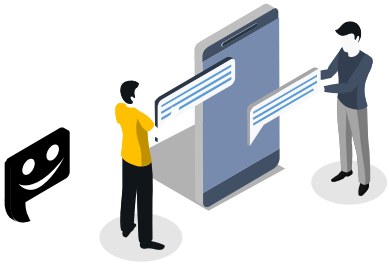
Build a Hotel Reservation Agent

Welcome to the AWS NLP Summit! This is an introductory lab to walk through how to create a simple Hospitability-related chatbot that can provide hotel reservations. It will also show you how to integrate the chat bot with AWS Lambda functions, as well as putting data into DynamoDB, our Amazon native NoSQL database, and lastly integration with Amazon Connect and Facebook Messager.

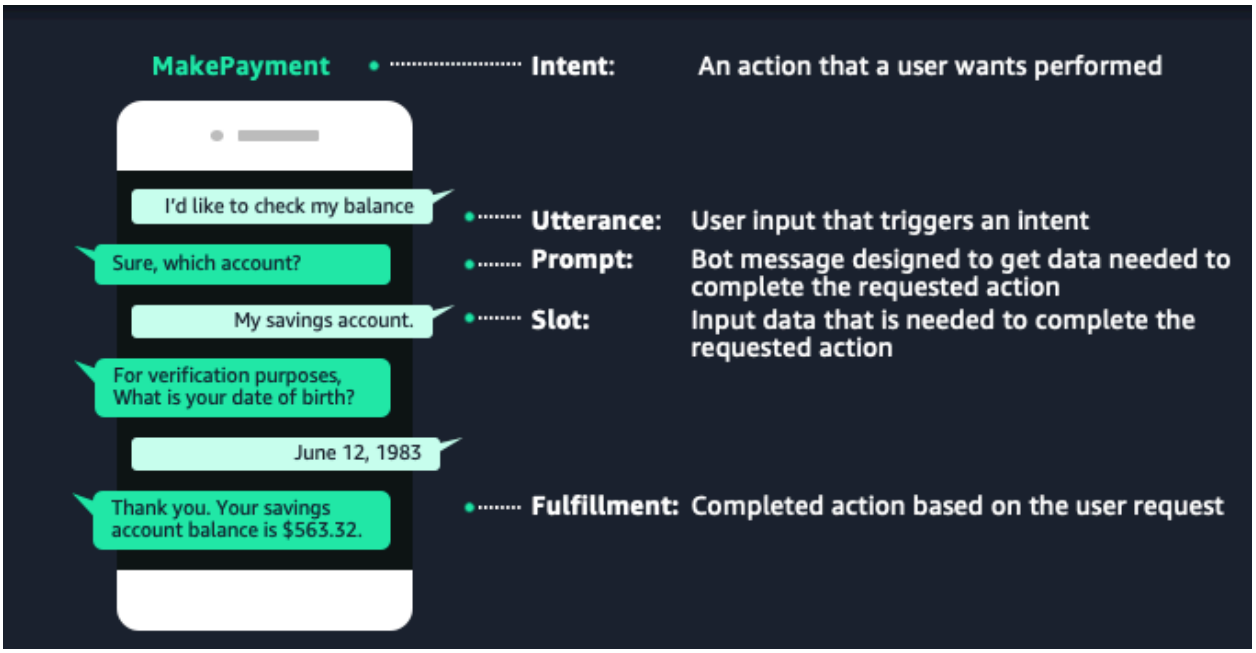


- **Lab Author: Jake Wen, Solutions Architect**
- **Co-Presenter: Michelle Gibbs, Customer Success Manager**

- **Expected duration:** 1.5 hours
- **Target audience:** Developers and Solutions Architects

Overview of Amazon Lex

Chatbot basic concepts:



