

# Amazon Lex: Build the BookHotel Bot

## And integrate it with Whatsapp

### What and Why Amazon Lex?

Amazon Lex is a service provided by Amazon Web Services (AWS) that enables developers to build conversational interfaces (chatbots) using natural language understanding (NLU) and automatic speech recognition (ASR). Here are some key details about Amazon Lex:

#### Conversational Interfaces:

Amazon Lex allows you to create conversational interfaces for applications using both text and voice. It supports natural language understanding to process user input and generate appropriate responses.

#### Intent Recognition:

Lex uses intent recognition to understand the user's intention behind a statement or question. You define intents that represent the different actions your bot can take, and Lex identifies the user's intent based on their input.

#### Slot Filling:

Within intents, Lex uses slot filling to extract specific pieces of information (slots) from the user's input. Slots represent the parameters needed for the intent to be fulfilled.

#### Utterances:

Utterances are sample phrases that users might say to convey a specific intent. By providing a variety of utterances, Lex improves its ability to understand and recognize similar expressions from users.

#### Bot Deployment:

Once you define your bot using the Lex console or API, you can deploy it on various platforms, such as messaging services, mobile devices, or web applications.

#### Integration with Other AWS Services:

Amazon Lex can be integrated with other AWS services, such as Lambda functions for backend processing, Amazon Polly for text-to-speech, and Amazon S3 for storing conversation logs.

#### Multi-Turn Conversations:

Lex supports multi-turn conversations, allowing your bot to maintain context across interactions. This enables more natural and context-aware conversations.

#### Voice and Text Input:

Lex supports both voice and text input. For voice input, it uses automatic speech recognition (ASR) to convert spoken language into text.

#### Channel Integration:

Lex allows you to integrate your bot with various messaging platforms, such as Facebook Messenger, Slack, and Twilio, making it versatile for different use cases.

#### Scalability:

Being a managed service on AWS, Amazon Lex is designed to scale automatically based on demand. This ensures that your chatbot can handle varying workloads.

### Security:

Lex provides security features such as encryption in transit and at rest. Additionally, you can control access to your Lex bots using AWS Identity and Access Management (IAM) roles.

### Analytics:

Amazon Lex provides analytics and logging capabilities, allowing you to monitor and analyse interactions with your chatbot. This helps in improving the performance and understanding user behaviour.

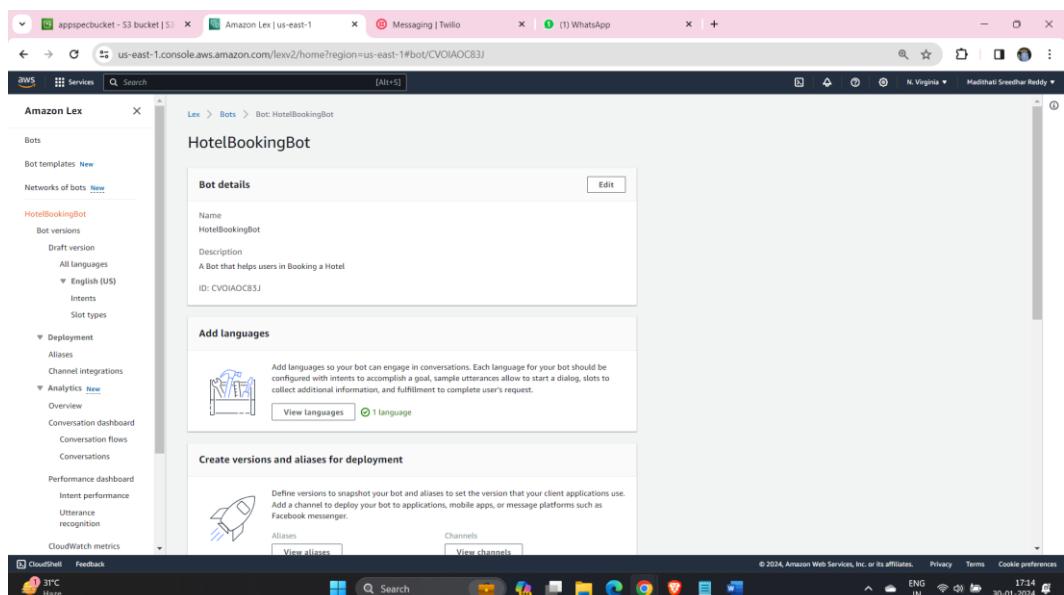
## Project Architecture:



## How to do?

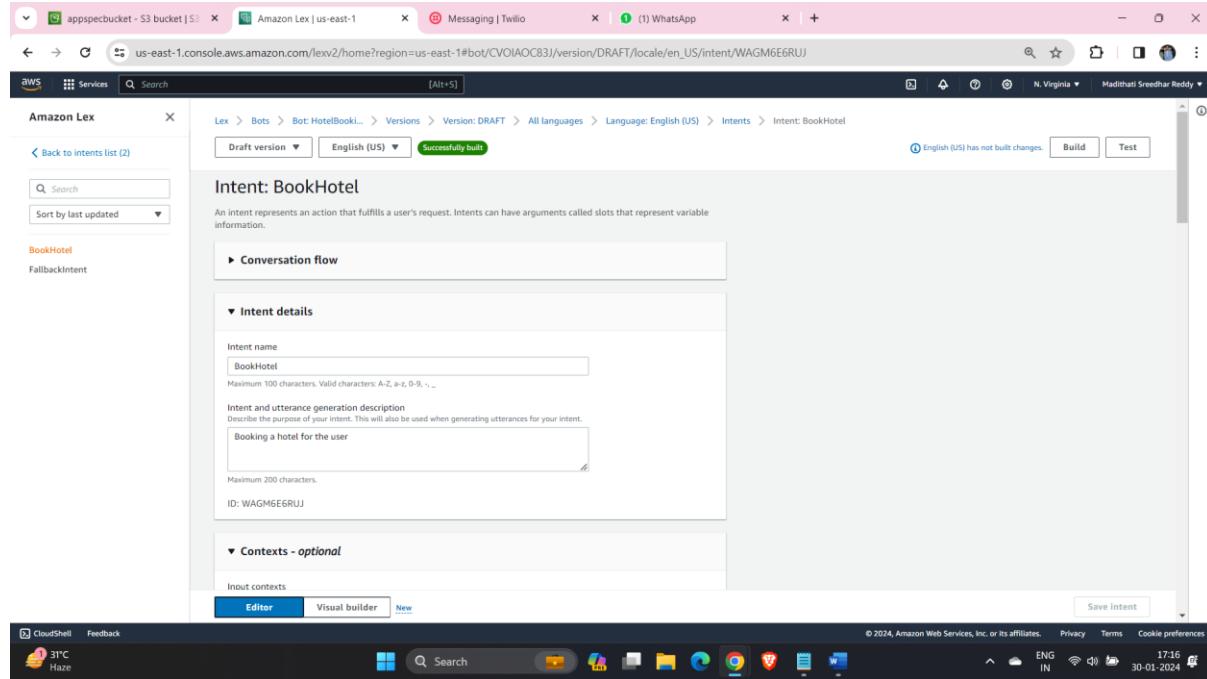
### Step 1: Creating and configuring the bot in Amazon Lex

1. Create an Amazon Lex Bot:
  - Go to the Amazon Lex console.
  - Click "Create" to create a new bot.
  - Choose a custom bot and provide necessary details.



## 2. Create Intents:

- Create an "BookHotel" intent.
- Define sample utterances like "Book a hotel," "Reserve a room," etc.
- Define slots for information like "location," "check-in date," "nights", etc.



