7/20/2023

Srinivasa Kandala

Zelis

**AWS lex chatbot interacting with Multiple databases**

**Introduction to AWS Lex Bot:**

****

**What is AWS Lex?**

AWS Lex is a powerful service provided by Amazon Web Services (AWS) that allows developers to build conversational interfaces,

commonly known as chatbots or virtual assistants. It utilizes natural language understanding (NLU) and automatic speech recognition (ASR)

to enable users to interact with applications and services in a natural and intuitive way.

Amazon Web Services (AWS) is a leading cloud computing platform offering a comprehensive suite of services, including computing, storage (Amazon S3, Amazon EBS), databases (Amazon RDS, Amazon DynamoDB), AI/ML (Amazon Lex, Amazon Polly), and analytics (Amazon Redshift, Amazon Quick Sight).

In our lex bot we have used few AWS services like Amazon lex chatbot for a retrieving a financial data from a file S3 file along with Lambda code and deployed the chatbot with CloudFormation and amplify. Also, used Twilio for converting the bot into a WhatsApp chat.

We have also generated a lex bot using API Gateway service along lambda code by using a dummy URL for extracting required data from the bot.

**Key Features of AWS Lex:**

* **Natural Language Understanding**: AWS Lex uses advanced machine learning algorithms to comprehend user inputs and extract intent, entities, and context from the text or voice input. This allows the bot to understand user requests and provide appropriate responses.
* **Automatic Speech Recognition:** Lex supports voice interactions, enabling users to communicate with the bot using their voice. Lex's automatic speech recognition capabilities convert voice input into text, making it easier to process and respond.
* **Built-in Integration with Other AWS Services:** Lex seamlessly integrates with various AWS services like AWS Lambda, AWS DynamoDB, AWS Polly, and more. This integration allows developers to create complex and dynamic conversational experiences.
* **Contextual Conversations:** Lex maintains session state, allowing users to have contextual conversations. Contextual understanding enables more natural and interactive conversations, as Lex remembers previous interactions within a session.
* **Customizable and Extensible:** Developers can create custom intents, slot types, and fulfillment logic to tailor the bot to specific use cases. Lex provides flexibility to adapt the bot's behavior according to business requirements.
* **Scalability and Availability:** As an AWS service, Lex offers high availability and scalability, ensuring that your bot can handle varying levels of user interactions and maintain responsiveness.
* **Built-in Prompts and Confirmation:** Lex supports prompts and confirmations, which helps guide users and seek clarifications to ensure accurate interactions.

**Conclusion**

AWS Lex empowers developers to create sophisticated chatbots with advanced natural language understanding capabilities. By leveraging AWS's robust infrastructure and integration with other AWS services, Lex bots can be customized to fulfill a wide range of business requirements. Whether for customer support, interactive user interfaces, or process automation, AWS Lex is a versatile tool for building conversational interfaces that deliver seamless and engaging user experiences.

**Multiple Use Case Scenarios:**

**Scenario 1:AWS lex Chatbot retrieving the information from excel file using lambda function & S3 bucket:**

Configure intent utterances to provide data for the bot's responses, enabling seamless user interactions. Also, add the slot types in the utterances.

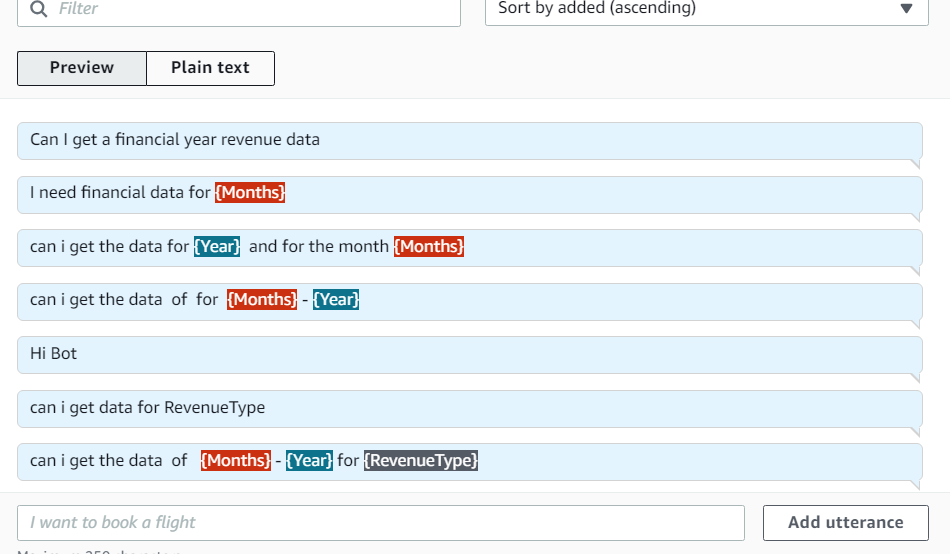
Before testing the bot deploy the bot in the intent along with Lambda code reading a S3 file.

You can deploy the bot on your own applications or messaging platforms such as Facebook, WhatsApp, Messenger or Slack or through other services in AWS.

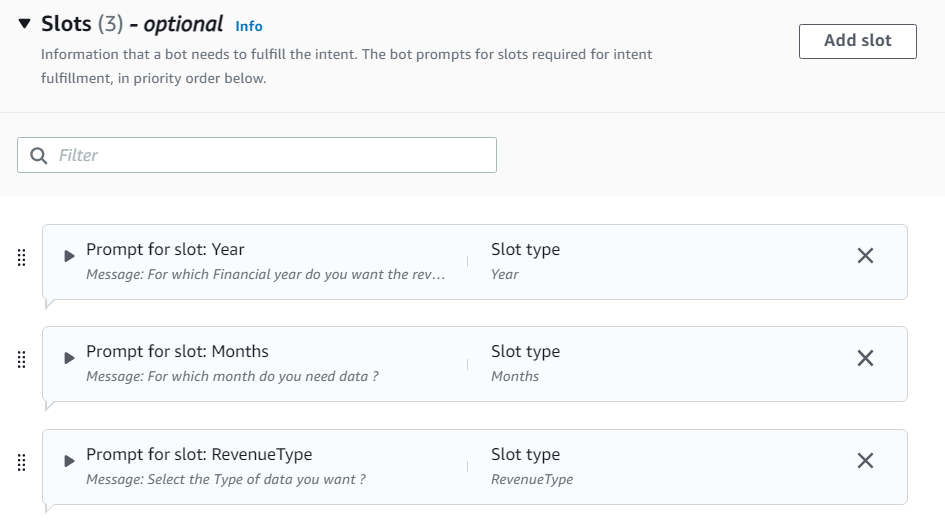
**Bot** – A bot performs automated tasks from FinancialData for getting required data responses such as any month, year and RevenueType form the S3 file.