1. **Intents** - this is an action that the user wants to perform. It enables the bot to understand and classify what task a user is trying to accomplish. A single bot can contain multiple different intents, allowing one bot to handle different - but most-likely related - requests from a user
2. **Utterances** - these are the textual representations of what a user has to type or say in order to trigger an intent. One intent can contain many different utterances, allowing users to trigger the bot using different phrases, potentially providing many different data parameters (or "slots") with their utterance
3. **Prompts** - Amazon Lex can define a different prompt for different stages of the conversation, such as when asking for a bank account number or a date for flight booking. They can also be overridden at run-time by your Lambda functions
4. **Slots** - these are the data items that the bot intent needs in order to be able to fulfill its task; e.g., for a flight booking intent you are likely to need slots for origin airport, destination airport, flight day, flight time, and booking class. Amazon Lex has a large number of built-in slot types, but you are free to create your own custom slot types for your use case
5. **Fulfilment** - this is the final state of the intent, normally with a closing message and some additional information for the user; e.g., for a flight booking you could confirm that the flight is booked, along with a brief summary of the route and the flight number that the user has been booked onto

Amazon Lex allows input using the following methods

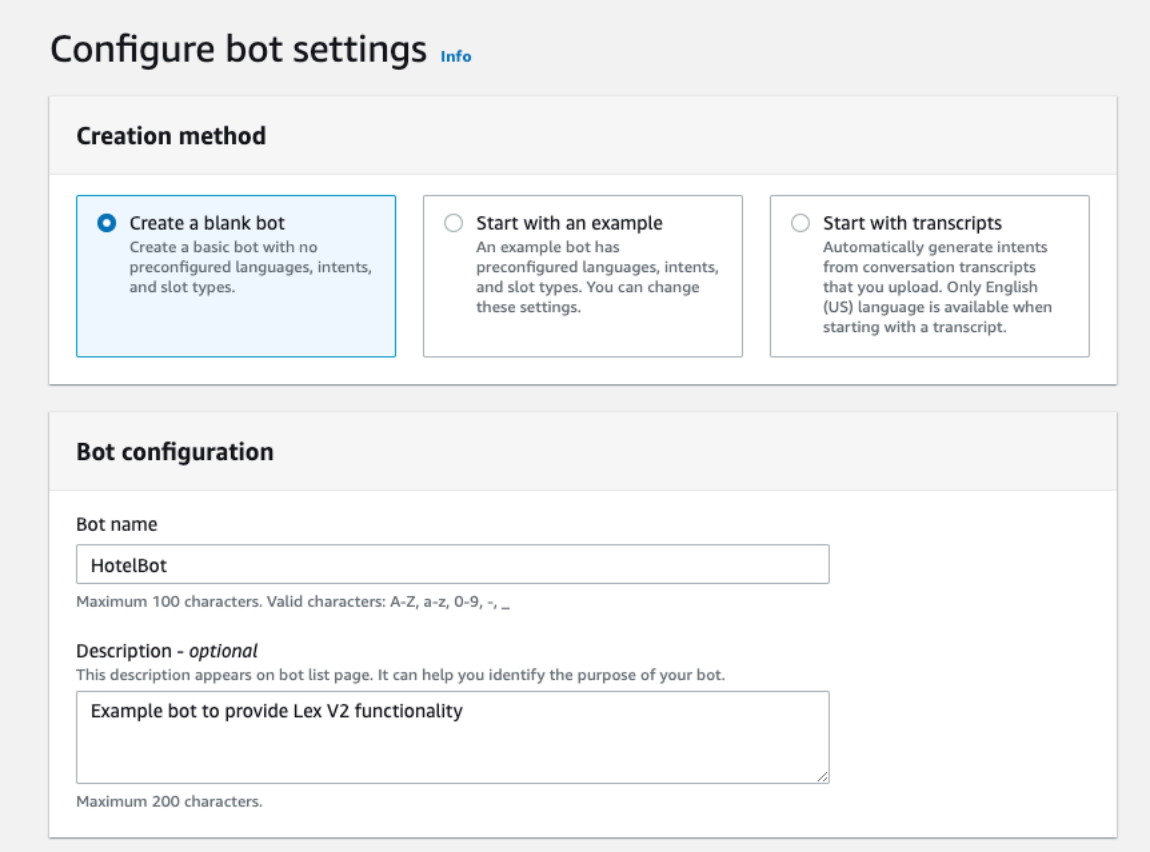
- **Text** - a text string is submitted to Amazon Lex, taken from some client UI such as a Support chat-bot interface
- **Voice** - audio of a supported format is submitted to Amazon Lex