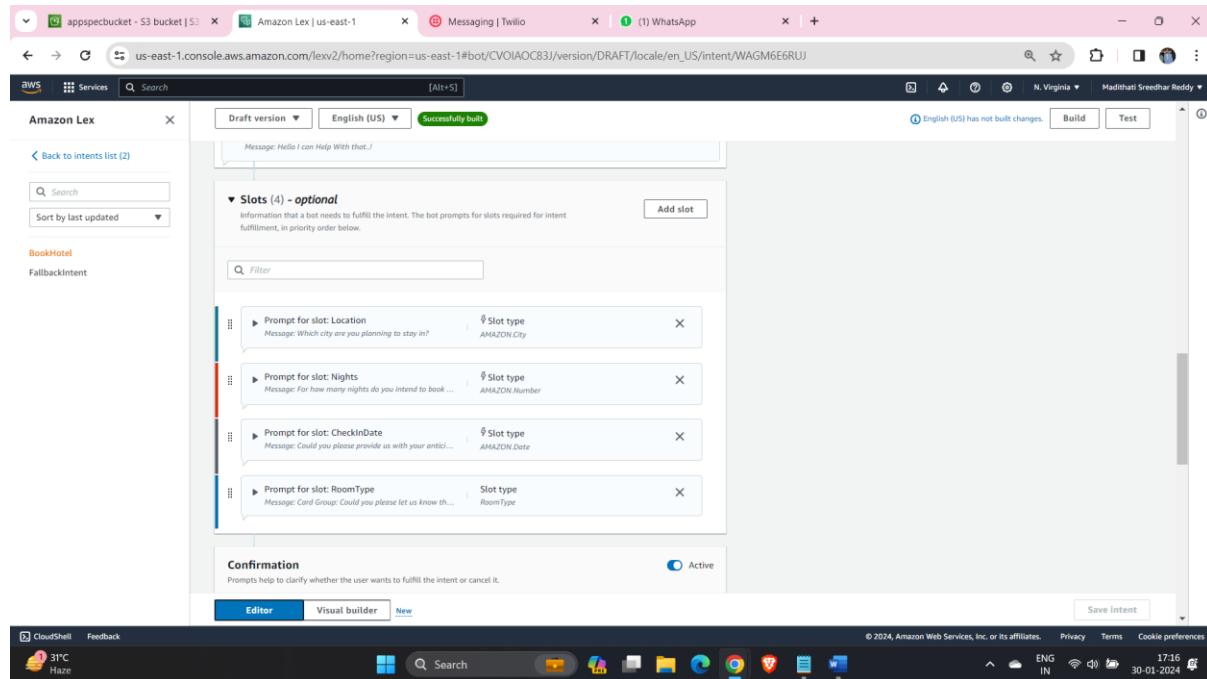
The screenshot shows the AWS Lambda function configuration page. The top navigation bar includes tabs for 'Code', 'Environment', 'Triggers', 'Logs', and 'Metrics'. The 'Code' tab is selected, showing the code editor with the following code:

```
function handler(event, context) {
  // Your code here
}
```

The 'Edit' button is highlighted in blue at the bottom of the code editor.

## 3. Configure Slots:

- Configure slot types for each slot in the intent (e.g., date, city).



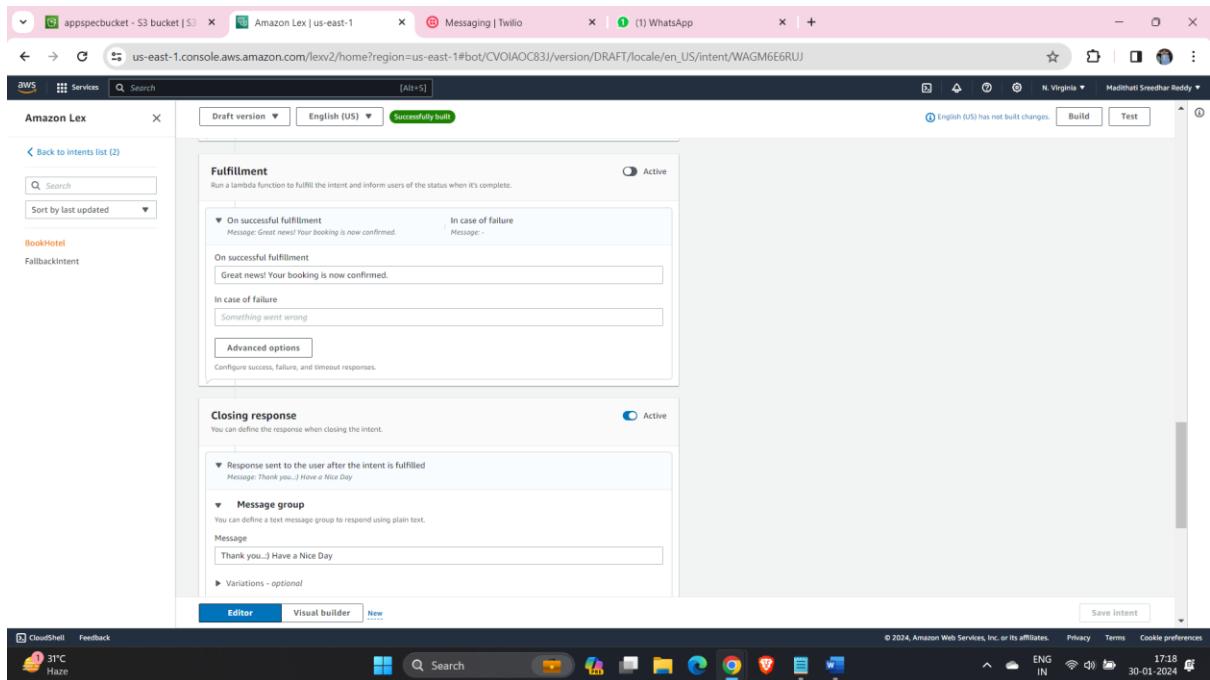
The screenshot shows the AWS Lambda function configuration page. The top navigation bar includes tabs for 'Code', 'Environment', 'Triggers', 'Logs', and 'Metrics'. The 'Code' tab is selected, showing the code editor with the following code:

```
function handler(event, context) {
  // Your code here
}
```