**Intent name** – A descriptive name for the intent is[FinancialData](https://ap-southeast-1.console.aws.amazon.com/lexv2/home?region=ap-southeast-1#bot/DHWSMI5OEY/version/DRAFT/locale/en_US/intent/YLRRONN8MU).

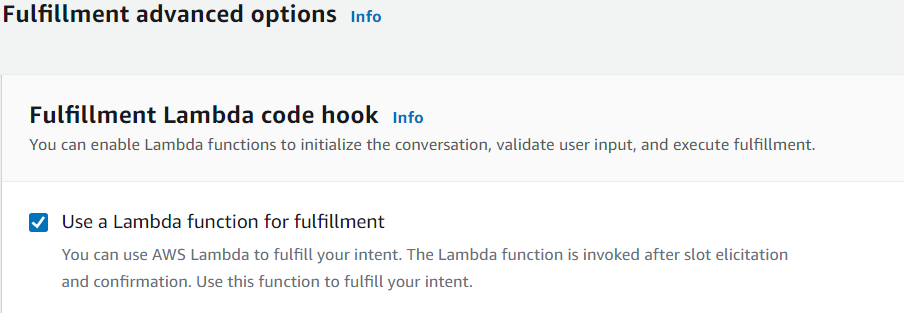
**Sample utterances** – How a user might convey the intent. For example, a user might say " Can I get a financial year revenue data" or " can i get the data of for {Months} - {Year}"



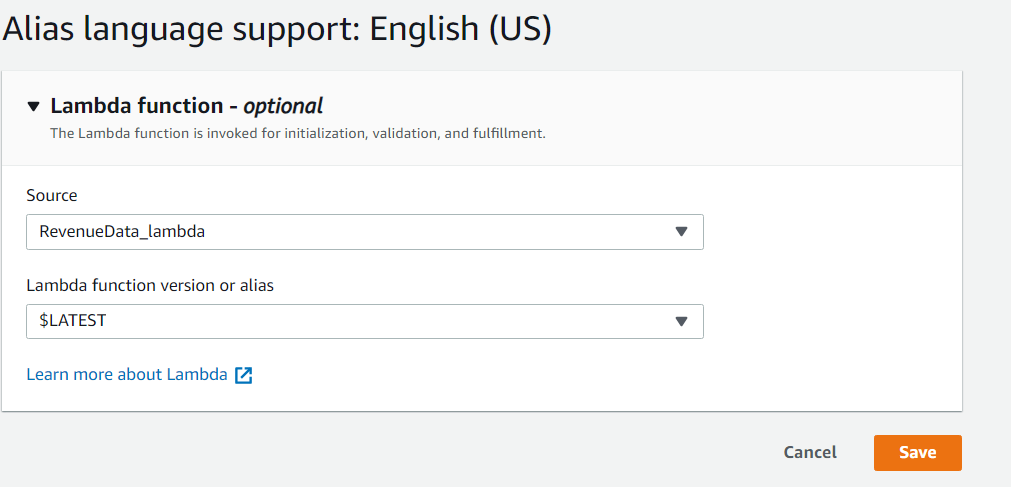
**Slot type** – Each slot has a type. You can create your own slot type, or you can use built-in slot types. For example, you might create and use the following slot types for the FinancialData intent



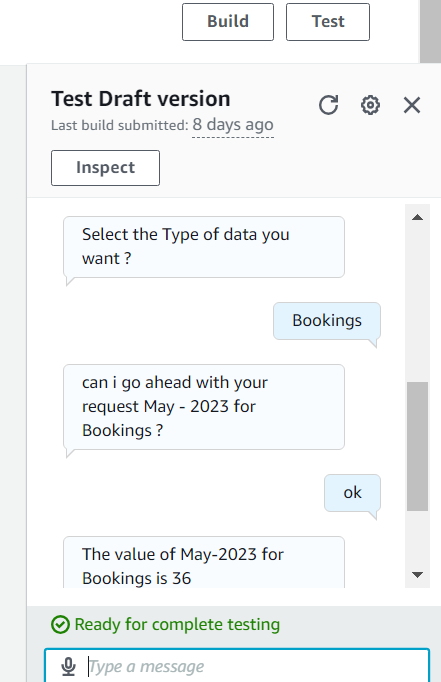
**How to fulfill the intent** – To fulfill the intent after the user provides the necessary information. We created a Lambda function triggering S3 file in the code to fulfill the intent.



**Alias** – An alias is a pointer to a specific version of a bot. After creation of bot we need to deploy the bot using aliases option. With an alias, you can update the version your client applications are using.