For both of these approaches it is strictly a one-to-one mapping of inputs and outputs - the user submits a request, Amazon Lex will process it and update the session state for that bot, and the result is returned. It is returned in the same fashion - text or voice. Voice responses are based upon an Amazon Polly voice that you have associated with your bot's language configuration. The response could be a request for more intent data, it could be a final fulfilment message, it could be a message dynamically created in your Lambda function or it could be, if the input request wasn't understood, some kind of static or dynamic clarification message.

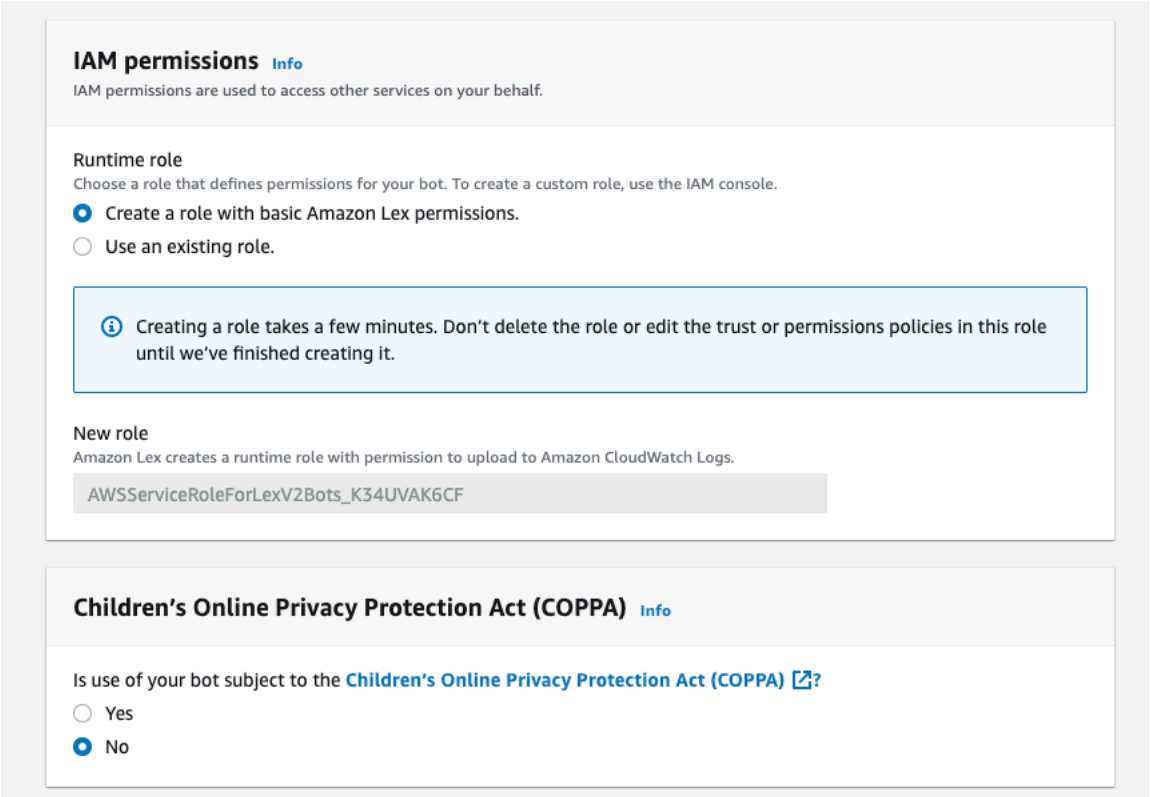
Initial Bot Creation

Create the initial bot:

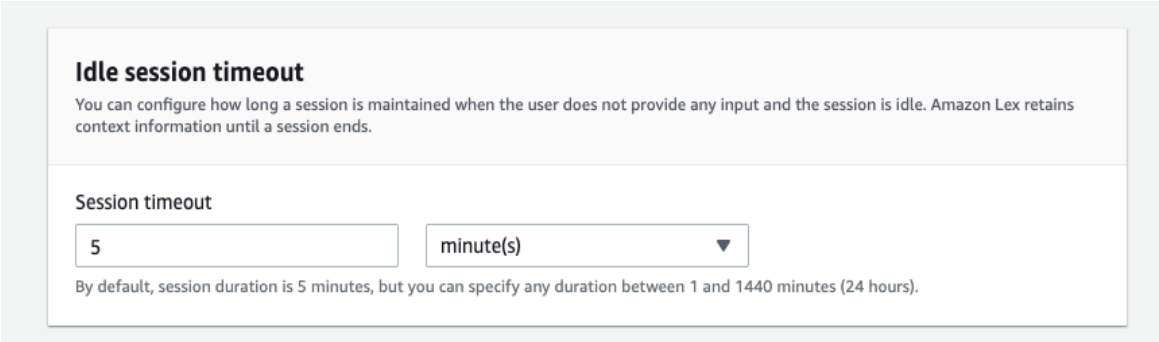
1. Before you login, make sure you follow the lab instruction and create the exact names listed on this document to avoid misconfiguration.
2. Login to the AWS Console using your event engine account and navigate to the Amazon Lex service, which can be done by typing Lex into the top search bar and select the Amazon Lex - Build Voice and Text Chatbots
3. Ensure that you are in the V2 console for Amazon Lex, which is indicated by the version number in the URL - in this example my current AWS Region is set to *N.Virginia (us-east-1)*.
4. Select the **Create bot** button on the top-right corner