The 'Edit' button is highlighted in blue at the bottom of the code editor.

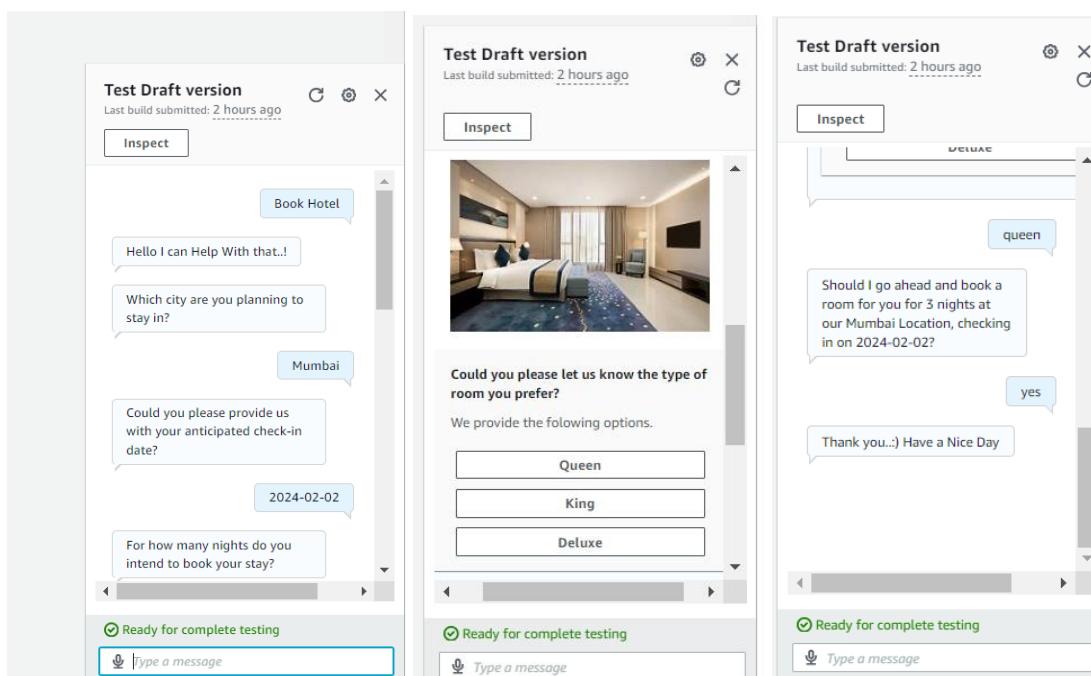
## 4. Fulfillment (Optional):

- Define a closing and success responses for fulfillment. This function will process the slots and take appropriate responses upon failure and success of request.



## 5. Build and Test:

- Build and test your bot in the Lex console to ensure that it understands user inputs and extracts the required information.