**Build and Test**: Save the intent and build the Lexbot. Utilize the Test feature to verify the bot's functionality.



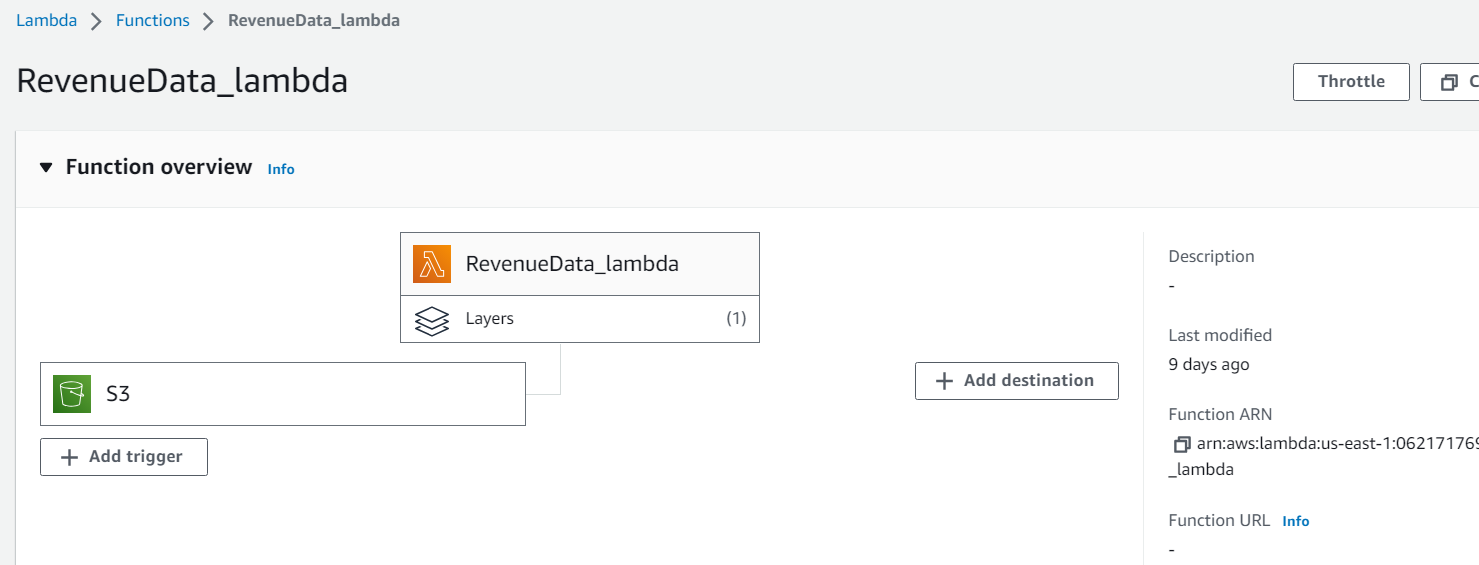
# Creating S3 bucket and uploading a excel file:

Create an bucket from S3 services , name it as **aws-test-1234** and upload an excel file to the s3 bucket where we want retrieve the information to the bot. For our case file is **revenuedata.xlsx**

**Lambda Code:**

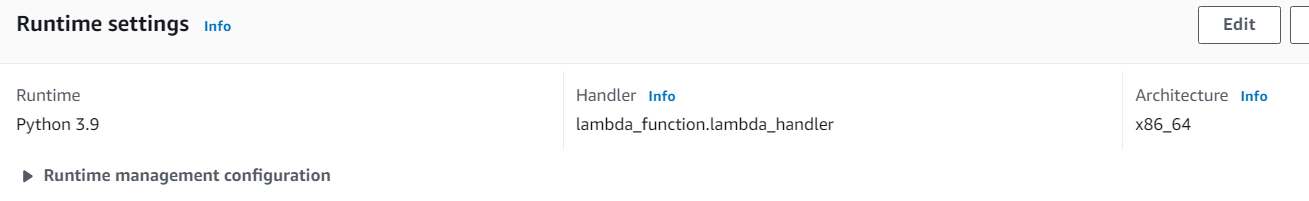
This code is for a chatbot built with AWS Lex, which can retrieve revenue data based on user inputs like year, month, and revenue type. The chatbot uses a Lambda function to process the user's request and fetch the data from an Excel file stored in an S3 bucket.

Add a trigger for the S3 bucket having uploaded excel file to the lambda function and add a layer (“**AWSSDKPandas-Python39**”)



Paste the lambda code from GitHub account in the code section of the lambda Function deploy the code and create test event for testing the functionality.

Under Runtime settings handler should be like lambda\_function.lambda\_handler

****

# Deploy a Web UI for Chatbot

1. After built a financial chatbot using Amazon Lex. Tested it using the Amazon Lex console. Now the bot is ready to deploy it website.
2. We have different ways to deploy the bot. In this Case, we have chosen CloudFormation service for deploying the bot into a website.
3. Press the Launch Stack button for the Region in which will area use your chatbot.
4. The FinancialData bot, CloudFormation and Amplify must be in the same Region while deploying the Bot.
5. In the CloudFormation create a stack with standard all new resources and add the S3 URL in this and accept the parameters as per requirements.
6. We opted for CloudFormation by creating a Stack named **lex-web-ui** creation (status: CREATE\_COMPLETE) then proceed to the Outputs tab in CloudFormation.
7. In the Outputs, locate and choose either WebAppUrl, ParentPageUrlor SnippetUrl and located all the files into a GitHub repository for easy access.

A screenshot of a computer

Description automatically generated

1. For all the URL from the outputs of our stack in the CloudFormation we have created a GitHub repository for accessing all the files. Here, is the link of the GitHub:
2. I AWS Amplify with host app to AWS-REAT-APP by GitHub and build into a website for Financial Bot.A screenshot of a computer

   Description automatically generated
3. Also, edit the URL of the react app website into the index.html file parent origin to link with the Financial Bot.
4. After deployment of the bot into website in amplify its ready for accessible to everyone for the interactions.
5. Here is the website of the Financial Data chatbot: [React App (amplifyapp.com)](https://main.d9r62xvz7si7s.amplifyapp.com/)

