

Creating a Chatbot using Amazon Flex Service

-Akhil Mittal

Table of Contents

Preface	1
Introduction	1
Setup AWS account.....	1
Create Bot using Amazon Lex	4
Test the Bot.....	10
AWS Lambda Function.....	13
Slots.....	17
Testing Slots	19
Error Handling	21
Deploy the Chatbot in Facebook Messenger.....	24
Live Chatbot on Facebook.....	35
Conclusion.....	37

Preface

This article is the submission against CodeProject's [Slack API Challenge](#). The contest officially began on January 7, 2019, and ends on March 7, 2019.

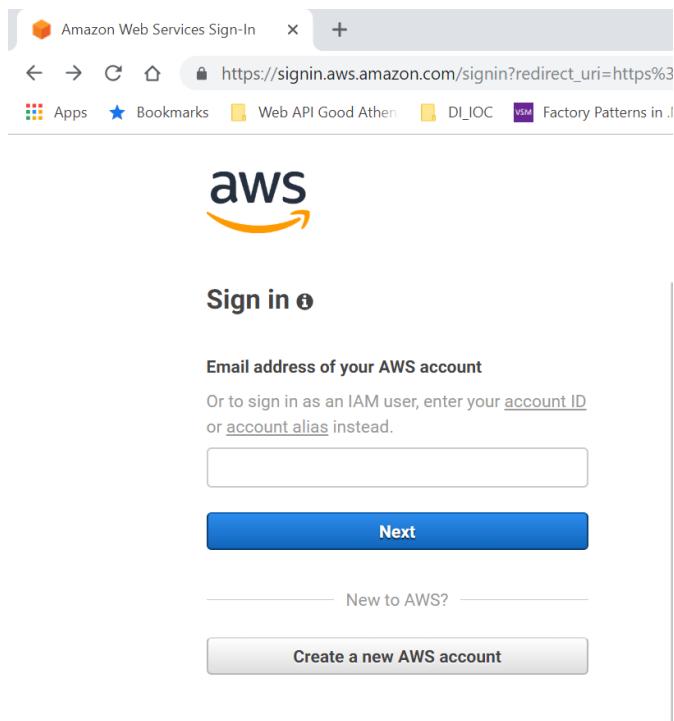
Introduction

Slack is no doubt the platform that every developer needs nowadays and is trending in a new way. It is easy to use and manage, and not only this we can create our own slack app and add a lot of custom functionality to it. You can learn how to create your own slack app in [this](#) article. It is very easy and may hardly take 20 minutes to create your first app.

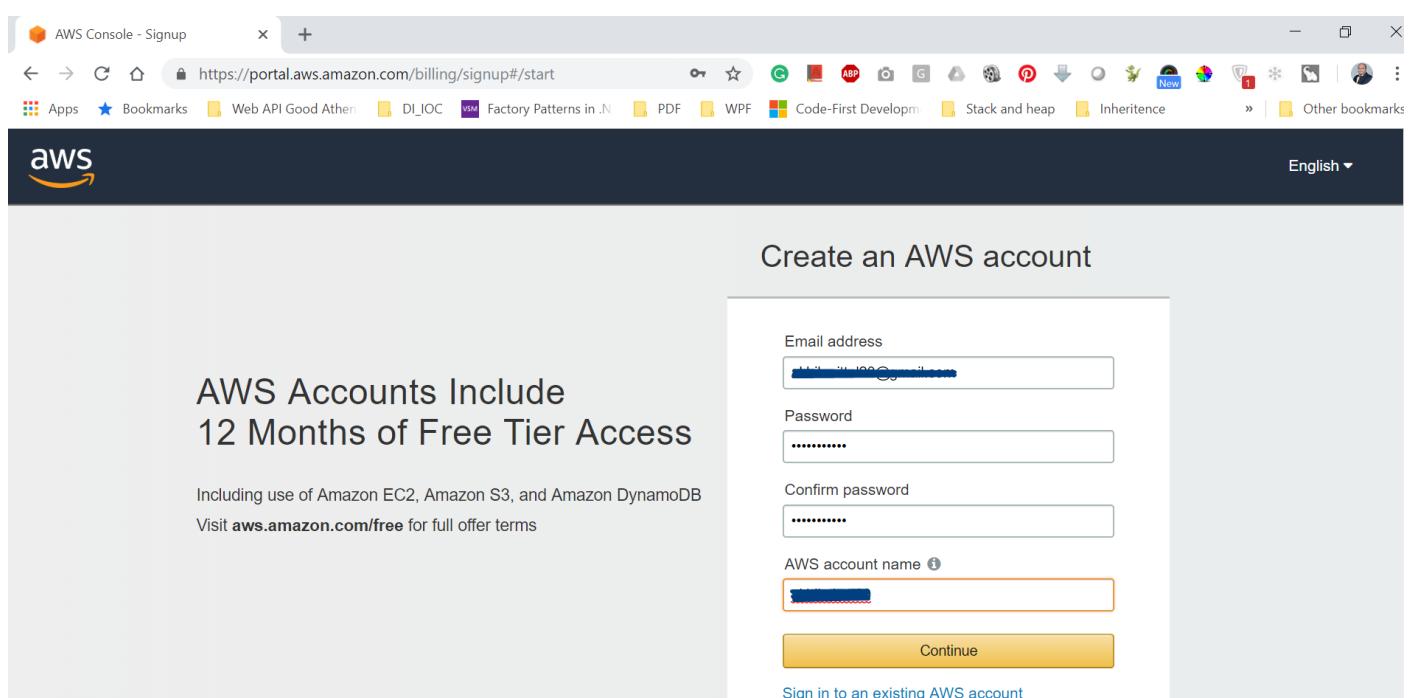
In this article, we'll discuss how to create a real-time chatbot using Amazon Flex service and deploy the bot over Facebook messenger. The reader of the article does not need to have prior knowledge of bot creation as the article will demonstrate the bot creation and deployment step by step using explanatory pictures. Trust me, it is fun and exciting to create your own bot and see it working over the thought of creating a bot. We'll setup AWS account to leverage Lex Services and Facebook developers account to deploy the bot on your own Facebook page and see it working. Let's jump directly into the tutorial.

Setup AWS account

1. Go to aws.amazon.com and sign in using your email address if you have an account or create a new account.



2. Provide the relevant details like email address and password while creating a new account.



3. On the next step, you can choose to create a Personal type account and provide needed information there.

Contact Information

All fields are required.

Please select the account type and complete the fields below with your contact details.

Account type i

Professional Personal

Full name

Phone number

Country/Region

▼

Address

4. Next page is the payment information page where you need to provide the payment mode and its details. Nothing to worry about as when you enter your card details it deducts a minimal amount i.e. 2 INR which would be reverted to the account after successful verification of the card.

"With Amazon Lex, you pay only for what you use. You are charged based on the number of text or voice requests processed by your bot, at \$0.004 per voice request, and \$.00075 per text request. For example, the cost for 1,000 speech requests would be \$4.00, and 1,000 text requests would cost \$0.75. Your usage is measured in "requests processed", which are added up at the end of the month to generate your monthly charges."

You can try Amazon Lex for free. From the date you get started with Amazon Lex, you can process up to 10,000 text requests and 5,000 speech requests per month for free for the first year." Read more about Lex pricing [here](#).

Payment Information

Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the [AWS Free Tier Limits](#). Review [frequently asked questions](#) for more information.



As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

Expiration date

01 ▾ 2019 ▾

Cardholder's name

* Cardholder's Name is a required field

Billing address

Use my contact address

- Once successfully entered the console. Choose the region US East (N. Virginia).

The screenshot shows the AWS Management Console homepage. At the top right, there is a dropdown menu for selecting a region. The dropdown is currently open, showing the following options:

- US East (N. Virginia) (highlighted)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)

Create Bot using Amazon Lex

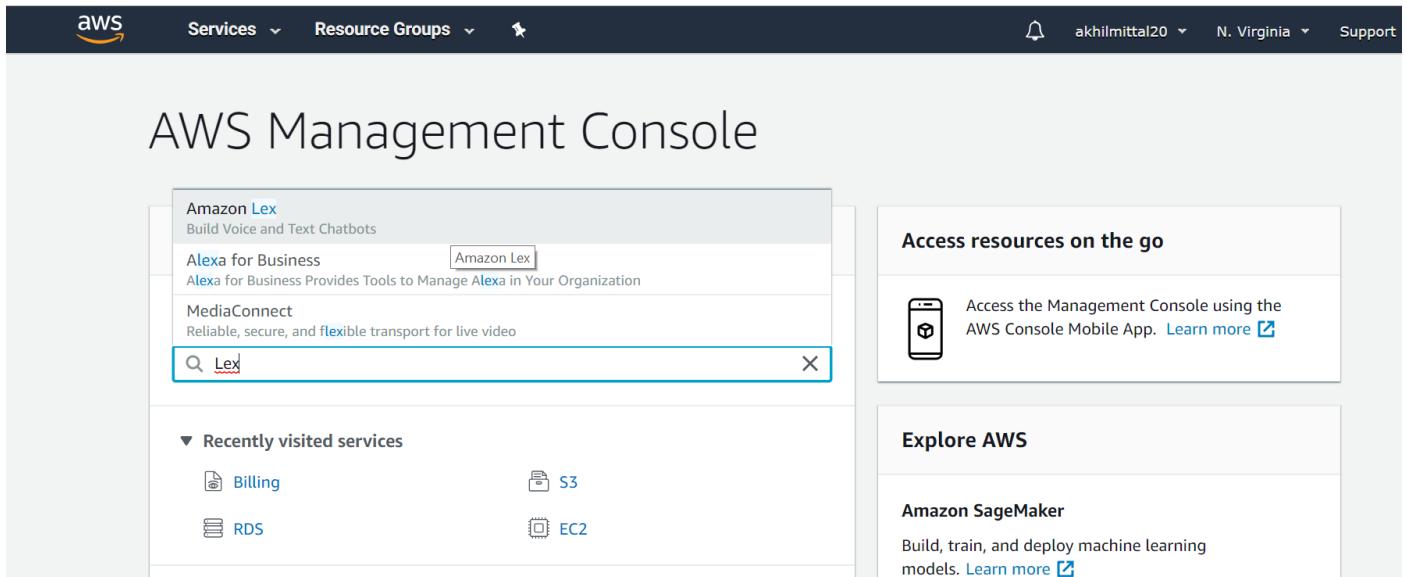
After setting up the AWS account, it is time to create the bot using Lex service. Let's get familiar with the terminology before we start.

A bot is a top-level component that is deployed. It is a kind of container. The bot that we'll develop will have one or more intents. Each one of them represents a different topic that a bot can understand and perform an action to fulfill it.

A Slot is something that we define to get the kind of information we need from the bot user to fulfill the intent.

A Lambda function is simply a place to execute your code that doesn't require you to provision or manage a server, or serverless. Lambda supports multiple programming languages and is how you integrate your bot with other systems. Your AWS account includes 1 million free requests per month and up to 3.2 million seconds of total execution time. Please visit the AWS Lambda pricing page for more details.

1. In the console, search for the service type Lex as shown in the following image.



AWS Management Console

Amazon Lex
Build Voice and Text Chatbots

Alexa for Business Amazon Lex

Alexa for Business Provides Tools to Manage Alexa in Your Organization

MediaConnect
Reliable, secure, and flexible transport for live video

Lex

Recently visited services

Billing S3

RDS EC2

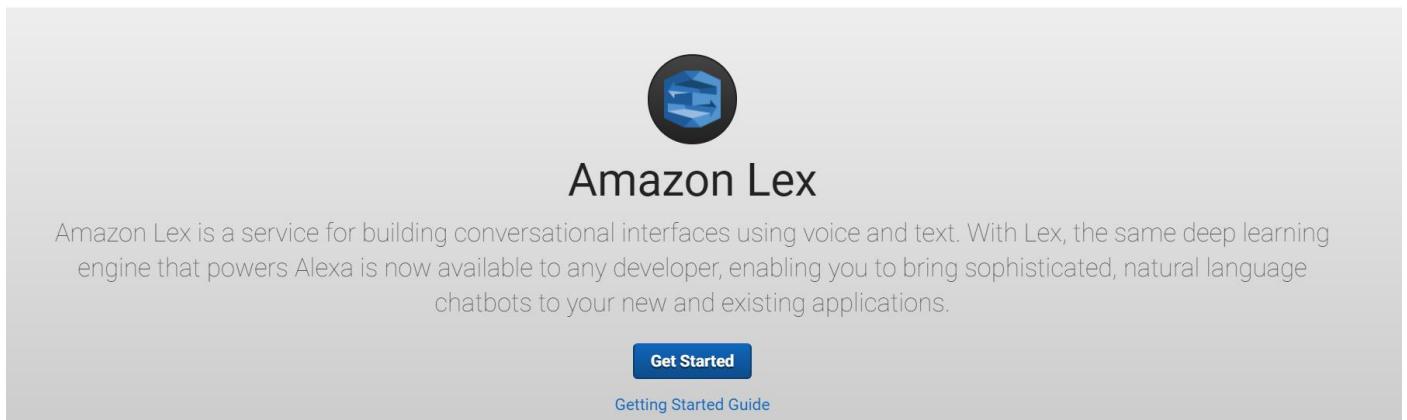
Access resources on the go

Explore AWS

Amazon SageMaker

Build, train, and deploy machine learning models. [Learn more](#)

2. You'll be shown the Amazon Lex landing page. Click on Get Started to start creating the bot.



Amazon Lex

Amazon Lex is a service for building conversational interfaces using voice and text. With Lex, the same deep learning engine that powers Alexa is now available to any developer, enabling you to bring sophisticated, natural language chatbots to your new and existing applications.