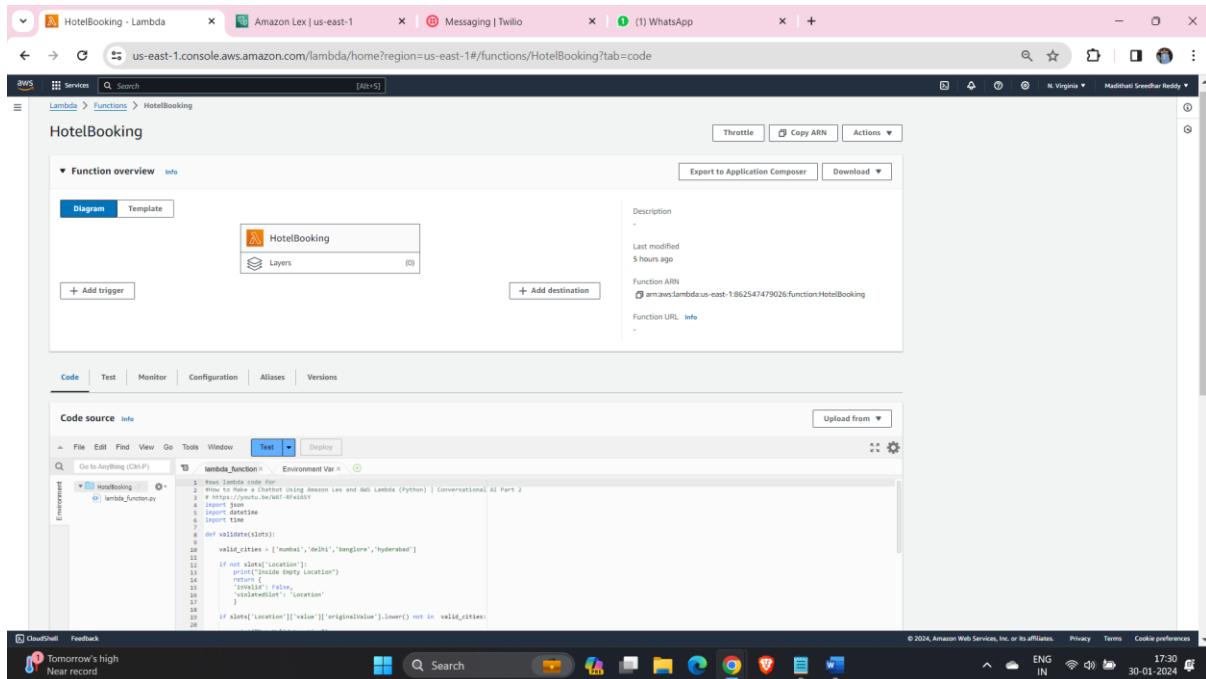
## Step 2: Configure AWS Lambda:

### 1. Create Lambda Function:

Define a Lambda function for fulfillment if you want to perform backend processing. This function will process the slots and take appropriate actions.

- Create a new Lambda function in the AWS Lambda console.
- Write the code to process the booking details received from Lex.
- Copy and paste the code from GitHub