**Scenario 2: AWS Lex Bot interacting with ThirdParty API’s**

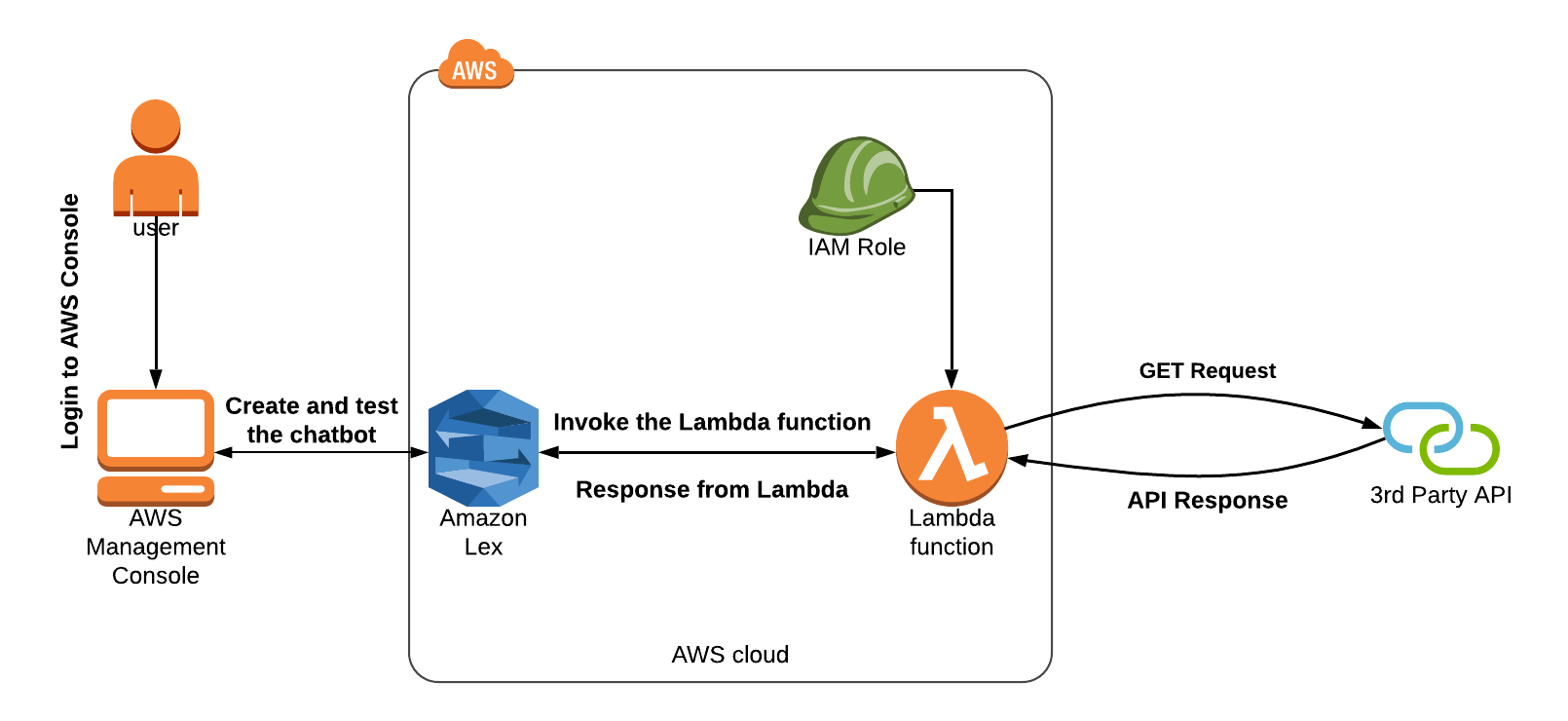
**Overview:**

The implementation described here showcases the development of an AWS Lex chatbot capable of interacting with third-party APIs.

This integration allows the chatbot to access external services and provide dynamic responses to user queries.

**Architectural Diagram:**

The architectural diagram demonstrates the flow of data between the user, AWS Lex, and the third-party API.



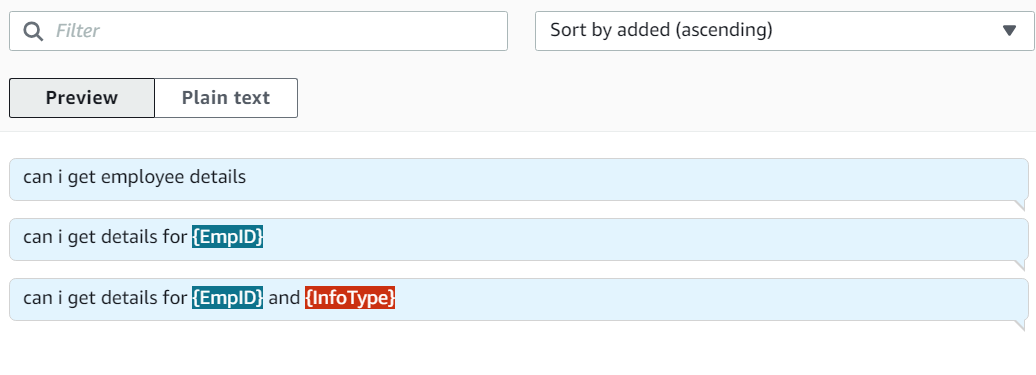
**Creating Lex Chatbot using AWS Console:**

**Bot Creation:** Begin by creating a new Lexbot in the AWS Management Console. Name it as "APIEmployeedata".

**Intents:** Inside the bot, define an intent called "RetrieveAPIInfo" to handle user queries related to employee data retrieval.

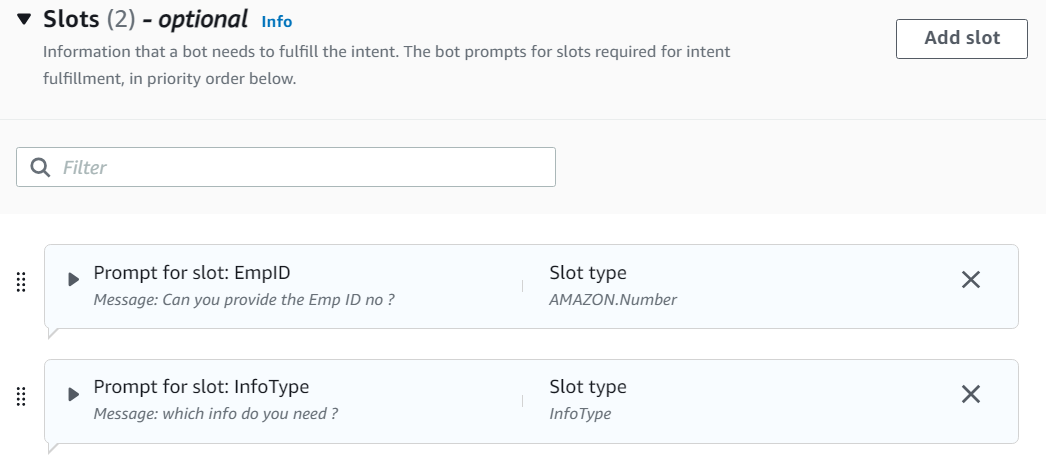
**Utterances:** Add sample utterances to the intent, representing different user queries that can trigger the bot's interaction.