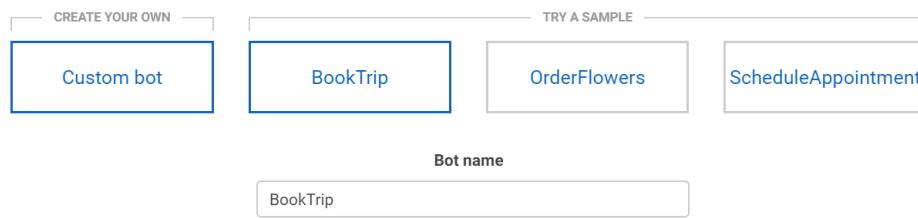
Get Started

Getting Started Guide

3. We can try the existing bot samples and can create our own as well. Since we are creating a bot from scratch, choose the option of the Custom bot and give it a name.

Create your bot

Amazon Lex enables any developer to build conversational chatbots quickly and easily. With Amazon Lex, no deep learning expertise is necessary—you just specify the basic conversational flow directly from the console, and then Amazon Lex manages the dialogue and dynamically adjusts the response. To get started, you can choose one of the sample bots provided below or build a new custom bot from scratch.



- Provide details like bot name. For output voice, choose None as this will only be a text-based bot. Set the timeout to 5 minutes. The IAM role is created for us, so there is no action needed on this line. For COPPA we can choose No.

Bot name BotService

Language English (US)

Output voice None. This is only a text based ... ▾

Session timeout 5 min ⓘ

IAM role AWSServiceRoleForLexBots ⓘ
Automatically created on your behalf

COPPA Please indicate if your use of this bot is ⓘ subject to the [Children's Online Privacy Protection Act \(COPPA\)](#). [Learn more](#)

Yes No

- Once done, you'll see the BotService page. Now it's time to create the intent as the bot is only just a container and it needs intents. Click on create an intent button to create a new intent.

Servi... Resource Groups

BotService Latest

Editor Settings Channels Monitoring

Intents +

No intents created

Slot types +

No slots created

Error Handling

Getting started with your bot

Welcome to your bot editor. You can start right away by adding an intent using the + button in the sidebar.

+ Create Intent

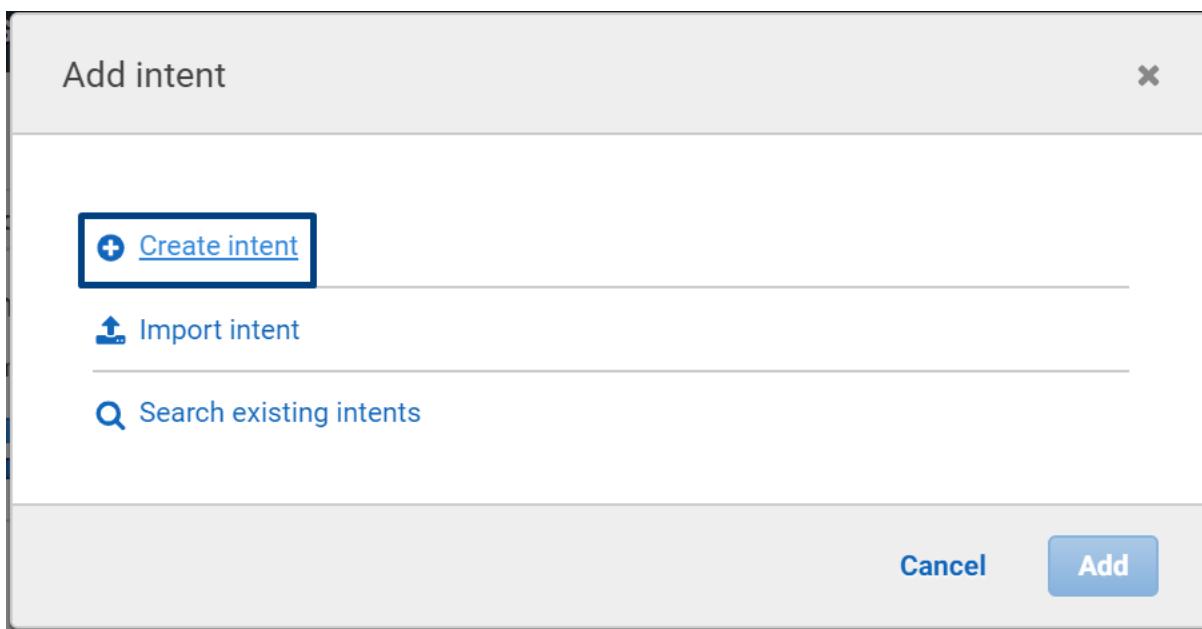
Components of your bot.

BookHotel

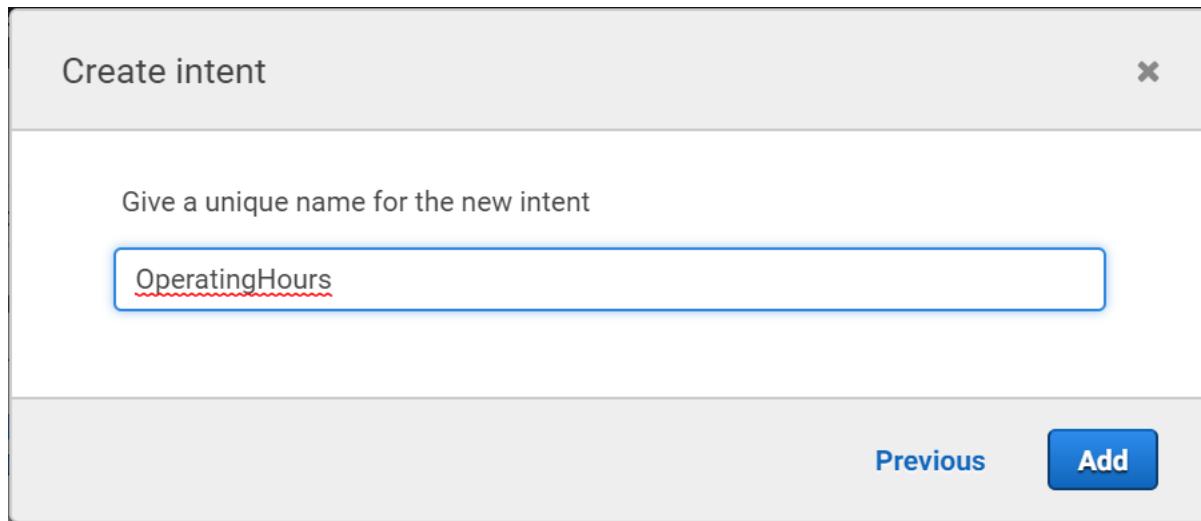
Intents
A particular goal that the user wants to achieve

This screenshot shows the AWS BotService Editor interface. On the left, there's a sidebar with tabs for 'Editor', 'Settings', 'Channels', and 'Monitoring'. Under 'Editor', sections for 'Intents' (no intents created), 'Slot types' (no slots created), and 'Error Handling' are listed. The main area has a heading 'Getting started with your bot' with a sub-instruction about adding an intent. A prominent blue button labeled '+ Create Intent' is centered. To the right, there's a diagram of a grey rectangular component with a blue circle containing a white dot labeled 'BookHotel'. A dashed orange line connects this to a definition of 'Intents' as 'A particular goal that the user wants to achieve'.

6. On the popup, it shows the option to search for existing intents, create new or import an intent. We'll select Create intent option to create a new intent.



7. Give a unique name to the intent. In our case, it is “OperatingHours”. The purpose of this intent to tell the user about the operating business hours of the company for which the user is seeking information. For e.g. when you reach out to a company’s contact us via chat section, the bot welcomes you and if the user asks, “what your working hours are?” then the bot should respond with the company’s working hours and tell that to the user.



8. Adding intent needs some more actions to be performed. For e.g. Sample utterances i.e. the kind of questions a user can ask to bot for e.g. “what are your working hours?”, “what time do you open?”, “what time do you close?” etc. Note that we do not have to provide all possible examples. The built-in language model will recognize similar phrases. Next is Lambda initialization and validation. This is where we can link to the code in an AWS Lambda function to validate the user inputs if we are collecting information as part of the intent. Slots are how we collect data from the user to pass into validation or fulfillment. For this intent, we don't need any specific data from the user, so we will not define any Slots.

OperatingHours Latest ▾

- ▶ Sample utterances ⓘ
- ▶ Lambda initialization and validation ⓘ
- ▶ Slots ⓘ
- ▶ Confirmation prompt ⓘ
- ▶ Fulfillment ⓘ
- ▼ Response ⓘ

Add Message

9. Provide some sample utterances as shown in the following picture.

Intents

OperatingHours

Slot types

No slots created

Error Handling

OperatingHours Latest

Sample utterances

- What are your official working hours?
- What time do you open?
- What time do you close?
- What time do you start your business?
- What are your working hours?

10. A confirmation prompt is for double checking with the user that we have everything understood correct from the user before the bot fulfills the intent. So, enable Confirmation prompt to see how it works. Since we didn't collect any data from the user, ask a simple question. If the user says yes, they move onto fulfillment. If they say no, then the message we put in this Cancel box will display, and the user will exit the intent.

Intents

OperatingHours

Slot types

No slots created

Error Handling

OperatingHours Latest

Slots

Confirmation prompt

Confirmation prompt

Confirm

Do you want to know hour working hours?

Cancel (if the user says "no")

OK. Thanks!

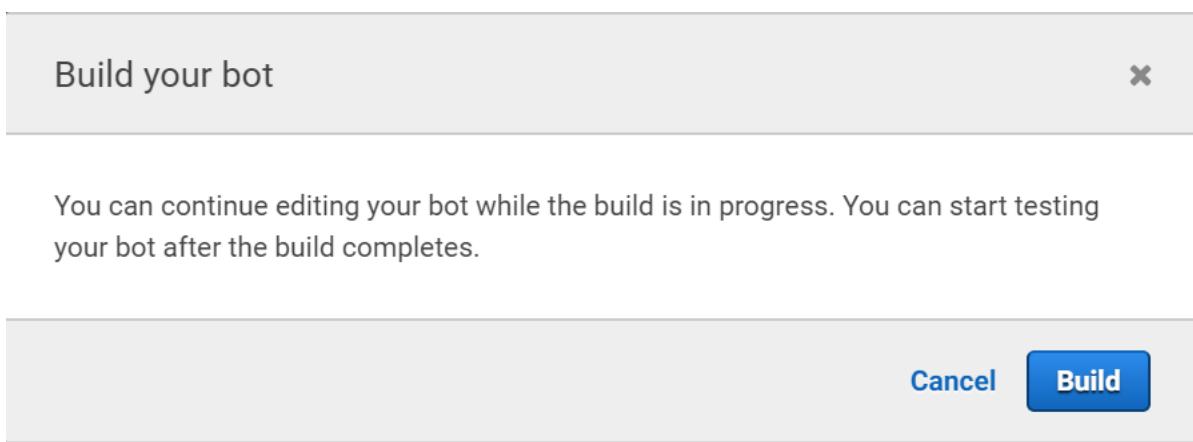
Fulfillment

Response

11. Finally, fulfillment is how the bot performs the result action. Typically, this is done by calling an AWS Lambda function to execute the appropriate business logic. There is also an option to return the parameters to the client. For now, you can choose that option as there is no lambda function for our bot.
12. Now, build the bot via Build the bot via Build option as shown in the following image.