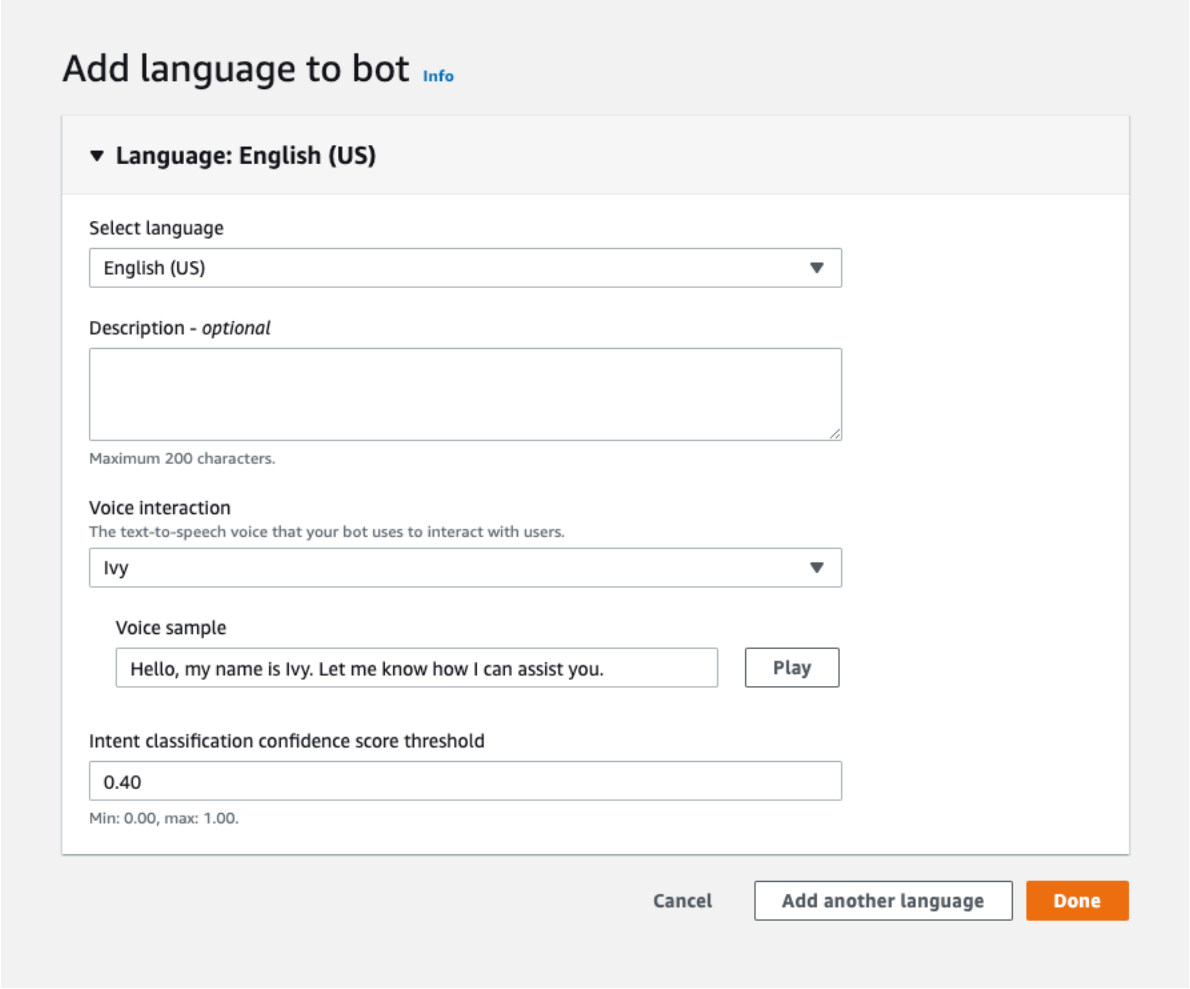
5. Ensure that the **Create a blank bot** option is selected, and type in the bot's name **HotelBot** and an optional description. Note that the bot's name only accepts certain characters.



6. For you **IAM permissions** you can select the ***Create a role with basic Amazon Lex permissions*** option - this will give create an IAM policy that gives Amazon Lex permissions to call other AWS services on your behalf, and it is just what we need for this workshop.
7. This bot has no COPPA implications, so you can just select ***No*** here.



8. Amazon Lex will only maintain a session for a set length of time. If the user is idle and does not provide any input for this length of time then the session will end. For our workshop, we will leave this by default. Click ***Next*** to the next page.



9. You can select your language and your favorite voice here, for this workshop, we will leave it by default and click **Done**.