* 1. can i get employee details
  2. can i get details for {EmpID}
  3. can i get details for {EmpID} and {InfoType}

**Slots:** Create slots to capture user inputs like "EmpID" (employee ID) and "InfoType" (custom slot type) to specify the type of employee information.

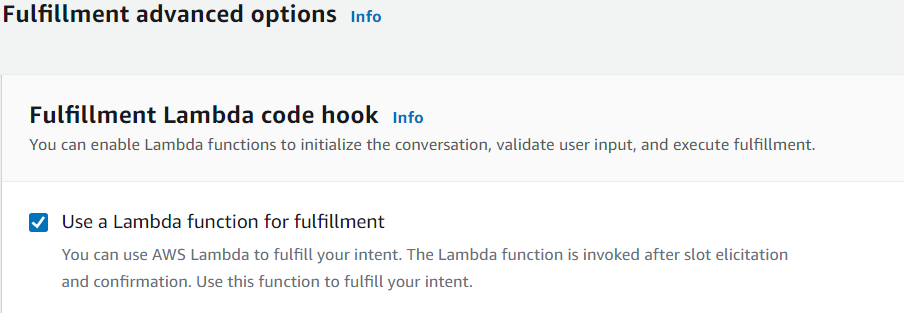
1.EmpID: Amazon.Number

2.InfoType: InfoType(custom slot type)

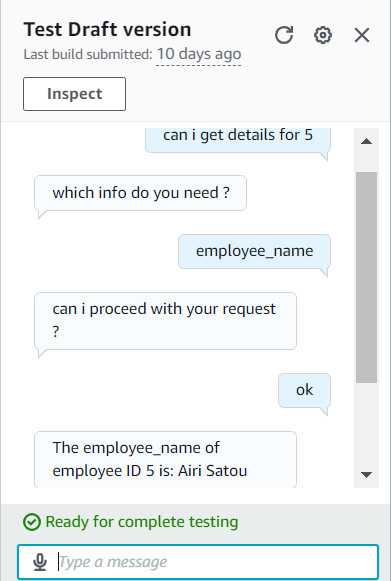


**Confirmation and Fulfillment:**

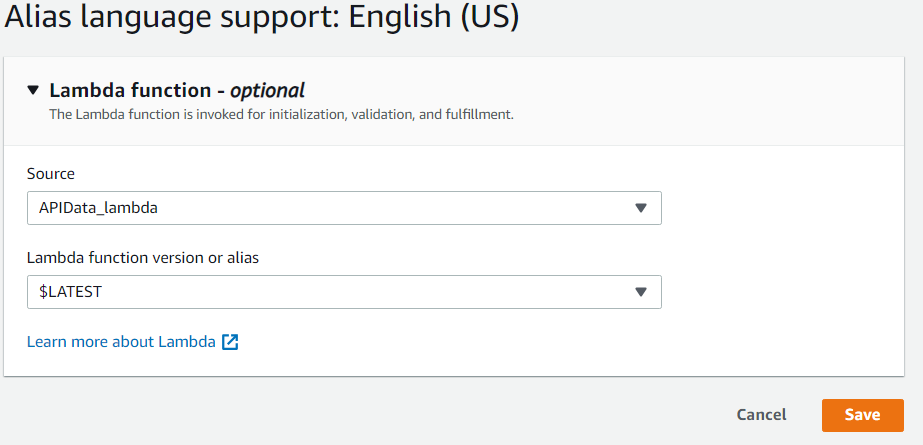
Provide a confirmation message to prompt users before proceeding with their request. Set up fulfillment using an advanced option, "Use a Lambda Function for Fulfillment."



**Build and Test:** Save the intent and build the Lexbot. Utilize the Test feature to verify the bot's functionality.



**Aliases**: Deploy the bot using aliases features and select the lambda code as source and version as LATEST.



**Lambda Function Code:**

The Lambda function code plays a crucial role in interacting with the third-party API and generating responses. The code receives user inputs from Lex (such as employee ID and info type),

makes an HTTP request to the third-party API, processes the API response, and formulates an appropriate response for the Lexbot.

**Pre-Requisites:**

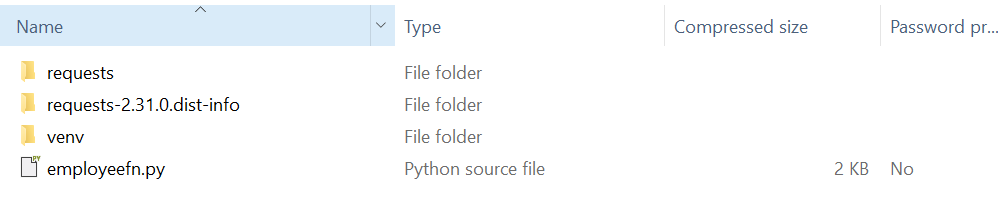
Before deploying the Lambda function, create a deployment package for the code, including dependencies like "requests" and "urllib3". Run the below command **pip install requests urllib3**

Create and activate a virtual environment for managing the dependencies.

Zip all the files in the directory, including the Python script, dependencies, and virtual environment files.

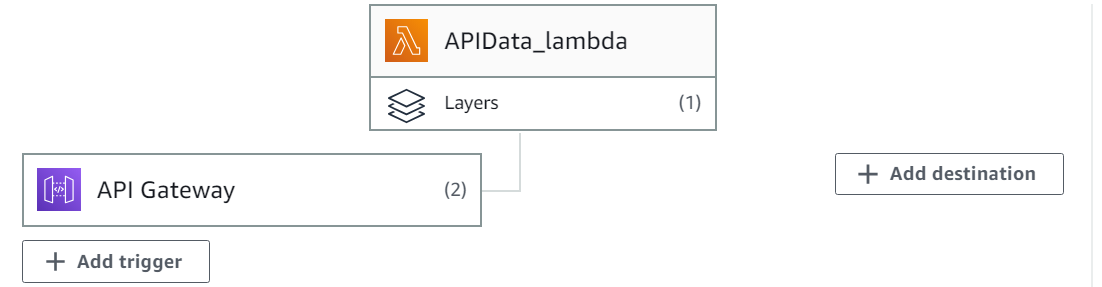
Create an IAM role for the Lambda function with the necessary permissions,

including AWSLambdaBasicExecutionRole, AmazonAPIGatewayAdministrator, AmazonLexRunBotsOnly, and AmazonLexFullAccess.