The screenshot shows the Microsoft BotService Editor interface. At the top, there's a navigation bar with 'BotService Latest' and buttons for 'Build', 'Publish', and a question mark icon. Below the navigation is a menu bar with 'Editor', 'Settings', 'Channels', and 'Monitoring'. On the left, there's a sidebar with sections for 'Intents' (with a plus sign), 'OperatingHours' (selected, indicated by an orange border), 'Slot types' (with a plus sign), 'No slots created', and 'Error Handling'. The main content area is titled 'OperatingHours Latest' and contains a section for 'Sample utterances' with five examples: 'e.g. I would like to book a flight.', 'What are your official working hours?', 'What time do you open?', 'What time do you close?', and 'What time do you start your business?'. Each example has a delete icon (an 'x') next to it.

13. Once you click built, you'll be prompted with a message to continue testing your bot while it is building. Click on Build.



14. After a successful build, the prompt will be shown as shown in the following picture.

The screenshot shows the Microsoft BotService interface. At the top, there's a dark header with user info (akhilmittal20, N. Virginia, Support) and a bell icon. Below the header is a navigation bar with 'Build', 'Publish', and a question mark icon. A green notification box is prominently displayed in the center, stating 'BotService build was successful' with a checkmark icon. It also says 'The build is now complete. You can now test the bot in the test window'. To the right of the notification, there's a message placeholder: 'Type an utterance below to begin conversation with your chatbot.' At the bottom, there's a 'Clear chat history' button.

Test the Bot

Our first version of Bot is created now, and we can test it.

1. At the right side of the page, you can see the option to test the latest bot. Start typing in now 😊.

The screenshot shows the Amazon Lex console interface. At the top, there are three tabs: 'Build' (gray), 'Publish' (blue, selected), and a question mark icon. To the right of the tabs, it says '> Test bot (Latest)' and 'Ready. Build complete.' with a green checkmark. A blue arrow points from the 'Publish' tab to the 'Test bot' section. In the 'Test bot' section, there is a text input field with a microphone icon and the placeholder 'Chat with your bot...'. Below the input field, a button says 'Clear chat history'. To the left, there is a list of utterances: 'Jrs' (marked with an 'x'), an empty line (marked with an 'x'), another empty line (marked with an 'x'), 'less' (marked with an 'x'), and another empty line (marked with an 'x'). A blue arrow points from the 'Chat with your bot...' input field to the 'Inspect response' panel. The 'Inspect response' panel has a 'Hide' link in the top right corner. It contains the text: 'When you chat with your bot, you can see the fulfillment state of your intent and the response here.'

2. If you type anything other than the utterances defined in the intent, the bot doesn't recognize that and gives a defined response as shown below.

The screenshot shows the Amazon Lex console interface. At the top, there are three tabs: 'Build' (gray), 'Publish' (blue, selected), and a question mark icon. To the right of the tabs, it says '> Test bot (Latest)' and 'Ready. Build complete.' with a green checkmark. A blue arrow points from the 'Publish' tab to the 'Test bot' section. In the 'Test bot' section, there is a text input field with a microphone icon and the placeholder 'Chat with your bot...'. Below the input field, a button says 'Clear chat history'. The 'Inspect response' panel is visible, showing 'Dialog State: ElicitIntent' and two radio buttons: 'Summary' (selected) and 'Detail'. A blue arrow points from the 'Chat with your bot...' input field to the 'Inspect response' panel. The 'Inspect response' panel also has a 'Hide' link in the top right corner. The 'Inspect response' panel contains the text: 'When you chat with your bot, you can see the fulfillment state of your intent and the response here.'

3. Ask your bot a question defined in utterances, for e.g. "what are your official working hours?" and it replies with the confirmation message that we provided in the intent.

> Test bot (Latest)

Ready. Build complete.

what are your official working hours?

Do you want to know hour working hours?

[Clear chat history](#)

Chat with your bot...

Inspect response

Dialog State: ConfirmIntent

[Hide](#)

Summary Detail

- Once you say, yes, it navigates to fulfilling the intent, but since we did not provide any lambda it says it is ready to fulfillment.

> Test bot (Latest)

Ready. Build complete.

Do you want to know hour working hours?

yes

Intent OperatingHours is ReadyForFulfillment:

[Clear chat history](#)

Chat with your bot...

Inspect response

Dialog State: ReadyForFulfillment

[Hide](#)

Summary Detail

- Again, ask for the relevant question.

> Test bot (Latest)  Ready. Build complete.

Intent OperatingHours is
ReadyForFulfillment:

what time do you open?

Do you want to know hour working
hours?

[Clear chat history](#)

 [Chat with your bot...](#)

Inspect response

Dialog State: ConfirmIntent

[Hide](#)

Summary Detail

6. Now type no and we'll see that the bot responds with OK, Thanks! i.e. the text we provided in the confirmation for no.

> Test bot (Latest)  Ready. Build complete.

Do you want to know hour working
hours?

no

OK. Thanks!

[Clear chat history](#)

 [Chat with your bot...](#)

Inspect response

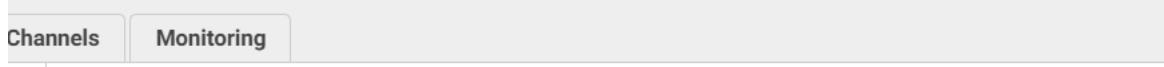
Dialog State: Failed

[Hide](#)

Summary Detail

AWS Lambda Function

Let's enable our bot to use a lambda function i.e the defined business logic that a bot executes for fulfillment i.e. for the user when he types "yes". In the Fulfillment section of the intent, choose AWS Lambda function. This will ask you to provide a lambda function, but since we do not have one, we need to create it before we use it.



- ▶ Sample utterances ⓘ
- ▶ Lambda initialization and validation ⓘ
- ▶ Slots ⓘ
- ▶ Confirmation prompt ⓘ

▼ Fulfillment ⓘ

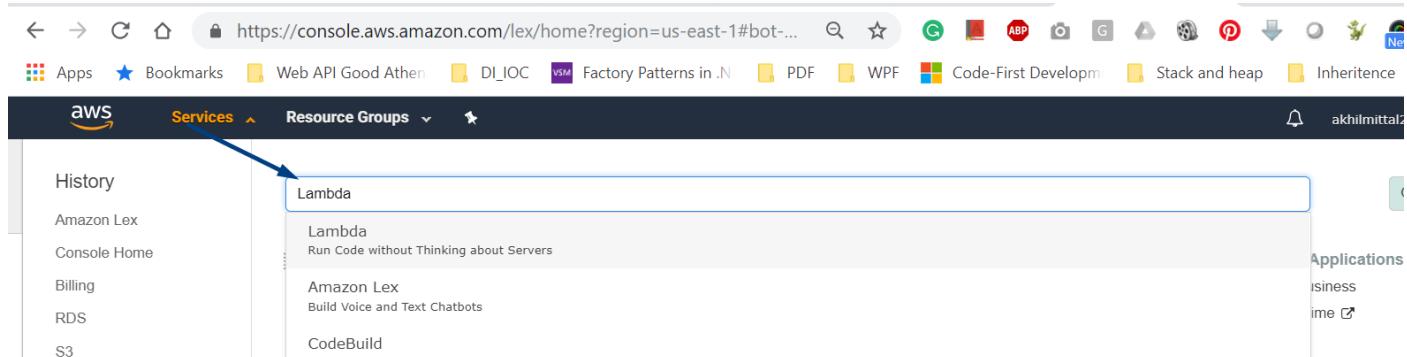
AWS Lambda function Return parameters to client

Lambda function

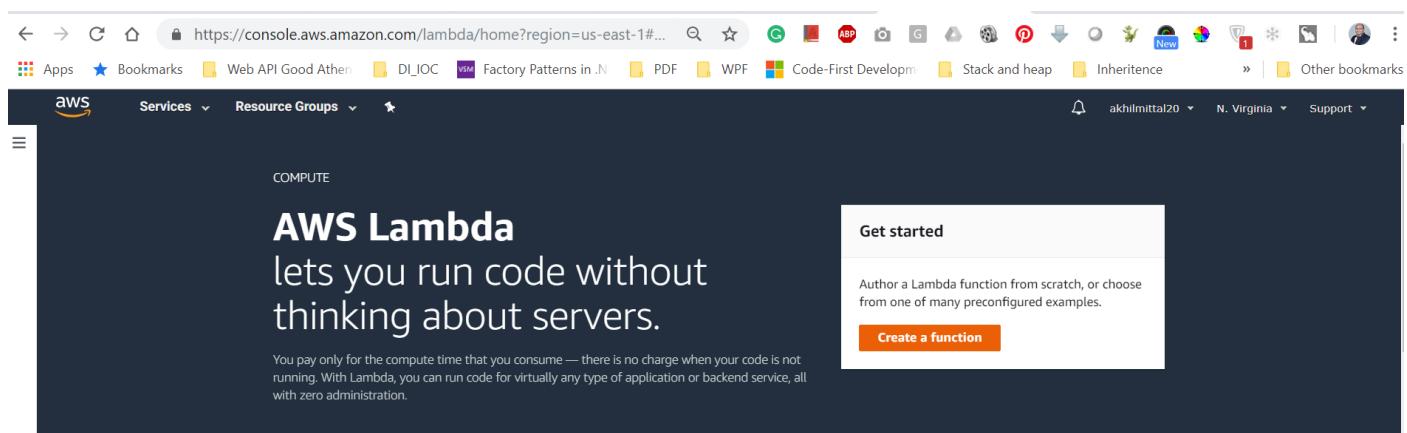
▼ Response ⓘ

+ Add Message

1. Go to Services search option and type Lambda. You'll get the suggestion options, choose the Lambda as shown.



2. You'll be landed to AWS Lambda page. Click on Create a function to create a new lambda function for fulfillment.



3. Select the option to author from scratch as shown below.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Author from scratch' option is selected, indicated by a blue border and a filled circle. Other options like 'Blueprints' and 'AWS Serverless Application Repository' are shown with unfilled circles. The 'Author from scratch' section contains fields for 'Name' (set to 'myFunctionName'), 'Runtime' (set to 'Node.js 8.10'), and 'Role' (set to 'Create a new role from one or more templates'). A note at the bottom states: 'Lambda automatically creates a role with permissions from this selected runtime template. Basic Lambda permissions (such as logging to Amazon CloudWatch) are automatically added if none are specified.'

4. Provide the name to the function. You can choose any runtime you are comfortable with to write the code. I am choosing here Python 3.7 to write simple python code and return the result.

This screenshot shows the same 'Create function' wizard as the previous one, but with different input values. The 'Name' field now contains 'WorkingHours'. The 'Runtime' dropdown has been changed to 'Python 3.7'. The 'Role' dropdown is set to 'Choose an existing role'. Under 'Existing role', the dropdown shows 'service-role/simpleBotService'. The 'Create function' button is visible at the bottom right.

5. Click on Create function and you'll see the option to write the code. Choose "Edit code inline". The Handler details are pre-populated i.e. "**function name.handler name**". Write the function to return the text as "Our working hours are 9 AM to 6 PM IST on weekdays" as shown below.

The screenshot shows the AWS Lambda Function Editor for a function named "WorkingHours". The "Function code" tab is selected. The code entry type is set to "Edit code inline". The runtime is Python 3.7, and the handler is "lambda_function.lambda_handler". The code itself is a Lambda function that returns a JSON response with a fulfillment state of "Fulfilled" and a message indicating working hours from 9 AM to 6 PM IST on weekdays.

```
import json

def lambda_handler(event, context):
    response = {
        'dialogAction': {
            'type': 'Close',
            'fulfillmentState': 'Fulfilled',
            'message': {
                'contentType': 'PlainText',
                'content': 'Our working hours are 9 AM to 6 PM IST on Weekdays'
            }
        }
    }
    return response
```

6. Test the function implementation by clicking the Test button and if everything is fine the test execution will show the result as succeeded.