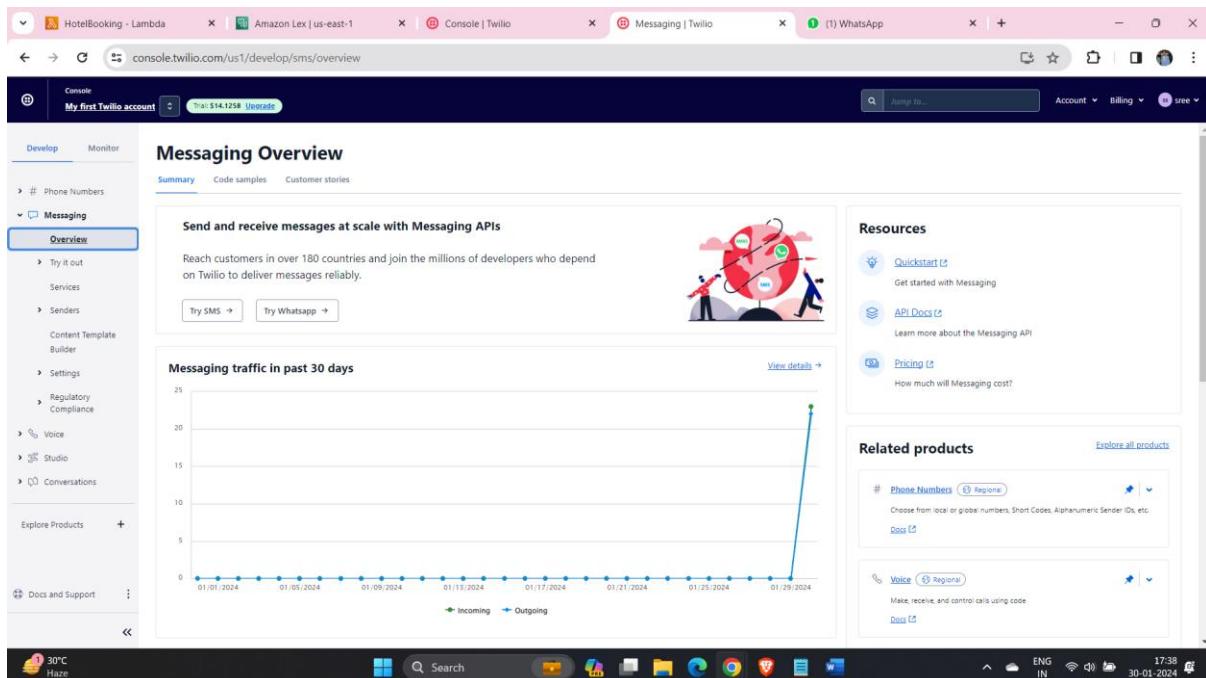
URL: <https://github.com/uniquesreedhar/Chat-Bot-Project.git>



## Step3: Twilio: Set Up Twilio Account

### 1. Sign Up for Twilio:

- Sign up for a Twilio account if you don't have one: [Twilio Sign Up](#).



### 2. Get Twilio API Credentials:

- After signing up, get your Twilio Account SID and Auth Token from the Twilio console.

A screenshot of the Twilio console interface. At the top, there are tabs for 'Console - Lambda', 'Amazon Lex | us-east-1', 'Console | Twilio', and 'Console | Twilio'. Below the tabs, it says 'Ahoy sree, welcome to Twilio!'. On the left, a sidebar shows 'My first Twilio account' with sections for 'Phone Numbers', 'Messaging', 'Voice', 'Audio', 'Conversations', 'Products', and a '+' button. The main content area has a heading 'Get a Twilio phone number' with a note about trial numbers. It shows 'You've got a trial phone number!' and a link to 'Account info below'. There's also a note about local numbers outside the USA/Canada and a link to 'Read the regulatory requirements'. To the right is an illustration of two people standing next to a globe with a speech bubble. Below the illustration are 'Helpful links' including 'How does Twilio work?', 'API documentation', 'Explore Marketplace', and 'Support help center'. A 'Skip' button is at the bottom right.

### 3. Create a Twilio WhatsApp Sandbox:

- In the Twilio console, go to the WhatsApp Sandbox.
- Follow the instructions to set up a WhatsApp sandbox.

A screenshot of the Twilio console under the 'Messaging' section. The left sidebar shows 'Try it out' with 'Send a WhatsApp message' selected. The main area is titled 'Try WhatsApp' with the sub-section 'Twilio Sandbox for WhatsApp lets you test your app in a developer environment without WhatsApp approval for your account.' It shows a 'Sandbox' button and 'Sandbox settings'. Below this, there are four options: 'Connect to sandbox', 'Business initiated message', 'User initiated conversation', and 'Wrap up'. A 'Next step' button is at the top right. The central part of the screen has a large text area for 'Send a WhatsApp message' with a placeholder 'Use WhatsApp and send a message from your device to the Twilio number.' It includes a QR code labeled 'Scan the QR code on mobile' and a link 'Open WhatsApp' with an 'OR' option. The bottom of the screen shows a Windows taskbar with various icons and system status.