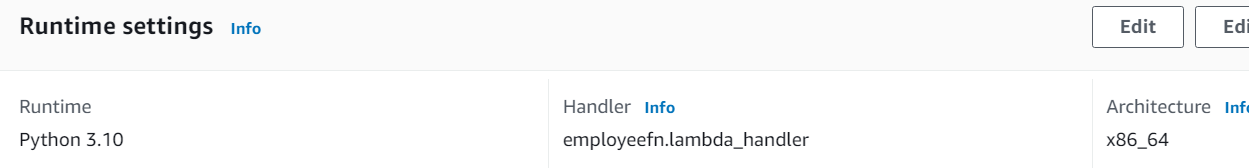


Uploading the code in lambda function:

1. Open the lambda function , add a layer (AWSSDKPandas-Python310) to the lambda function
2. Add a trigger, select API gateway-🡪 create a API gateway for the get method using http request to invoke lambda function.



1. Go to the code section , click on Upload from button , select .zip and select the deployment package that we created.
2. Under Runtime settings handler should be like lambda\_function.lambda\_handler. (For our case handler function is employeefn.lambda\_handler).



5. Create a test event to validate the functionality of the Lambda function.

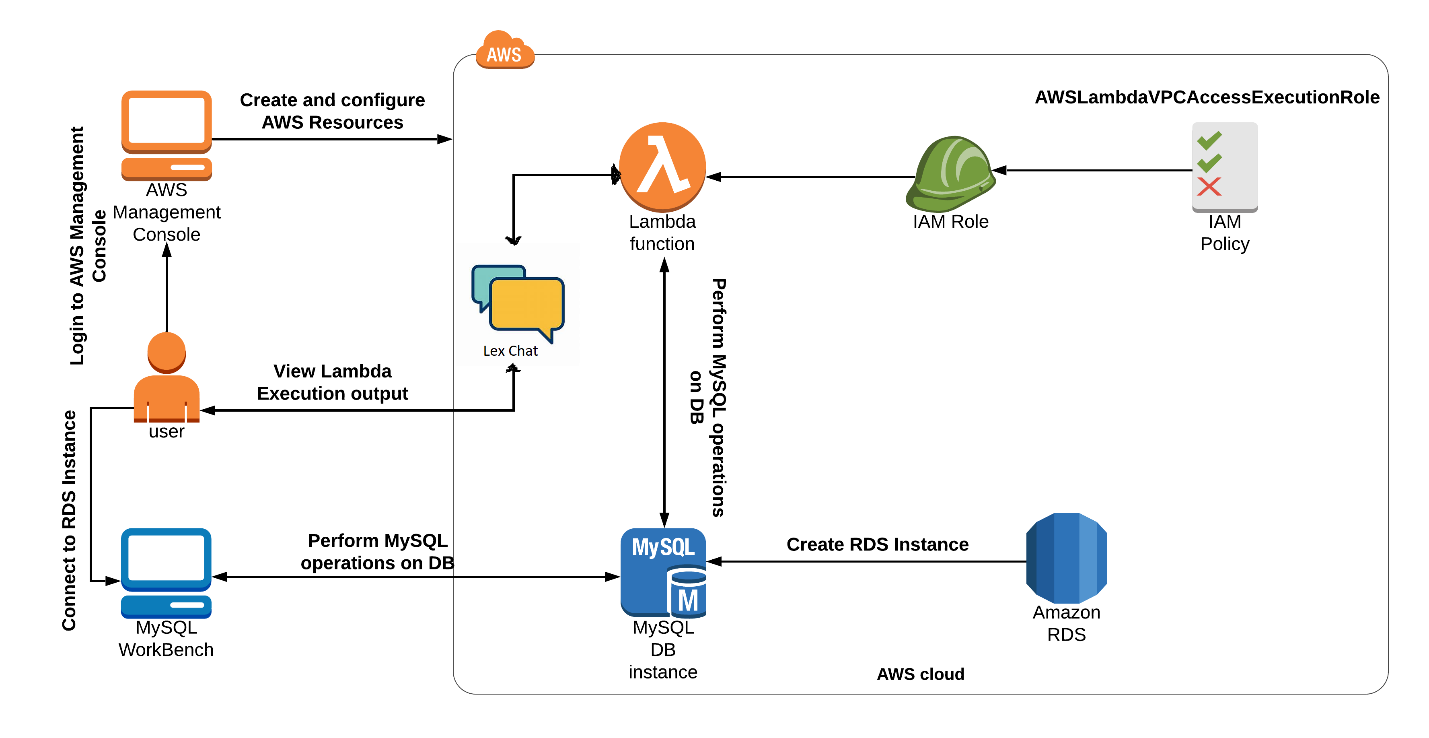
**Scenario 3: AWS lex chatbot interacting with Relational Database Services integrating with Mysql for employee counts**

**Introduction and Overview of the AWS Lex Chatbot with RDS Database Integration**

Welcome to our groundbreaking AWS Lex chatbot implementation, where we have leveraged the power of AWS services to create a seamless and intelligent conversational interface. Our chatbot is not only capable of interacting with users using natural language but also integrates smoothly with a real-world RDS database, allowing dynamic retrieval of employee count data based on user queries.

**Architectural Diagram:**

Our architectural diagram showcases the flow of data between the user, AWS Lex, AWS Lambda function, and the RDS database.



**Creating RDS Database:**

1. search for RDS services, click on databases.
2. Create database, choose database method as **easy create**
3. Select configuration type as MySql
4. Choose DB instance size as per requirement (for my case I choose free tier)
5. Give a name to DB Cluster Identifier
6. Give Master Username
7. Create Password (save username and password for reference)
8. Click on create database.
9. Click on the created database, after some time it will generate endpoint and port for the database under connectivity and security tab.
10. Under security section Publicly accessible should be **Yes**

Under security group we should have these rules

|  |  |  |
| --- | --- | --- |
| **Security group** | **Type** | **Rule** |
| [default (sg-0b530e1a100fc81c3)](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#SecurityGroups:search=default) | CIDR/IP - Inbound | 0.0.0.0/0 |
| [default (sg-0b530e1a100fc81c3)](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#SecurityGroups:search=sg-0b530e1a100fc81c3) | EC2 Security Group - Inbound | sg-0b530e1a100fc81c3 |
| [default (sg-0b530e1a100fc81c3)](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#SecurityGroups:search=default) | CIDR/IP - Inbound | 0.0.0.0/0 |
| [default (sg-0b530e1a100fc81c3)](https://ap-southeast-1.console.aws.amazon.com/ec2/v2/home?region=ap-southeast-1#SecurityGroups:search=default) | CIDR/IP - Outbound | 0.0.0.0/0 |

Next, Install Mysql server in our system, open mysql workbench from our local system and add mysql connection using **+** button.