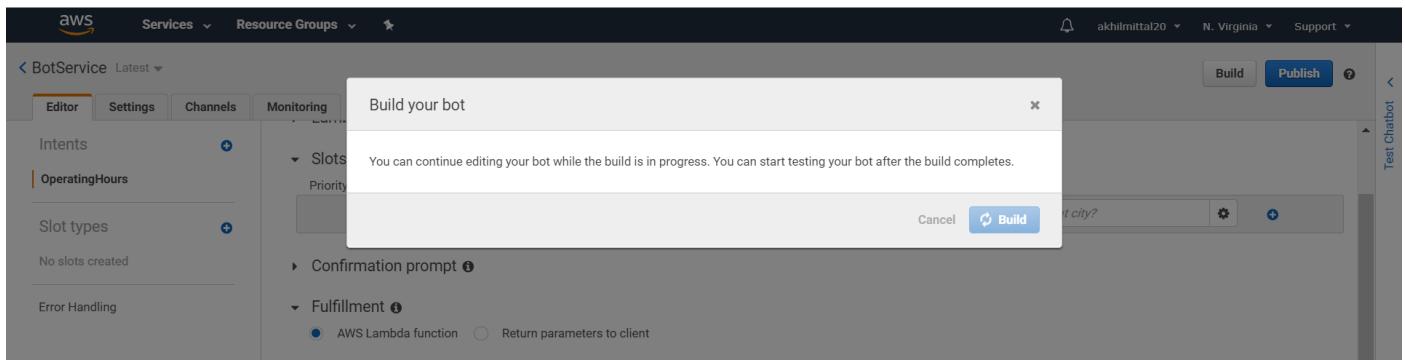
The screenshot shows the AWS Lambda console. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and user information ('akhilmittal20', 'N. Virginia', 'Support'). Below the navigation is a breadcrumb trail: 'Lambda > Functions > WorkingHours'. To the right of the trail is the ARN: 'arn:aws:lambda:us-east-1:830121621230:function:WorkingHours'. The main area is titled 'WorkingHours'. On the right side of this title are several buttons: 'Throttle', 'Qualifiers ▾', 'Actions ▾', 'TestHours ▾', 'Test' (which is highlighted with a blue box), and 'Save'. Below the title, a green success message is displayed: 'Execution result: succeeded (logs)'. A 'Details' link is shown below the message. A note says: 'The section below shows the result returned by your function execution.' Below this note is a JSON response:

```
{  
  "dialogAction": {  
    "type": "Close",  
    "fulfillmentState": "Fulfilled",  
    "message": {  
      "contentType": "PlainText",  
      "content": "Our working hours are 9 AM to 6 PM IST on Weekdays"  
    }  
  }  
}
```

- Now go back to the Intent and under fulfillment when AWS Lambda option is chosen, the recently created function is available. Choose that function.

The screenshot shows the AWS Lambda function configuration interface. At the top, there's a navigation bar with 'aws' logo, 'Services', 'Resource Groups', and user information 'akhilmittal20', 'N. Virginia', and 'Support'. Below the navigation, the page title is 'BotService Latest'. The main content area has tabs 'Editor' (selected), 'Settings', 'Channels', and 'Monitoring'. On the left, a sidebar lists 'Intents' (OperatingHours selected), 'Slot types' (No slots created), and 'Error Handling'. The main panel starts with a section for 'Slots': Priority (e.g. Location), Required (e.g. AMAZON.US_CITY), Name (e.g. What city?), Slot type (dropdown), Version (dropdown), and Prompt (e.g. What city?). Below this are sections for 'Confirmation prompt', 'Fulfillment' (selected AWS Lambda function), and 'Response'.

- ## 8. Again, build your bot.

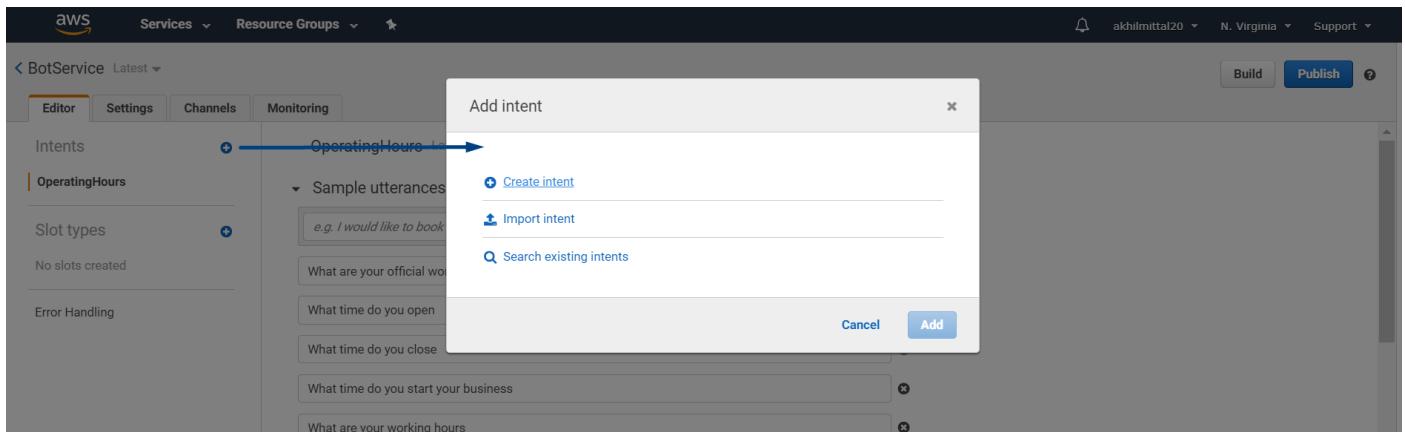


9. Now when you test the bot and after the confirmation of the bot if you type yes that means the fulfillment needs to be called and the bot here returns the text that we provided as a return text in the lambda function. So, our fulfillment and the lambda function work fine.

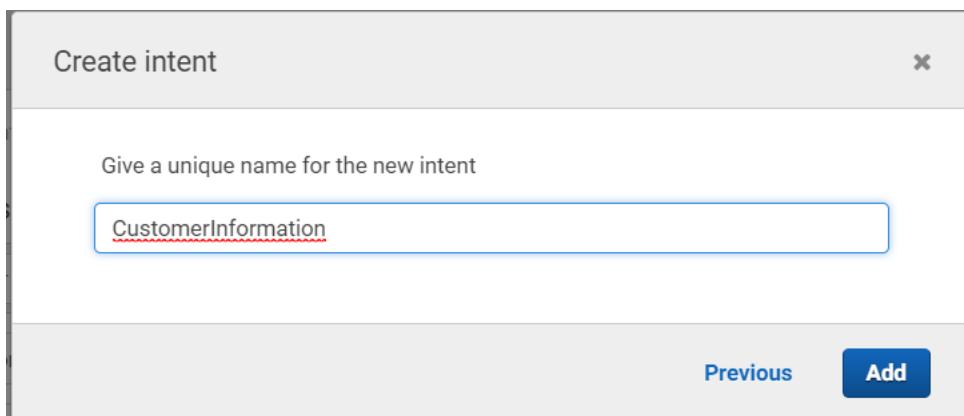
Slots

In this section, we'll see how to collect customer information using Slots.

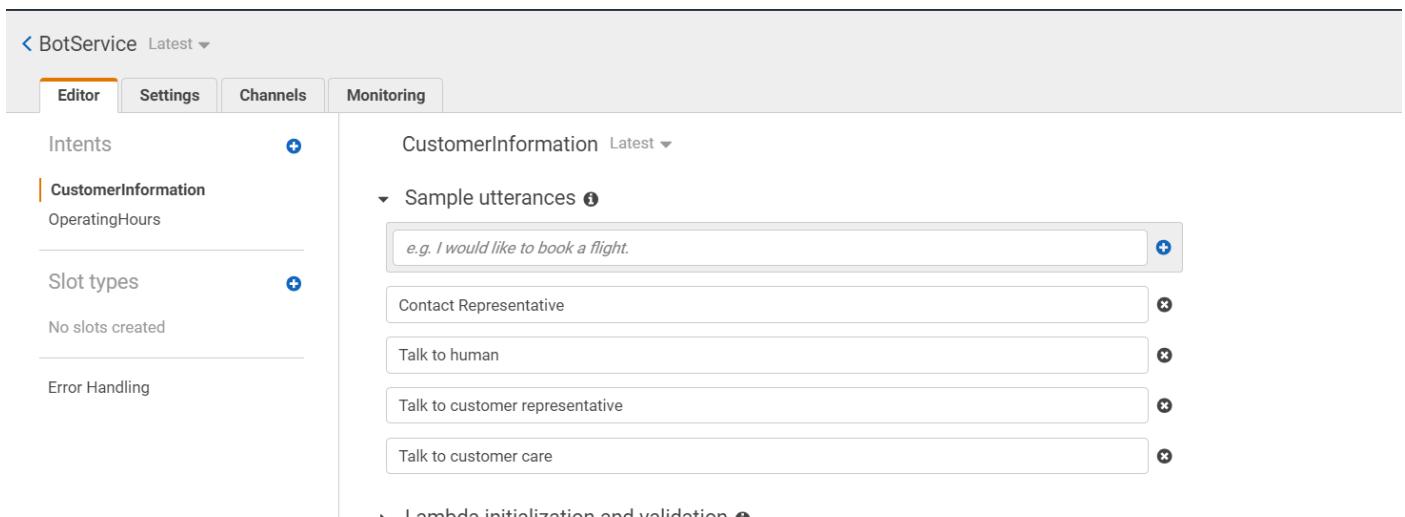
1. You can add a new intent to prompt a customer to speak to the representative.



2. Add the new intent and name it CustomerInformation.



3. You can provide the sample utterances like "Contact Representative", "Talk to human" etc.



Coming down to slots section, provide the name of the slot, the type and the prompt. Lex service provides various types of slots, we'll use those.

4. Choose slot Name as name, choose slot type as AMAZON.person and prompt as "What is your name". You can explore more on slot types by checking in the dropdown on what information you need to seek from the customer.

Talk to customer representative

Talk to customer care

- ▶ Lambda initialization and validation ⓘ
- ▼ Slots ⓘ

Priority	Required	Name	Slot type	Version	Prompt
		name	AMAZON.Person		What is your name? ⚙️ +
- ▶ Confirmation prompt ⓘ

We'll add few more slots like phone number, email id, preferred date of calling to the customer, preferred time of calling to the customer. Check the following picture for the implementation.

- ▼ Slots ⓘ

Priority	Required	Name	Slot type	Version	Prompt
		e.g. Location	e.g. AMAZON.US_CITY		e.g. What city? ⚙️ +
1.	✓	name	AMAZON.Person	Built-in	What is your name? ⚙️ ✎
2.	✓	phononenumber	AMAZON.NUMBER	Built-in	What is your phone number? ⚙️ ✎
3.	✗	emailaddress	AMAZON.EmailAddress	Built-in	What is your email address? ⚙️ ✎
4.	✗	Date	AMAZON.DATE	Built-in	Preferred Date? ⚙️ ✎
5.	✗	Time	AMAZON.TIME	Built-in	Preferred time? ⚙️ ✎
- ▼ Confirmation prompt ⓘ

Chatbot collects all the information from the user when the user intents to speak to the customer. You can check the Required checkbox to seek all the mandatory information.

- Now add the confirmation prompt to make sure that all the information provided by the customer is correct. In the Confirmation prompt, you can reference Slot values by surrounding the Slot name with curly braces. So, we will include name, phone number, Date, and Time in our Confirmation prompt. If the user decides they don't want us to call them and says no at the confirmation, we will just say "OK. Thanks!".
- Now we can save, build, and test our bot.

The screenshot shows the AWS BotService Editor interface. On the left, there's a sidebar with tabs for Intents, CustomerInformation, Slot types, and Error Handling. The main area has tabs for Editor, Settings, Channels, and Monitoring. A modal window titled "Build your bot" is open in the center. It contains a message: "You can continue editing your bot while the build is in progress. You can start testing your bot after the build completes." Below this is a "Build" button. To the right of the modal, there's a list of slots with their types and prompts:

What is your name?	⚙️
What is your phone number?	⚙️
What is your email address?	⚙️
Preferred Date?	⚙️
Preferred time?	⚙️

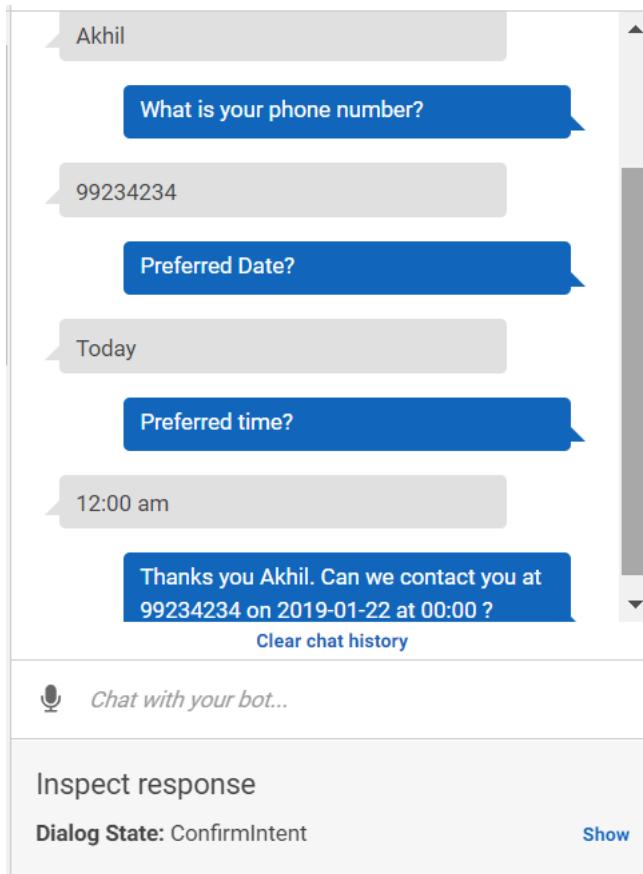
At the bottom of the modal, there are sections for "Confirmation prompt" and "Fulfillment".

