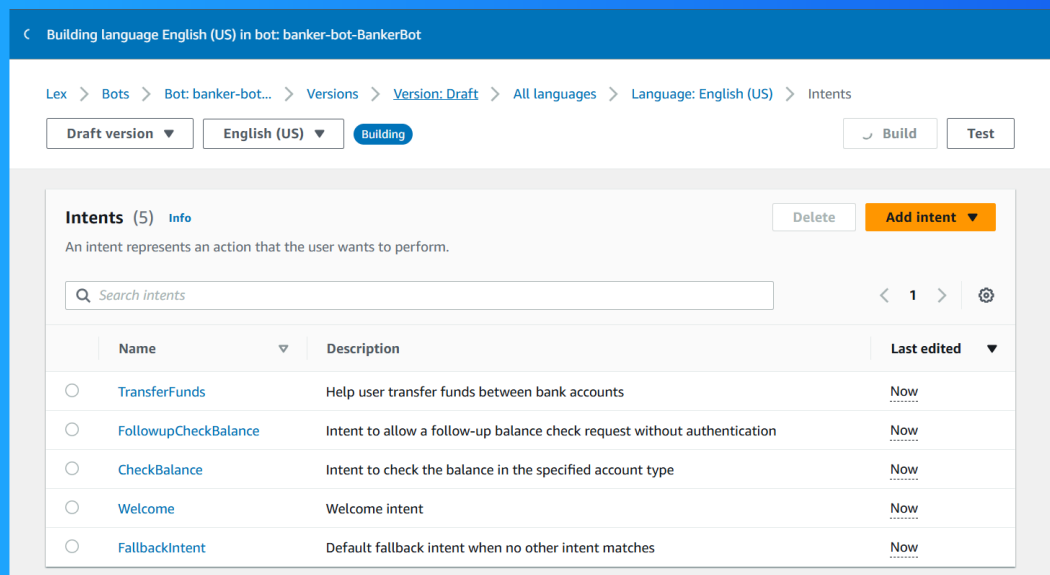
You could also set up your intent using a visual builder! A visual builder transforms our set up page into a visual chart. We can use this view to edit our chatbot's intent instead of the normal editor as well.



# AWS CloudFormation

AWS CloudFormation is a service that allows us to create preset templates that we can use to very easily setup certain AWS services with the exact settings we want. It makes the setting up of common services with presets very easy and automated.

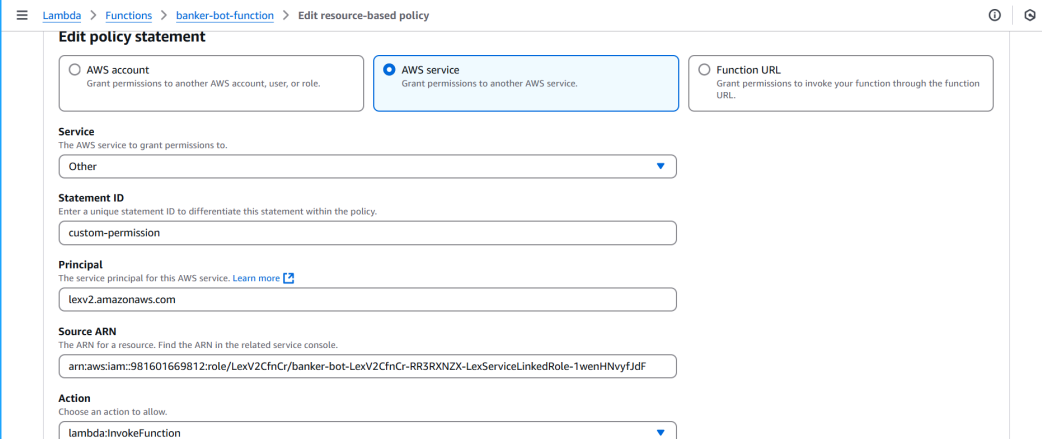
I used CloudFormation to automate the creation of my banking-bot AWS Lex Service.



# The final result!

Re-building my bot with CloudFormation took me just 5 minutes with the help of the yaml template.

There was an error after I deployed my bot! The error was "denied access". I fixed this by creating a new Lambda function and linking it to my banker-bot.



The screenshot shows the 'Edit resource-based policy' page in the AWS IAM console. The breadcrumb navigation at the top reads: Lambda > Functions > banker-bot-function > Edit resource-based policy. The page title is 'Edit policy statement'. There are three radio buttons for selecting the grant type: 'AWS account' (unselected), 'AWS service' (selected), and 'Function URL' (unselected). Below these, the 'Service' dropdown is set to 'Other'. The 'Statement ID' field contains 'custom-permission'. The 'Principal' field contains 'lexv2.amazonaws.com'. The 'Source ARN' field contains 'arn:aws:iam:981601669812:role/LexV2CfnCr/banker-bot-LexV2CfnCr-RR3RXNZX-LexServiceLinkedRole-1wenHNvyfJdf'. The 'Action' dropdown is set to 'lambda:InvokeFunction'.

**Edit policy statement**

☐ AWS account  
Grant permissions to another AWS account, user, or role.

☒ AWS service  
Grant permissions to another AWS service.

☐ Function URL  
Grant permissions to invoke your function through the function URL.

**Service**  
The AWS service to grant permissions to.

Other

**Statement ID**  
Enter a unique statement ID to differentiate this statement within the policy.

custom-permission

**Principal**  
The service principal for this AWS service. [Learn more](#)

lexv2.amazonaws.com

**Source ARN**  
The ARN for a resource. Find the ARN in the related service console.

arn:aws:iam:981601669812:role/LexV2CfnCr/banker-bot-LexV2CfnCr-RR3RXNZX-LexServiceLinkedRole-1wenHNvyfJdf

**Action**  
Choose an action to allow.

lambda:InvokeFunction

# Using the visual builder

Using the visual builder, I added the ability to check for an account balance after transferring funds between accounts. What this means for an end user is that they can confirm the transfer of funds manually if they so choose to do.

The new Get slot value will trigger when the transfer of money between two accounts is successful. This card will try to capture the user's intent to get the account balance of an account.