Give connection name, under parameters tab give

* Hostname: endpoint Url generated in RDS database from aws console.
* Username: username from RDS-database
* Port: give the port number

Click on Test Connection button, click ok. Then open the connection enter the password, click ok.

**Creating data in Mysql database:**

Creating Database :**CREATE DATABASE** RDS**\_TEST;**

Use database**: Use** **RDS\_TEST;**

Creating Table : **CREATE TABLE Department (**

**dept\_name VARCHAR(255) NOT NULL,**

**emp\_count INT NOT NULL,**

**PRIMARY KEY (dept\_name)**

**);**

Create data in Department table:

**INSERT INTO Department (dept\_name, emp\_count)**

**VALUES**

**('Sales', 10),**

**('Marketing', 8),**

**('Finance', 15),**

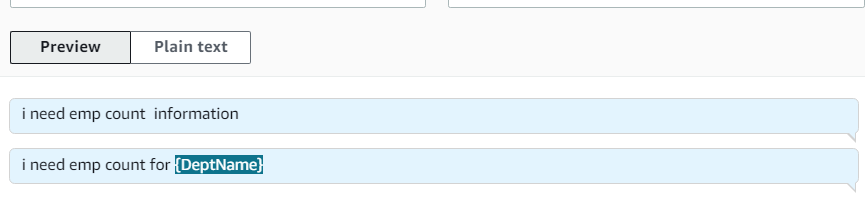
**('CCS', 25),**

**('Payments', 20),**

**('Innovation', 10);**

**Creating lex chatbot in aws console:**

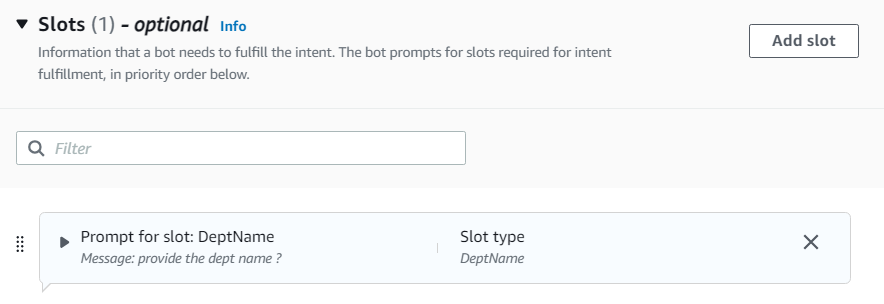
1. Go to aws management console , search for amazon lex services click on create a bot name it as [RDSDatabaseBot](https://ap-southeast-1.console.aws.amazon.com/lexv2/home?region=ap-southeast-1#bot/11WIQCIZI1).
2. Click the APIEmployeedata bot, go to intents section give it a name for the intent of the lexbot named as [MysqlInstance](https://ap-southeast-1.console.aws.amazon.com/lexv2/home?region=ap-southeast-1#bot/11WIQCIZI1/version/DRAFT/locale/en_US/intent/2C5LTQMXCV).
3. Add some utterances in order to activate the conversation with the lexbot
4. i need emp count information
5. i need emp count for {DeptName}



1. Add slots for the lexbot to take values as user input and mark it as required for this intent

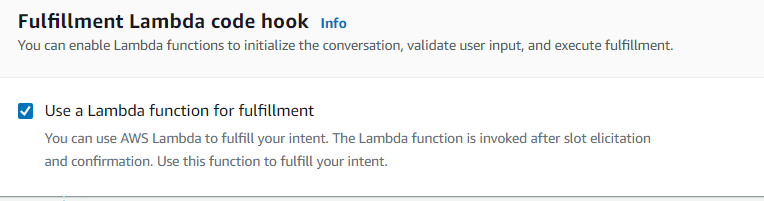
1. DeptName: DeptName (custom slot type)

This custom slot type having Values (Sales,Marketing,Finance, CCS, Payments ,Innovation)

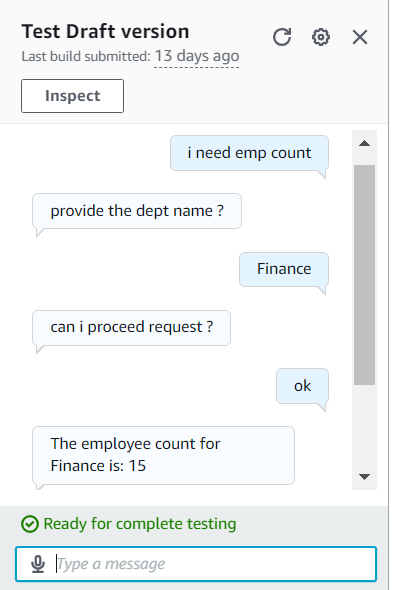


5. Go to the conformation section, add confirmation message (“can I proceed with our request?”).

6. Go to the fulfillment section, select advanced options and check the checkbox for Use a Lambda Function for Fulfillment.



7. save the intent and click on Build. And test the bot using Test feature.



Alias: Deploy the bot using aliases features and select the lambda code as source and version as LATEST.

**Lambda Code:**

The AWS Lambda function plays a vital role in fetching the employee count from the RDS database based on the department name provided by the user. The function establishes a connection with the MySQL server, executes the query, and retrieves the employee count for the specified department. The Lambda function then prepares a response, including the employee count, which is sent back to the Lex chatbot for user feedback.

**Pre-requisities:**

For to run above code we need to install the dependencies in the local directory folder by running this command **pip install pymysql** and save the lambda code as **lambda\_function.py** in the local directory folder and zip it. And upload the lambda code using zip folder. Deploy the code Create Test event , and test the code.