Testing Slots

After the bot is built again. You can test the intent and the slots again to check if the bot is working fine.

So, when you type the sample utterances phrase like “Speck to a representative”, the bot starts collecting information as asked in the slot.

It asks, “What is your name?”, “What is your phone number” etc whatever was provided in the slot.



Once all the information is provided, it shows the confirmation prompt with the customer details that we provided in the prompt message for confirmation in curly braces.

Today

Preferred time?

12:00 am

Thanks you Akhil. Can we contact you at
99234234 on 2019-01-22 at 00:00 ?

yeah

Intent CustomerInformation is
ReadyForFulfillment: Date:2019-01-22
Time:00:00 emailaddress:null name:Akhil
phonenumer:99234234

[Clear chat history](#)

 Chat with your bot...

Inspect response

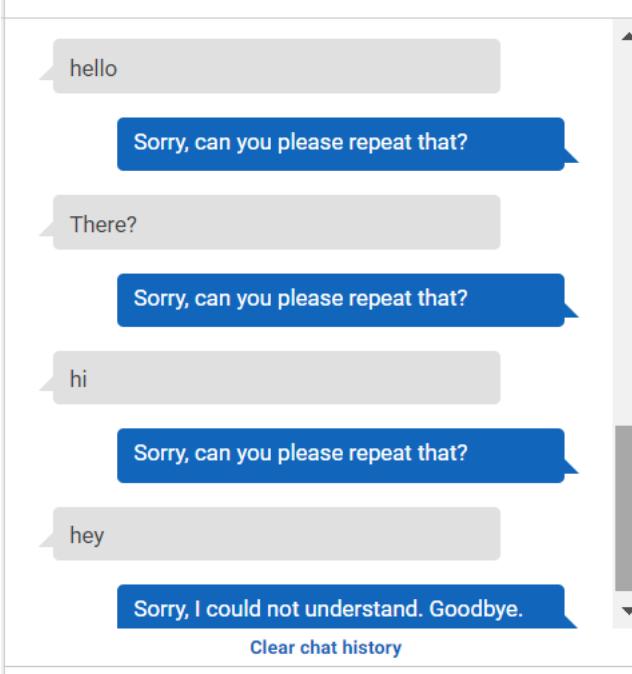
Dialog State: ReadyForFulfillment

[Show](#)

Error Handling

We saw that our bot does not respond well to the user saying “Hi” or “Hello” or something that is not mentioned in the intent utterances and it keeps on showing the defaulted message and after few prompts, it says that it does not understand what user is saying and goodbye. This is all configured for the bot as the default setting. We can change that as per our requirement using the Error Handling feature of Lex.

> Test bot (Latest) Ready. Build complete.



Microphone icon | Chat with your bot...

Inspect response

Dialog State: Failed

Show

1. Click on the orange color link saying Error Handling on the BotService.

The screenshot shows the AWS BotService configuration page for the "CustomerInformation" intent:

- Intents**: CustomerInformation (highlighted in orange), OperatingHours
- Slot types**: No slots created
- Error Handling** (highlighted in orange)

CustomerInformation Latest

- Sample utterances**:
 - e.g. I would like to book a flight.
 - Contact Representative
 - Talk to human
 - Talk to customer representative
 - Talk to customer care
- Lambda initialization and validation**
- Slots**

2. In the clarification prompt, give the prompt on what your bot is capable of so that it shows the meaningful information to the user if user types in anything that bot do not understand. You can limit a maximum number of tries as your leisure. I am limiting it to 2. After the two tries if still, user does not ask what the bot actually understands then give a meaningful hang-up phrase that user can call the customer care at a particular number or something else meaningful. You can delete the defaulted clarification prompt and hang-up phrase now.

The screenshot shows the 'BotService Latest' interface with the 'Editor' tab selected. On the left sidebar, under 'Intents', there are entries for 'CustomerInformation' and 'OperatingHours'. Under 'Slot types', it says 'No slots created'. A box highlights the 'Error Handling' section. The main area displays 'Error handling' settings. It includes a 'Clarification prompts' section with a placeholder 'e.g. Sorry, can you please repeat that?' and a text input field containing 'I can help you with our operating hours or talk to representative.' A blue box highlights this input field. Below it is a 'Maximum number of retries' field set to '2'. The 'Hang-up phrase' section contains a placeholder 'e.g. Sorry, I could not understand. Please contact customer support.' and a text input field with the message 'Sorry, I do not understand that. Please call us at 800-110-8112'. A blue box highlights this input field. At the bottom is a 'Save' button.

3. Now test the bot and you see that when we type phrases like “hi” or something else the bot prompts for two times on what it can help with and after that shows the hang-up phrase.

> Test bot (Latest)

Ready. Build complete.

hi

I can help you with our operating hours or talk to representative.

hey

I can help you with our operating hours or talk to representative.

There?

Sorry, I do not understand that. Please call us at 800-110-8112

[Clear chat history](#)

 Chat with your bot...

Inspect response

Dialog State: Failed

[Show](#)

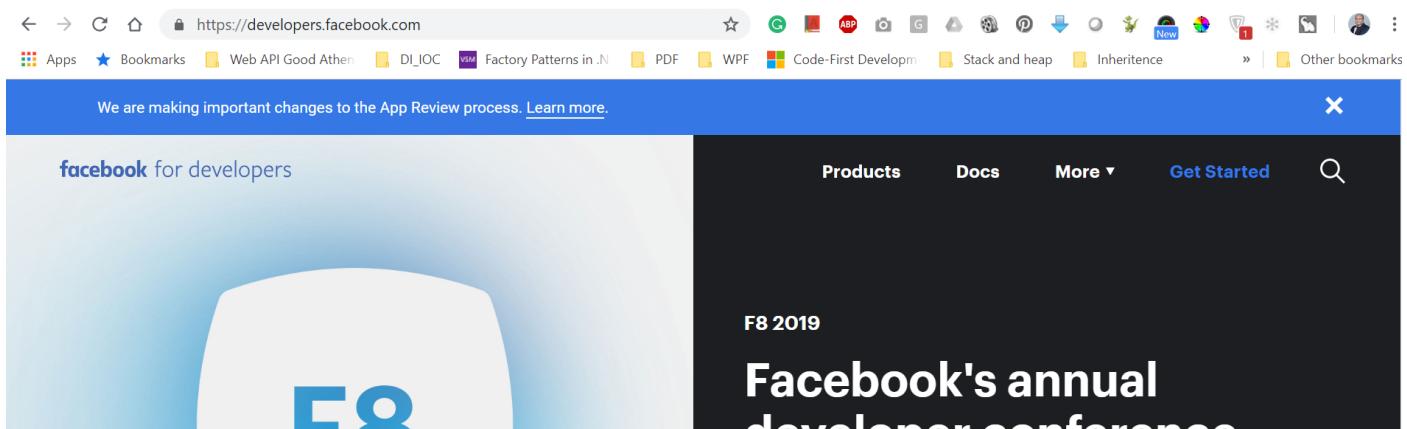
Deploy the Chatbot in Facebook Messenger

Our initial basic bot is completed, you can enhance it more as per your will and requirement. In this section, we'll see how to deploy the bot on Facebook messenger as a chatbot to your custom Facebook page.

So, the need here would be that suppose you have a Facebook page for your business or company and you need to deploy a feature of auto chatbot in that page so that any user can seek information or talk to that bot.

First things first, for deployment of the bot to the Facebook page, we need to have a Facebook page.

1. Open the URL <https://developer.facebook.com> and sign in with your existing Facebook credentials or create a new account



2. I am signing in with my existing Facebook account. It will ask to create a Facebook developers account, and yes we need to, so click Next.
-

1 Register —— 2 First App —— 3 Tell Us About You

Welcome to Facebook for Developers



Create a Facebook for Developers account

Next

By proceeding, you agree to receive marketing related electronic communications from Facebook, including news, events, updates, and promotional emails. You may unsubscribe at any time in Developer Settings.