Create Your First Intent

Overview

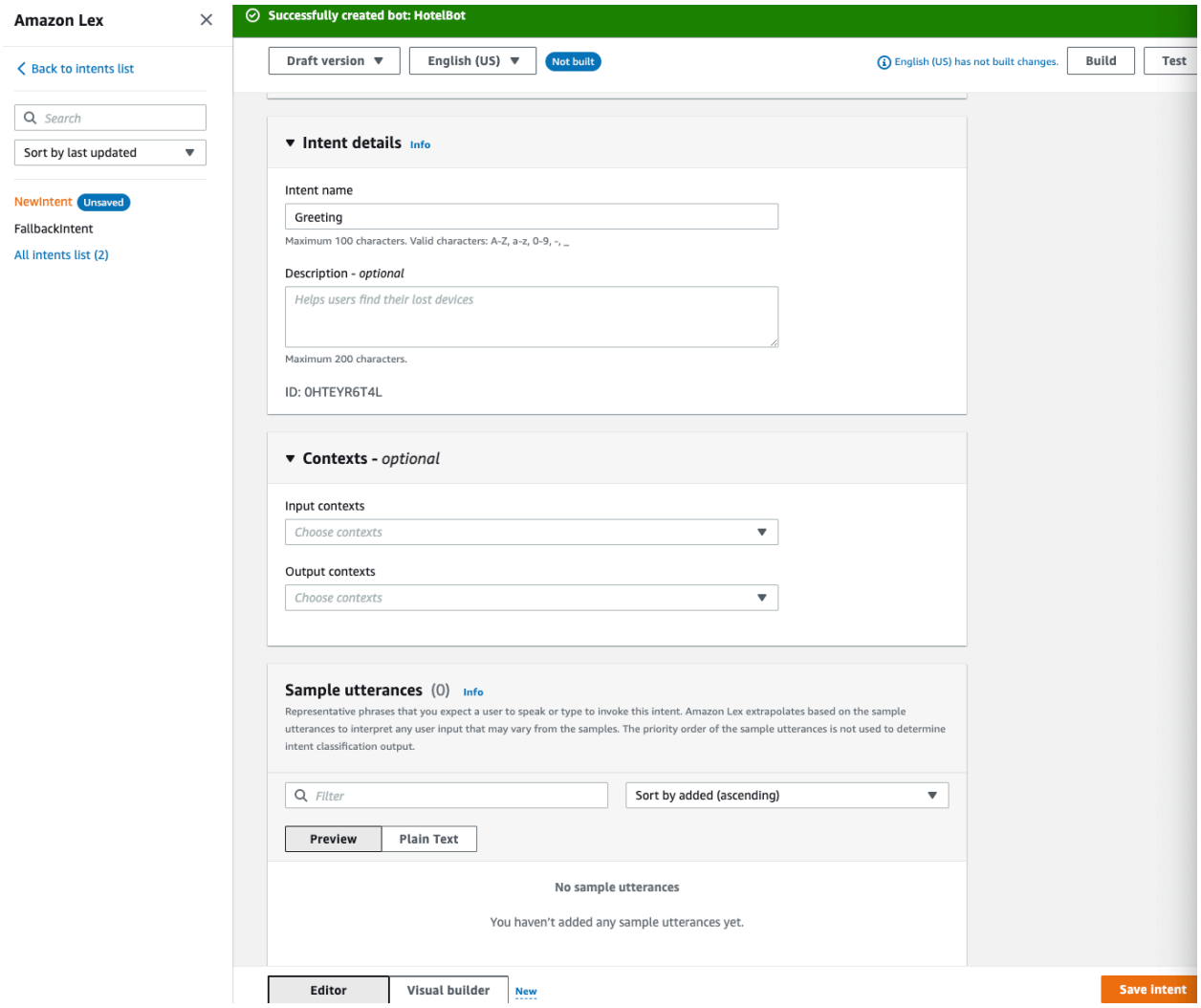
We will work purely in English for this workshop, so we will not need to create any additional languages.

We will start off with the basic **Greeting** intent in this section, along with a simple **FallbackIntent**, and then increase the complexity of the bot by adding:

- Custom slot
- Custom intent
- Lambda function

Creating the Greeting Intent

Now you are in the ***Intents*** screen within Amazon Lex. You will see that the default **FallbackIntent** and **Newintent** has already been created, and that this initial version of the bot is the **Draft version**.