**Configuration**:

* Make sure the lambda Runtime handler should be **lambda\_function.lambda\_handler**.
* And add a layer to the lambda function **AWSSDKPandas-Python310**
* For IAM role that is associated with lambda function should contain these permissions

[AmazonRDSFullAccess](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonRDSFullAccess)

[AWSLambdaBasicExecutionRole](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2Fservice-role%2FAWSLambdaBasicExecutionRole)

[AmazonLexRunBotsOnly](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonLexRunBotsOnly)  
[AmazonLexReadOnly](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonLexReadOnly)  
[AmazonLexFullAccess](https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/policies/details/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonLexFullAccess)

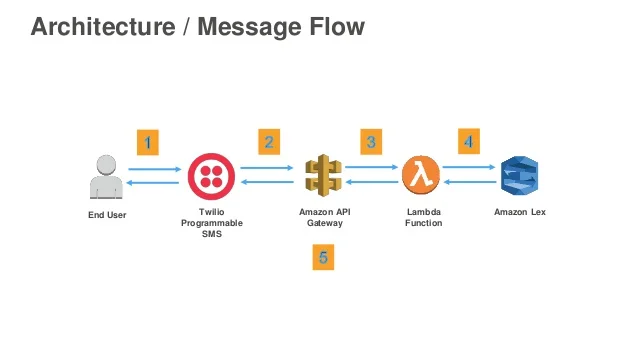
**Scenario 4: AWS Chatbot Interacting with Messaging Services like Whatsapp using Twillio**

**Introduction and Overview: Building a Twilio Integrated Chatbot using AWS Lex**

Welcome to the world of intelligent and interactive chatbots! In this implementation, we have combined the power of AWS Lex and Twilio to create a cutting-edge chatbot that seamlessly communicates with users over WhatsApp. Our chatbot is designed to provide an engaging and personalized experience, allowing users to interact with it via WhatsApp messages.

**Architecture Diagram:**

The architectural diagram illustrates the flow of data and interactions between various components of our chatbot system. The user interacts with the chatbot through WhatsApp, which is facilitated by Twilio. Twilio acts as the communication API, enabling the connection between the user's WhatsApp and our chatbot.

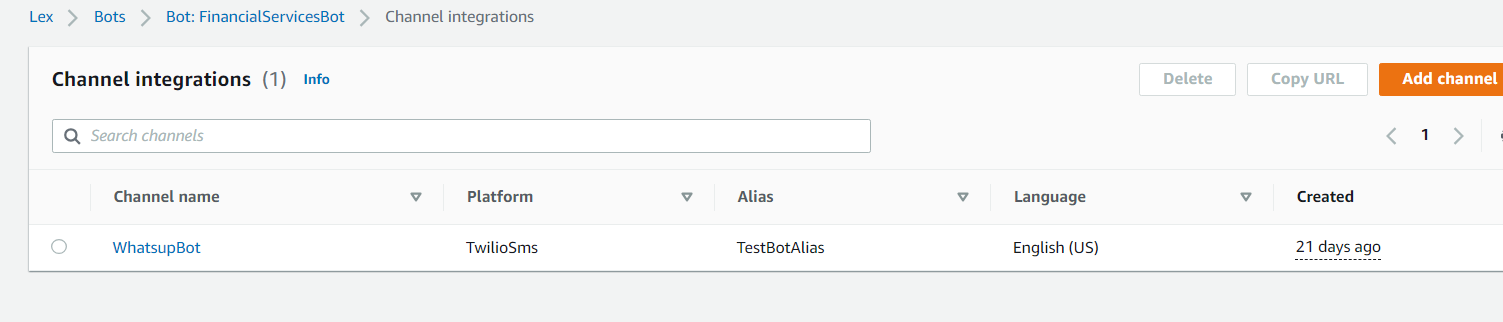


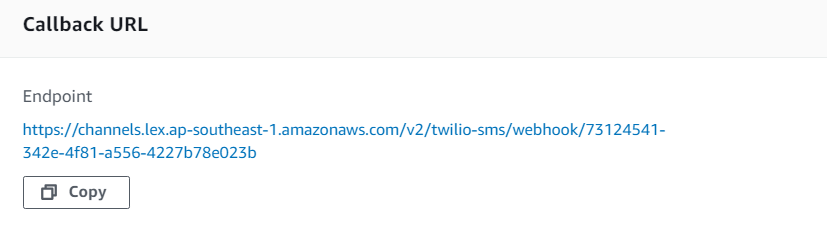
**Creating Twillio Account:**

* Go this link [Communication APIs for SMS, Voice, Video & Authentication | Twilio](https://www.twilio.com/en-us)
* Click on start for free button.
* Create it with First name, lastname, email and password.
* It will ask us to verify email address, verify it. And we need to enter our phone number and verify it with OTP and click submit.
* After that it will ask few questions to enter.
  + Questions:
  + Which Twillio product are you here to use : Whatsapp
  + What are you plan to build with Twillio : Contact Center
  + How do you want to build with Twillio : With Code
  + Click on continue
  + What is your preferred language ?: Python
  + Would you like Twillio to host your code? No, Iwant to use my own hosting service.
  + Click continue.
* It will open the Twillio console, activate your sandbox, After activating it will ask us send a message to whatsapp number with custom code.
* We need to save the whatsapp number and send a message with a given custom code from our whatsapp then sandbox will be activate from our whatsapp.
* In Twillio console, after activating, we can send default template message to our whatsapp to check its properly configured or not by sending make request.
* Next to this there is option to select the python code, click on show auth token checkbox and save the account\_sid and auth\_token (in notepad) click on next step.
* Next go to the aws console, go the bot which we created under channel integrations and select the platform as Twillio SMS and give the name and select alias as TestBotAlias Languase as EN\_US

**Addition Configuaration Settings:**

* Alias: TestBotAlias
* Access Token: account\_sid(from twillio)
* Secret Key: auth\_token (from twillio)
* Click add. After creating it will generate an Callback Url we need to copy that for reference.





* Go to twillio cosole go to sandbox settings and paste the url in when a message come in field and click on save.
* Now go to the whatsapp and activate the sandbox with custom code and send a message with utterance to chatbox and then lexchatbot will start interacting with users.