3. Provide the App Name (whatever you want) and contact email and click Next.
-

1 Register —— 2 First App —— 3 Tell Us About You

Welcome to Facebook for Developers

Lets get started with your first app

App Name
Akhil's ChatBot

You can change the name now or later

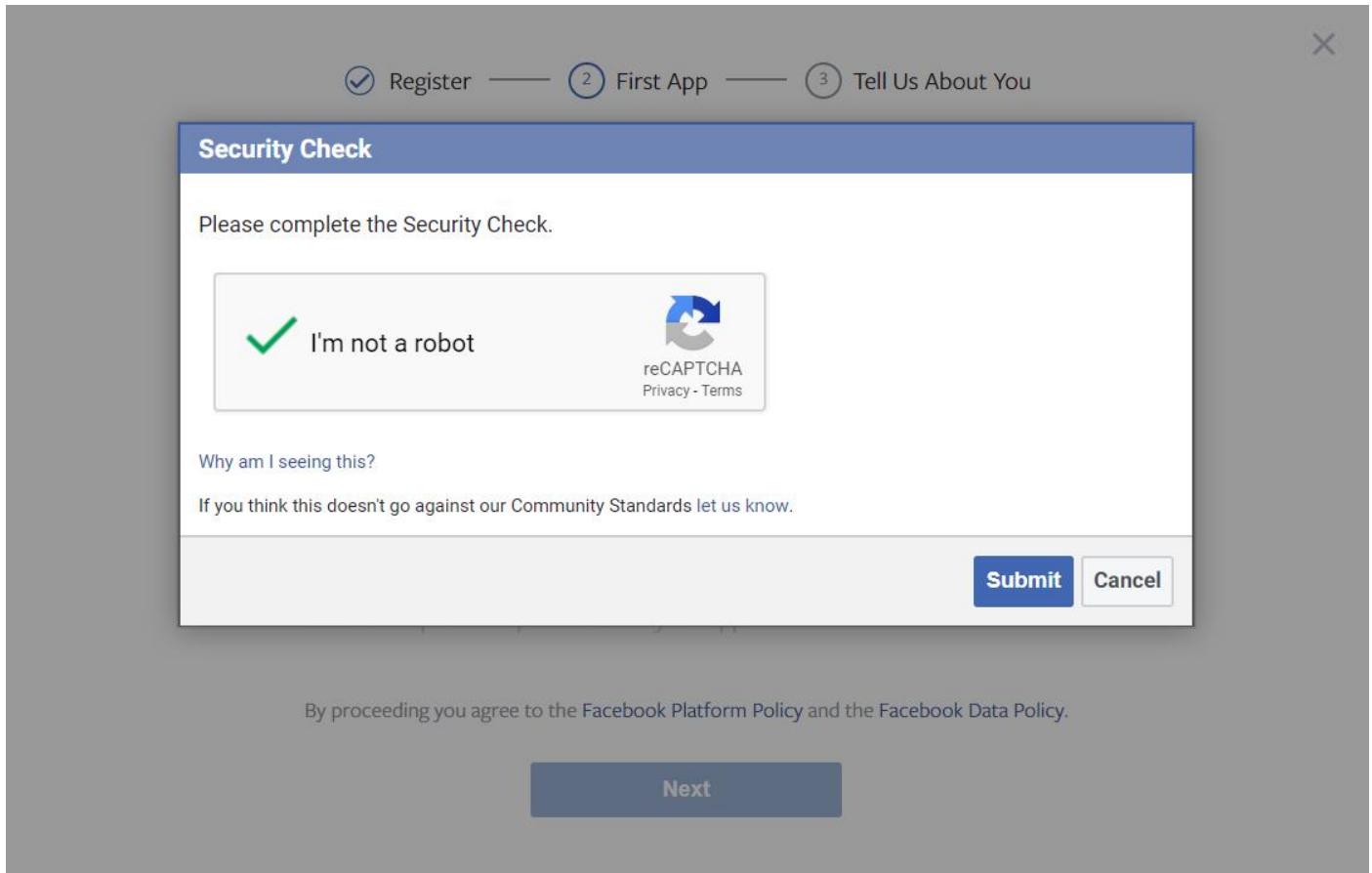
Contact Email
akhil.mittal20@gmail.com

Used for important updates about your app

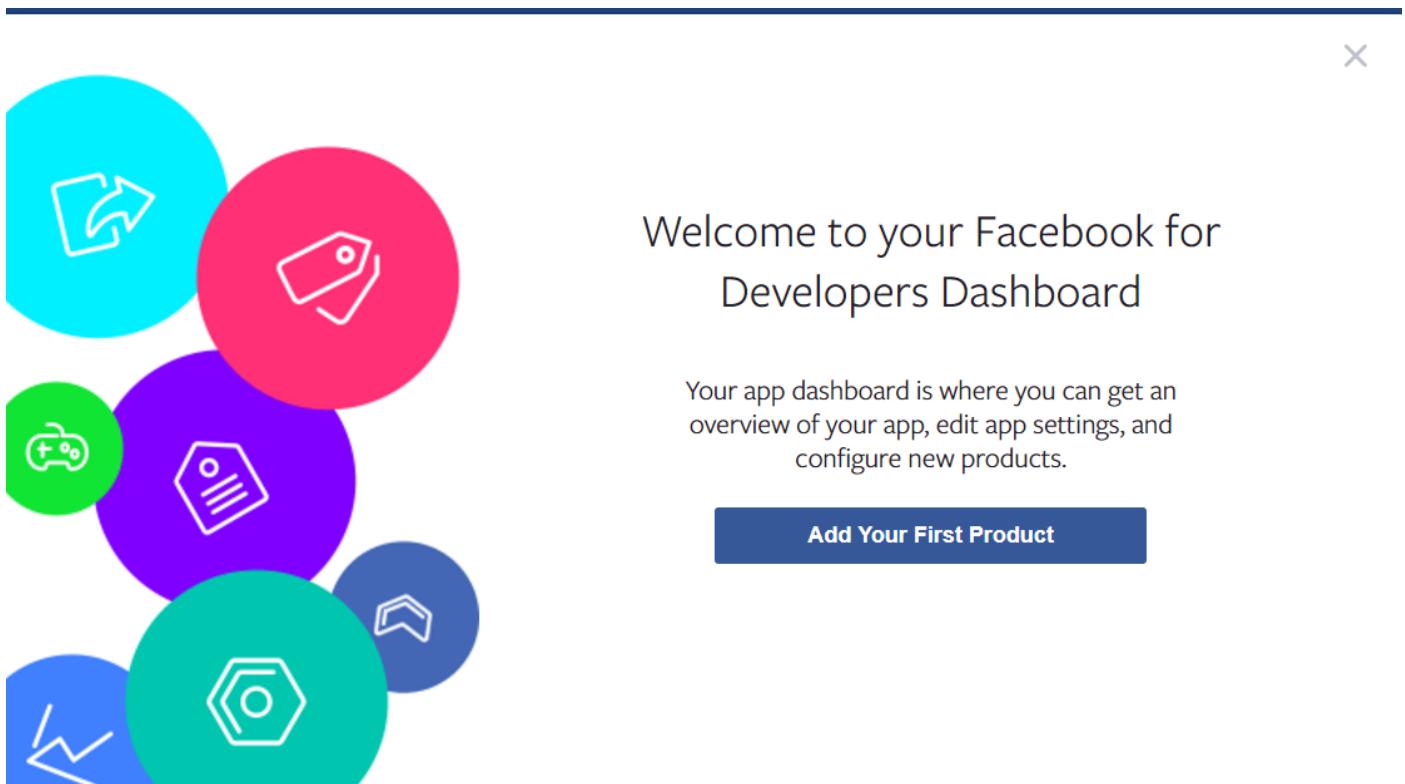
By proceeding you agree to the Facebook Platform Policy and the Facebook Data Policy.

Next

4. Pass the security check and click Submit.



5. It will show the welcome message and ask to add your first product. Click on “Add Your First Product” button.



6. The next screen you'll see is the Facebook developer's dashboard. Since we need to set up the Facebook Messenger, click on the button to setup messenger as shown below.

The screenshot shows the Facebook Developers Dashboard for the app 'Akhil's ChatBot'. The top navigation bar includes 'Docs', 'Tools', 'Support', 'My Apps', and a search bar. The main content area features several sections: 'Analytics' (with a 'Set Up' button), 'Messenger' (with a 'Set Up' button highlighted with a blue border), and other products like 'Page' and 'Webhooks'. A sidebar on the left lists 'Dashboard', 'Settings', 'Roles', 'Alerts', 'App Review', and 'PRODUCTS' (with a plus sign). A right sidebar titled 'Welcome, Akhil!' provides links to 'Get Started', 'Development overview', 'Learn about user permissions', and 'Learn about available features'.

7. You need to have an access token and a secret key for that. So, navigate to Settings and scroll down to the section that says Token Generation. Select an existing page if you have or create a new one as per your wish.

The screenshot shows the 'Settings' page under the 'Messenger' product. The left sidebar includes 'Dashboard', 'Settings', 'Roles', 'Alerts', 'App Review', 'PRODUCTS' (with a plus sign), and 'Messenger' (selected and highlighted with a blue border). The 'Settings' section contains a 'Token Generation' section with instructions: 'Page token is required to start using the APIs. This page token will have all messenger permissions even if your app is not approved to use them yet, though in this case you will be able to message only app admins. You can also generate page tokens for the pages you don't own using Facebook Login.' Below this is a form for generating a page access token, which includes a dropdown menu 'Select a Page' and a note: 'You must select a Page to generate an access token.' A 'Create a new page' button is also present. At the bottom of the page are sections for 'Webhooks' and 'Setup Webhooks'.

8. Select the type of page you want to create i.e. Community or Business.



Business or Brand

Showcase your products and services, spotlight your brand and reach more customers on Facebook.

[Get Started](#)



Community or Public Figure

Connect and share with people in your community, organization, team, group or club.

[Get Started](#)

9. Provide the page name and the category.

Business or Brand

Connect with customers, grow your audience and showcase your products with a free business Page.

Page Name

CodeBusiness

Category

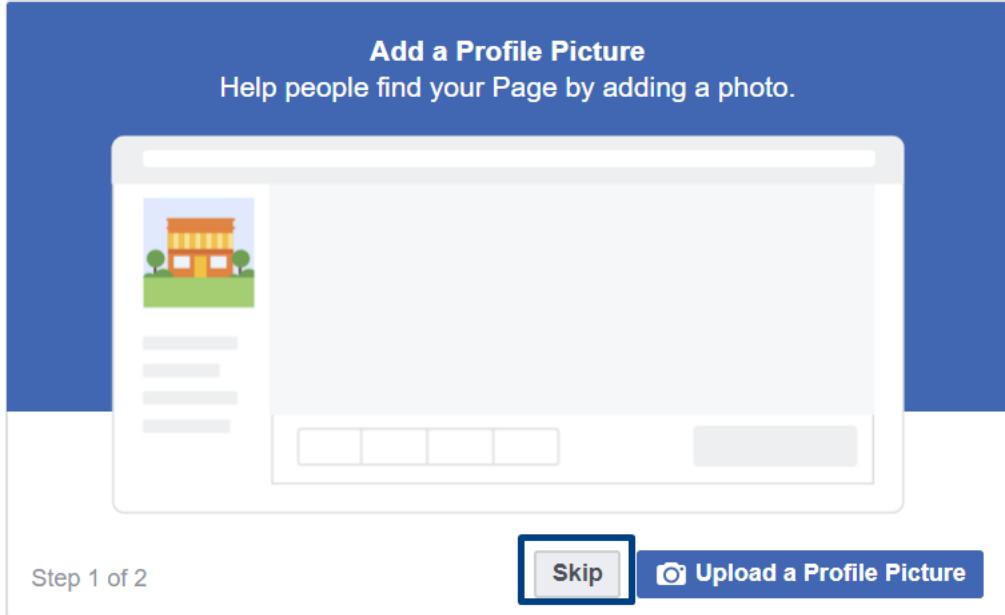
E-commerce Website



When you create a Page on Facebook the [Pages, Groups and Events Policies](#) apply.

[Continue](#)

10. You can skip the steps like providing profile pic or choose to apply one.



11. In the next step, you'll see the page created.

CodeBusiness

Add a Cover

CodeBusiness
Create Page @Username

Home Posts Reviews See more Promote

Welcome to Your New Page

Before you share your Page with others, try these tips that

+ Add a Button

12. Now again go to settings and Token generation section and choose the newly created page.

Token Generation

Page token is required to start using the APIs. This page token will have all messenger permissions even if your app is not approved to use them yet, though in this case you will be able to message only app admins. You can also generate page tokens for the pages you don't own using Facebook Login.

Page Access Token

Select a Page ▾ You must select a Page to generate an access token.

✓ Select a Page

CodeBusiness

A Practical Approach - www.codeteddy.com

Webhooks

Setup Webhooks

13. It will again prompt you for authentication. Do that.

Akhil's ChatBot would like to send messages from Pages you manage, use a user phone number to send messages from Pages you manage and send messages from Pages you manage at any time after the first user interaction.

Choose what you allow

Not Now OK

14. After the authentication, in the Token Generation section, we see now a Page Access token is generated. Copy that token and keep that at some location. We'll soon need that.

Page

CodeBusiness

P Copy Text to Clipboard

EAAEuTCk4568BAJQtBa19DT52jZAT1R4yaxmha2jtkSUpWaFLJlzsEI595Iz9YIZALRzM29yi1liShsiu29nZB5QVuePUObKgb5wz

Create a new page

15. No, go to Settings->Basic to get the App Secret key as shown below. Copy and keep the App Secret key at some safe location, we'll need that soon.

The screenshot shows the AWS Lambda function configuration page for 'Akhil's ChatBot'. The left sidebar has 'Settings' selected under 'Basic'. The main area shows the function configuration with fields like App ID, Display Name, App Domains, Privacy Policy URL, App Secret, Namespace, Contact Email, Terms of Service URL, and a 'Reset' button.

16. Go back to the Amazon Lex service, its time to publish your bot to the Facebook Messenger for the newly created Facebook page. Click on Publish on the BotService.

The screenshot shows the AWS BotService configuration page for 'BotService'. The top navigation bar includes 'Build' and 'Publish' buttons. The main area shows tabs for 'Editor', 'Settings', 'Channels', and 'Monitoring', with 'Editor' currently selected. A 'Publish' button is visible on the right.