## Step 4: Integrate Amazon Lex with Twilio

1. Create a Version or can use the default one :

The screenshot shows the AWS Lambda console for the 'HotelBooking' Lambda function. The function name is 'HotelBooking'. The configuration tab is selected. Under the 'Handler' section, the handler is set to 'index.handler' and the runtime is 'Node.js 18.x'. The 'Memory' dropdown is set to '128 MB'. The 'Timeout' dropdown is set to '300 seconds'. The 'Environment' section shows environment variables: 'BOT\_NAME' (value: 'HotelBookingBot'), 'BOT\_ALIAS' (value: 'HotelBotAlias'), 'BOT\_REGION' (value: 'us-east-1'), and 'BOT\_STACK\_ID' (value: 'CVOIAOC83J'). The 'Logs' section shows log groups: '/aws/lambda/HotelBooking' and '/aws/lambda/HotelBooking/CloudWatchLogsGroup'. The 'Tracing' section shows tracing is disabled. The 'Deployment' section shows the deployment role 'AWSLambdaRoleForLexV2Bots\_NTK13FILXF'. The 'Logs' section shows log groups: '/aws/lambda/HotelBooking' and '/aws/lambda/HotelBooking/CloudWatchLogsGroup'. The 'Metrics' section shows CloudWatch Metrics is enabled. The 'CloudWatch Metrics' section shows CloudWatch Metrics is enabled. The 'CloudWatch Metrics' section shows CloudWatch Metrics is enabled.

## 2. Then create an alias for that version:

The screenshot shows the Amazon Lex console. The left sidebar shows the navigation path: Lex > Bots > Bot: HotelBookingBot > Aliases > Alias: HotelBotAlias. The main area displays the 'Alias: HotelBotAlias' details page. The 'Details' section shows the alias name 'HotelBotAlias' and the associated version 'Version 3'. The 'Languages' section shows 'English (US)' is selected and successfully built. The 'Conversation logs' section shows 'Conversation logs is disabled'. The bottom status bar indicates the date and time as '30-01-2024 19:21'.

## 3. Configure Twilio Integration in Amazon Lex:

- In the Lex console, go to the "Channels" tab.
- Add a new messaging channel and choose Twilio.

The screenshot shows the AWS Lambda console interface. A new channel is being created for the 'HotelBooking' bot. The 'Twilio SMS' integration is selected. In the 'Integration configuration' section, the channel name is set to 'HotelTwilio' and the description is 'Customer Support'. The IAM role is 'AWSLambdaServiceRoleForLexV2Channels\_CVOIAOC83J\_2FF800HURSM'. The KMS key is 'aws/lex'. The deployment alias is 'HotelBotAlias' and the language is 'English (US)'.

- Enter your Twilio Account SID and Auth Token.
- Then Add the Channel and copy the endpoint URL

The screenshot shows the AWS Lambda console interface. The 'General configuration' section for the 'HotelTwilio' alias is displayed. The alias name is 'HotelTwilio', the IAM role is 'AWSLambdaServiceRoleForLexV2Channels\_CVOIAOC83J\_2FF800HURSM', the platform is 'TwilioSMS', the alias is 'alias/awsls/lex', and the language is 'English (US)'. The 'Callback URL' section shows the endpoint URL: <https://channels.lex.us-east-1.amazonaws.com/v2/twilio-sms/webhook/a74c5519-5fde-4de6-b43d-75f602693179>. A 'Copy' button is available to copy the endpoint URL.

- Configure the Lambda (if used) as the fulfillment Lambda in the alias created.

The screenshot shows the AWS Lambda function configuration for the HotelBooking bot's Alias language support. The Lambda function is set to 'HotelBooking' and the version is '\$LATEST'. The 'Save' button is visible at the bottom right.

## Step5: Test the Integration:

### 1. Test in Twilio Sandbox:

- Use the Twilio WhatsApp sandbox number to test your integration.

The screenshot shows the Twilio Console interface for testing WhatsApp messages. It displays the Twilio Sandbox configuration for WhatsApp, including the webhook URL and status callback URL. The 'Sandbox settings' tab is selected.

2. Send messages to the Twilio WhatsApp number, and the messages should be processed by your Amazon Lex bot.



If you get expected responses to the requests, you made...:

**Congratulations you have successfully completed the project... 😊**