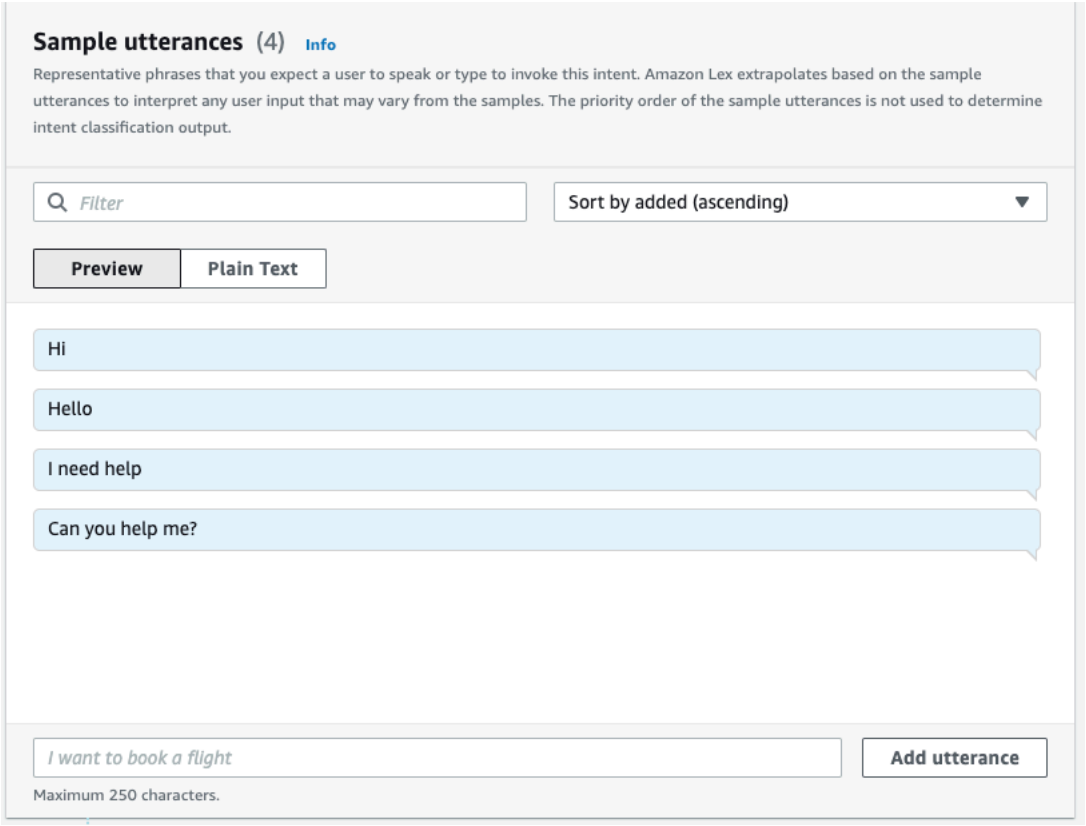
On the **Newintent** section, give it the name **Greeting** and click ***Save Intent***.

This will rename the intent, so at this point add an optional description inside the ***Intent details*** panel, then scroll down to and open the ***Sample utterances*** panel, then click the ***Plain text*** button.

We are now going to add multiple utterances at once - when you are adding multiple utterances that you already have in a list this is the easiest method. Copy the text below, which represent the utterances that will trigger this intent, and paste it into the text window.

Hi
Hello
I need help
Can you help me?