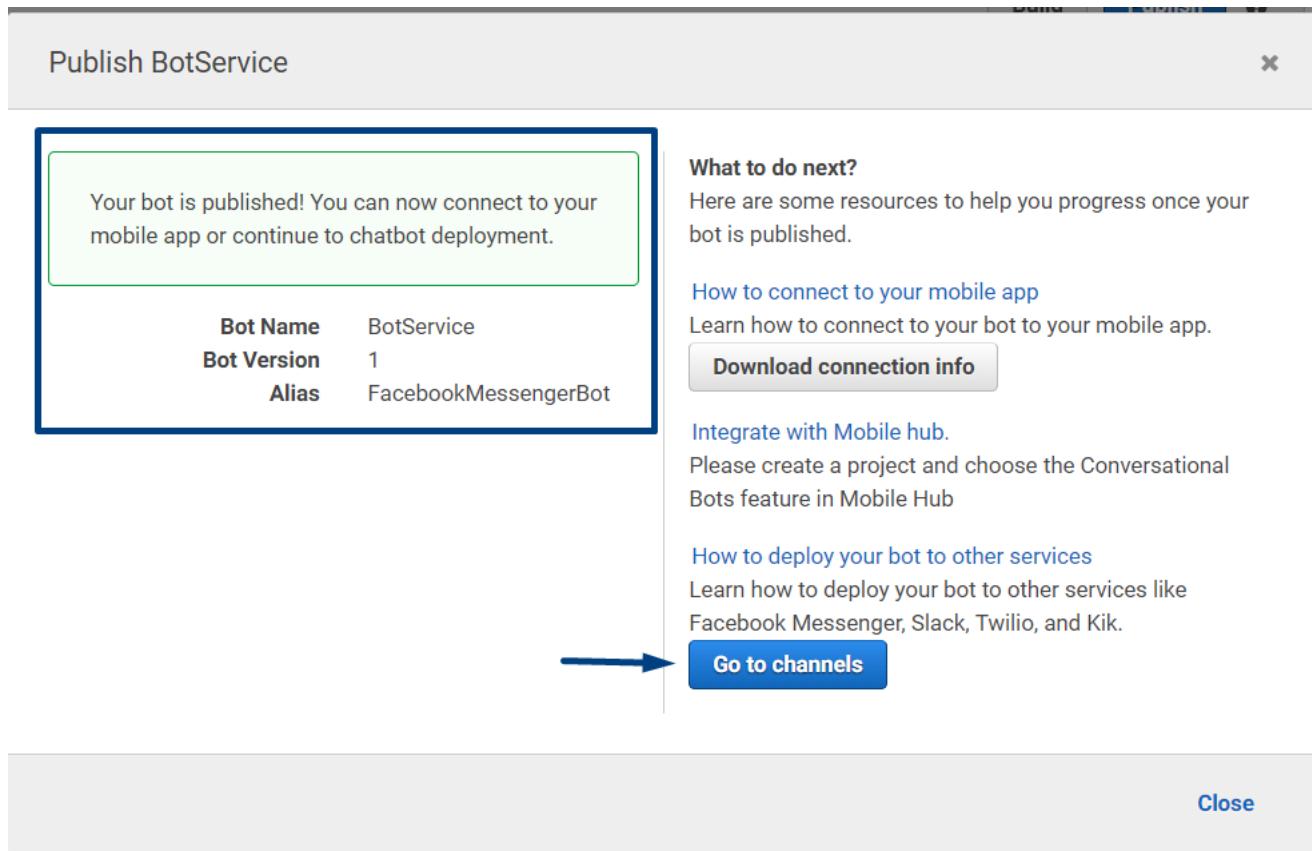
17. It will ask for the alias. Provide the alias name of your choice and remember it or store it somewhere.

The screenshot shows a 'Publish BotService' dialog box. It contains a text input field labeled 'Create an alias' with the value 'FacebookMessengerBot'. Below the input field is a link 'Update an existing alias'. At the bottom right are 'Cancel' and 'Publish' buttons.

18. Click on publish, which may take some time.

The screenshot shows a 'Publish BotService' dialog box with a large circular loading icon. Below the icon is the text 'Publishing. This should take a couple of minutes.' At the bottom right are 'Cancel' and 'Publish' buttons.

19. Once the bot is published, you'll see the confirmation message and a blue colored button which says, "Go to channels".



20. This will show up different channels like Facebook, Kik, Slack, and Twilio. Since we need to integrate that bot to Facebook, choose that. Provide the name of the channel, the alias that we created while publishing. Give a token name and remember it or store it. Provide the saved access token and the API secret key that we stored earlier.

The screenshot shows the configuration for the Facebook channel. On the left, under "Channels", "Facebook" is selected. The main form fields are: Channel Name*: FacebookMessengerBot, Channel Description: (empty), IAM Role: AWSServiceRoleForLexChannels (Automatically created on your behalf), KMS Key: (empty), Alias*: FacebookMessengerBot, Verify Token*: FacebookMessengerBot, Page Access Token*: EAAEuTCk4568BAJQtBa19DT52jZAT1R4yaxmha2jtkSUUpW, App Secret Key*: 7637b85bcd2d411e4b3297ec27ae1f1. Arrows point from the instructions to the corresponding input fields: "Select facebook" points to the Facebook entry in the channel list; "Remember the token name" points to the Verify Token* field; "Provide saved access token" points to the Page Access Token* field; and "Provide saved secret key" points to the App Secret Key* field.

21. Click on Activate after providing all the information on that page.

Alias* FacebookMessengerBot i

Verify Token* FacebookMessengerBot i

Page Access Token* EAAEuTCk4568BAJQtBa19DT52jZAT1R4yaxmha2jtkSUPW i

App Secret Key* 7637b85bcd2d411e4b3297ec27ae1f1 i

Activate

22. Once activated, you'll get the Callback URL at the bottom itself. Copy that URL and save it.

App Secret Key* App Secret Key i

* Required Field

Activate

Callback URLs

FacebookMessengerBot (FacebookMessengerBot)	2019-01-22 23:51:08	Created	Delete
Endpoint URL: https://channels.lex.us-east-1.amazonaws.com/facebook/...			Copy

23. Again, go back to Facebook's developer's account and click Settings. Go to Webhooks section and click Setup Webhooks to provide the webhook to that copied call-back URL in the last step.

facebook for developers

APP ID: 332379620698031

Status: In Development

Dashboard

Settings

Roles

Alerts

App Review

Messenger

Webhooks

Setup Webhooks

Show recent errors

24. In the New Page Subscription i.e. the page opened after clicking setup webhooks, provide the Callback URL and verify token name that we stored earlier. Check the options like messages, postbacks, and optins as shown below. Click Verify and Save.

New Page Subscription

Callback URL

Verify Token

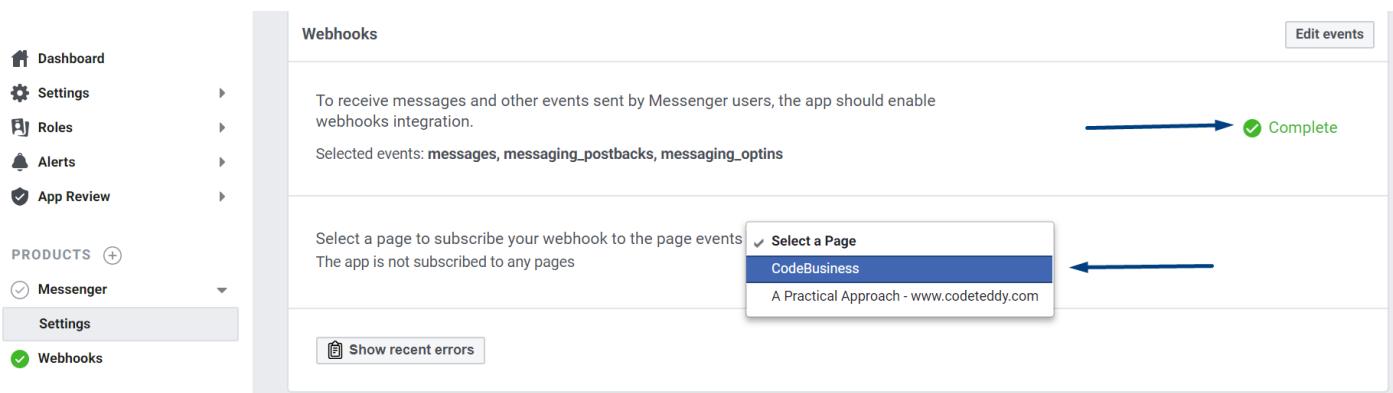
Subscription Fields

<input checked="" type="checkbox"/> messages	<input checked="" type="checkbox"/> messaging_postbacks	<input checked="" type="checkbox"/> messaging_optins
<input type="checkbox"/> message_deliveries	<input type="checkbox"/> message_reads	<input type="checkbox"/> messaging_payments
<input type="checkbox"/> messaging_pre_checkouts	<input type="checkbox"/> messaging_checkout_updates	<input type="checkbox"/> messaging_account_linking
<input type="checkbox"/> messaging_referrals	<input type="checkbox"/> message_echoes	<input type="checkbox"/> messaging_game_plays
<input type="checkbox"/> standby	<input type="checkbox"/> messaging_handovers	<input type="checkbox"/> messaging_policy_enforcement

Learn more

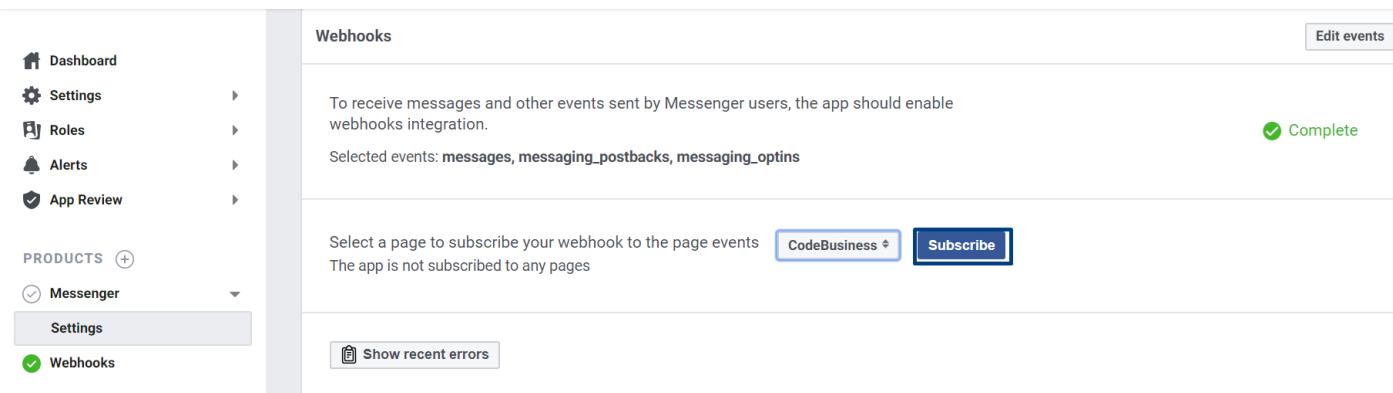
[Cancel](#) **Verify and Save**

25. Once the webhook setup is complete, select the page to subscribe to webhook events. So select your Facebook page there.



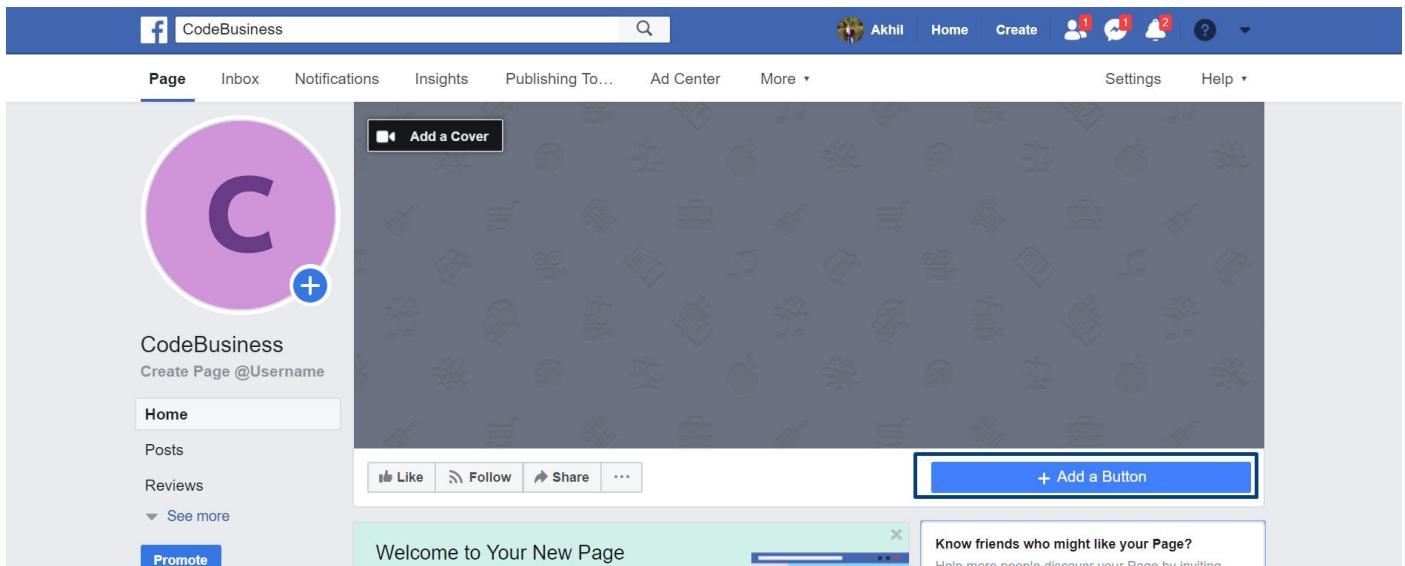
The screenshot shows the Facebook App Dashboard under the 'Webhooks' section. On the left, there's a sidebar with links like Dashboard, Settings, Roles, Alerts, App Review, and sections for Products, Messenger, and Webhooks. The 'Webhooks' section on the right has a heading 'Webhooks' and a note about enabling webhooks integration. It shows 'Selected events: messages, messaging_postbacks, messaging_optins'. Below that, it says 'Select a page to subscribe your webhook to the page events' and 'The app is not subscribed to any pages'. A dropdown menu is open, titled 'Select a Page', showing 'CodeBusiness' as the selected option, with 'A Practical Approach - www.codeteddy.com' listed below it. A green checkmark and the word 'Complete' are visible next to the 'Edit events' button at the top right.

26. Click on the “Subscribe” button.



This screenshot shows the same 'Webhooks' configuration screen as the previous one, but with a focus on the 'Subscribe' button. The 'CodeBusiness' page is now selected in the dropdown menu. The 'Subscribe' button is highlighted with a blue box. The rest of the interface is identical to the previous screenshot, including the 'Edit events' button with a green checkmark.

27. Once done, go back to the Facebook page and add a new button by clicking “Add a button” button as shown below.



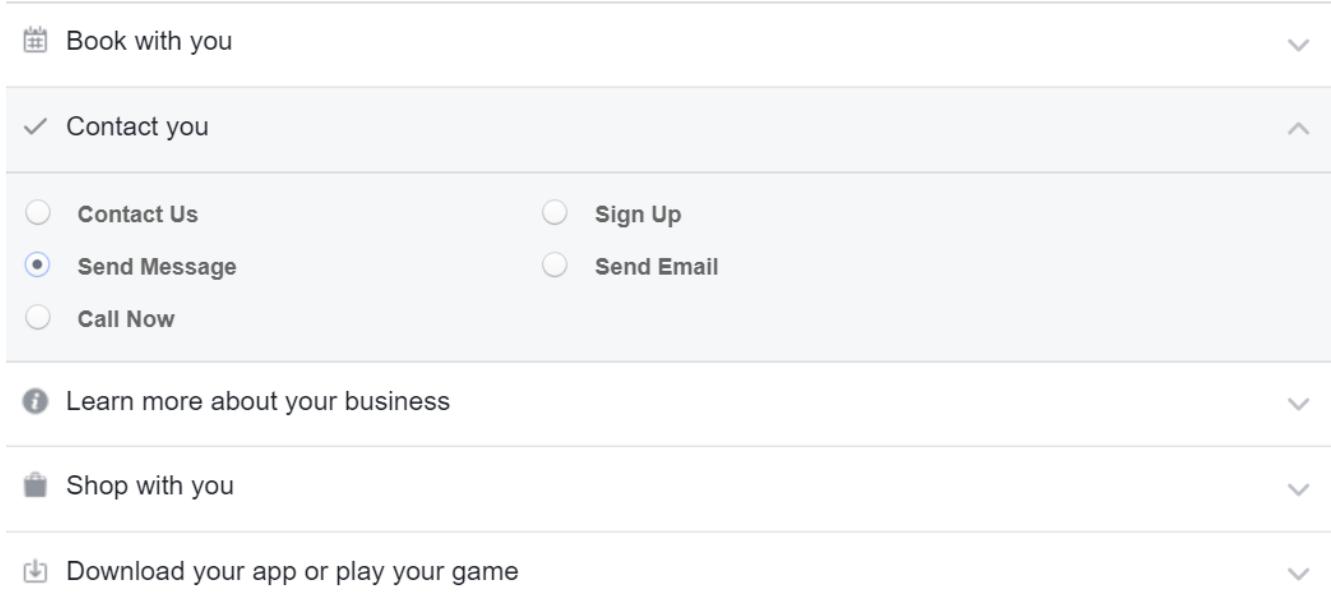
The screenshot shows the Facebook Page settings interface. At the top, there's a navigation bar with 'Page', 'Inbox', 'Notifications', 'Insights', 'Publishing To...', 'Ad Center', 'More', 'Settings', and 'Help'. Below the navigation is a profile picture placeholder with a large letter 'C' and a blue '+' button. The page title is 'CodeBusiness' and the subtext is 'Create Page @Username'. On the left, there are links for 'Home', 'Posts', 'Reviews', and a 'Promote' button. In the center, there are social sharing buttons: 'Like', 'Follow', 'Share', and '...'. A prominent blue button labeled '+ Add a Button' is highlighted with a yellow box. At the bottom, a green banner says 'Welcome to Your New Page' and a white box says 'Know friends who might like your Page?'. There are also small icons for a video camera and a question mark.

28. Provide details for all the steps i.e. selecting Contact you feature for my messenger type. Select the option to send messages.



Step 1: Which button do you want people to see?

The button at the top of your Page helps people take an action. People see it on your Page and in search results when your Page appears. You can edit it any time.



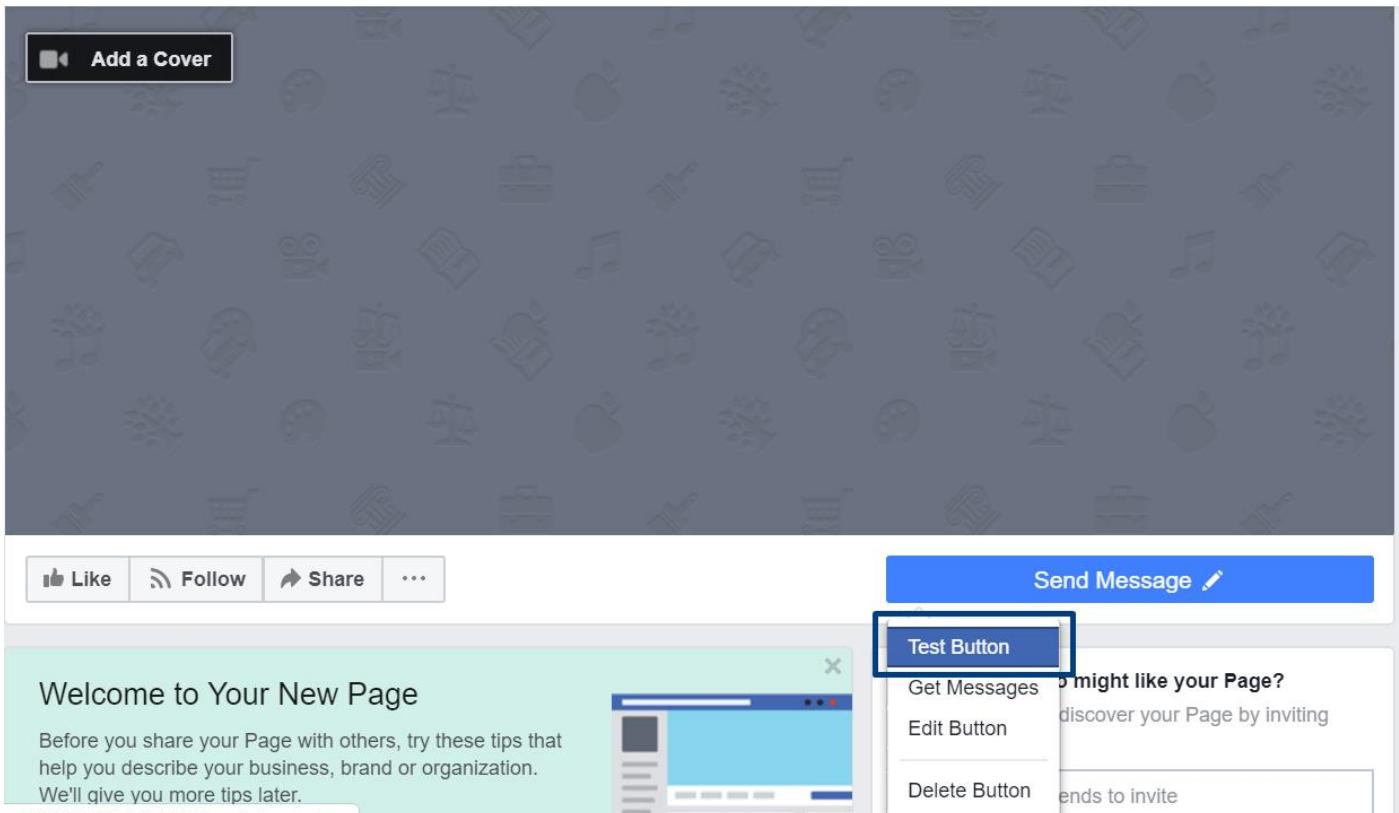
This screenshot shows the 'Buttons' section of a Facebook Page settings page. It lists various button options with their corresponding icons and descriptions:

- Book with you
- Contact you (selected, indicated by a checked checkbox)
- Contact Us
- Send Message (selected, indicated by a checked radio button)
- Call Now
- Learn more about your business
- Shop with you
- Download your app or play your game

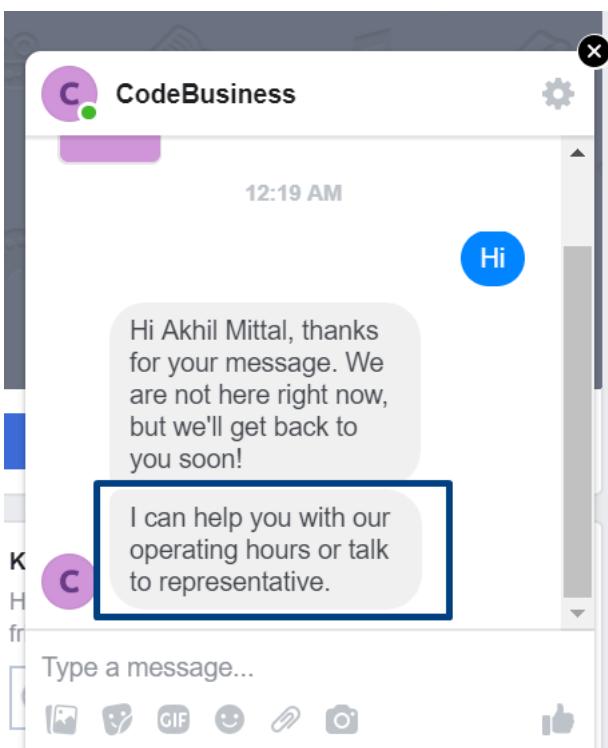
Live Chatbot on Facebook

After performing all the steps in the last section, we can say that our bot is successfully deployed and integrated with our Facebook page. Time to test it.

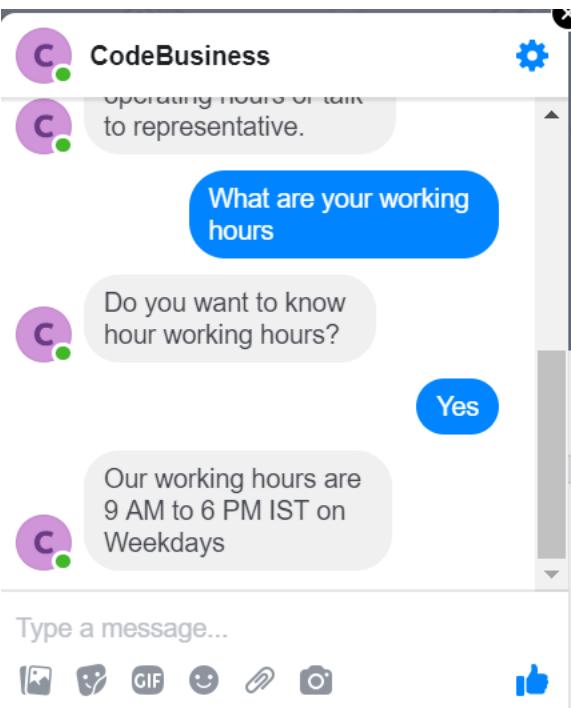
1. Click on newly added “Send Message” button and from the context menu click “Test Button”.



2. As soon as the Test Button link is clicked, the chatbot will open and note that the name of the chatbot is CodeBusiness. You type Hi, and it shows the message we provided in the error handling section.



3. You type the utterances in the first intent and it replies back with what we tested in the Bot Service.



Yippie. Our bot seems to be working well. Like-wise you can test everything that we tested on the BotService on Lex like talking to a human, seeing slot information and all. You can also provide the tester permissions to the user you want to see this bot when they see your page on Facebook. I find it very cool.

Conclusion

In this detailed article, we learned what a chatbot is, how to create your own bot using Amazon's Lex service. Deploy and integrate the bot with the Facebook messenger. The bot integration could also be done to other platforms like Slack, Twilio etc. Just a matter of exploring and playing with it.