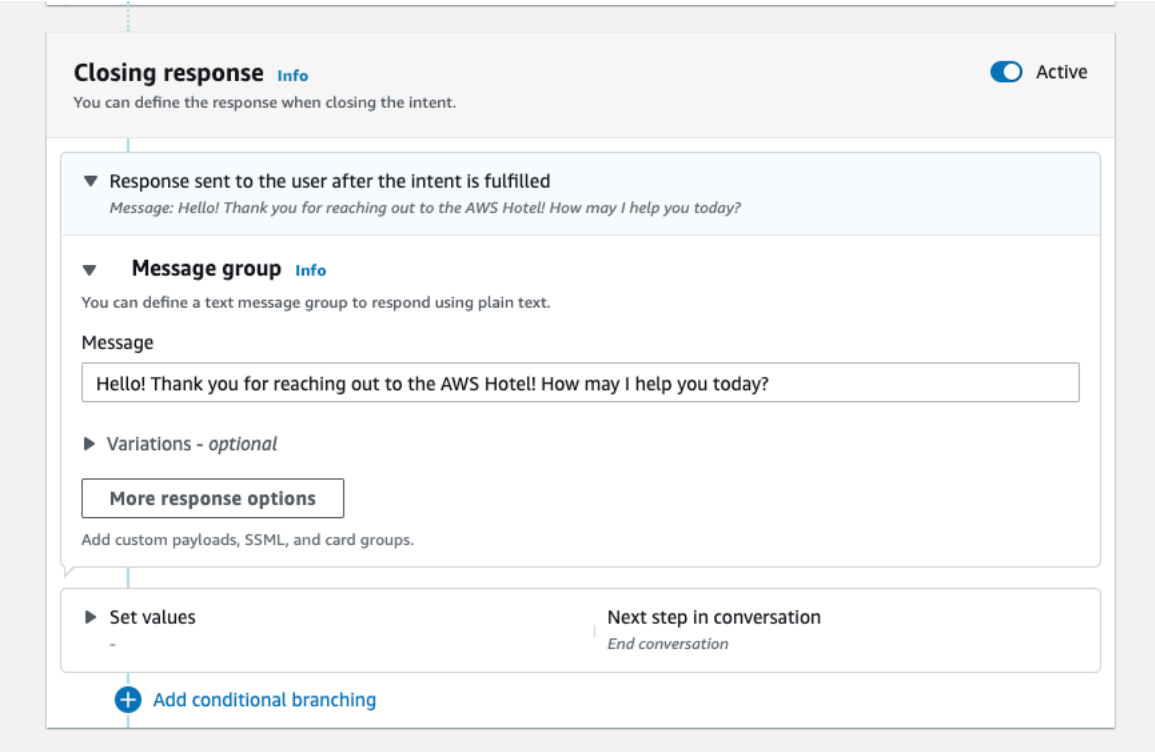
Clicking on the **Preview** button will show these defined as four distinct utterances within this intent:



At this moment we have not defined any **Slots** for this intent because we are not intending to do any processing here, and we just want to open the conversation with a polite response.

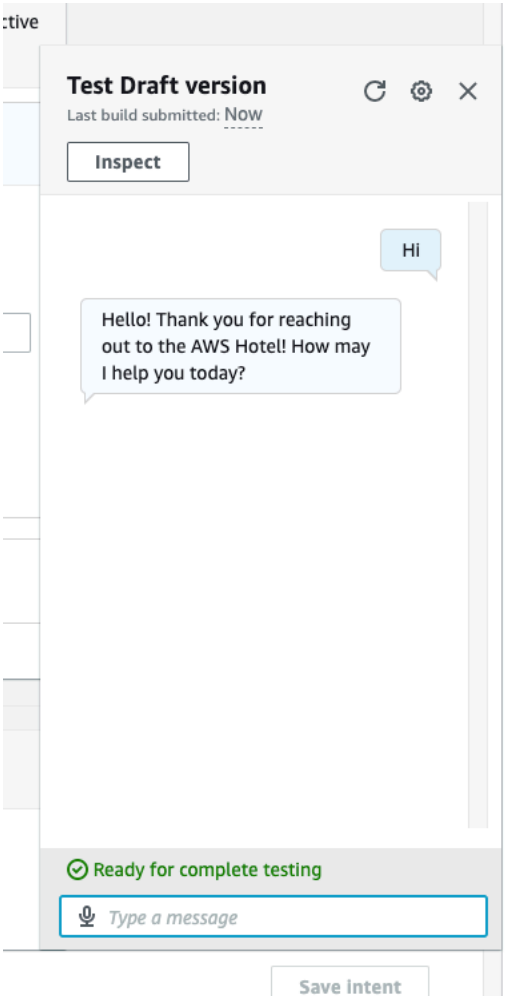
Now we need to create a closing message for the intent, which is what Amazon Lex will respond with if the intent processing is complete. Scroll down to ***Closing responses***, expand the ***Response sent to the user after the intent is fulfilled*** section, and in the ***Message*** field enter the following message:

Hello! Thank you for reaching out to the AWS Hotel! How may I help you today?



Hit the ***Save intent*** button, followed by the ***Build*** button - this will then build the bot and enable you to test it. The process will take about 30 seconds.

Once built - assuming that there are no errors - you can test out your new greeting bot by selecting the **Test** button. The following dialog will pop up, and you can interact with the bot by entering your opening message.

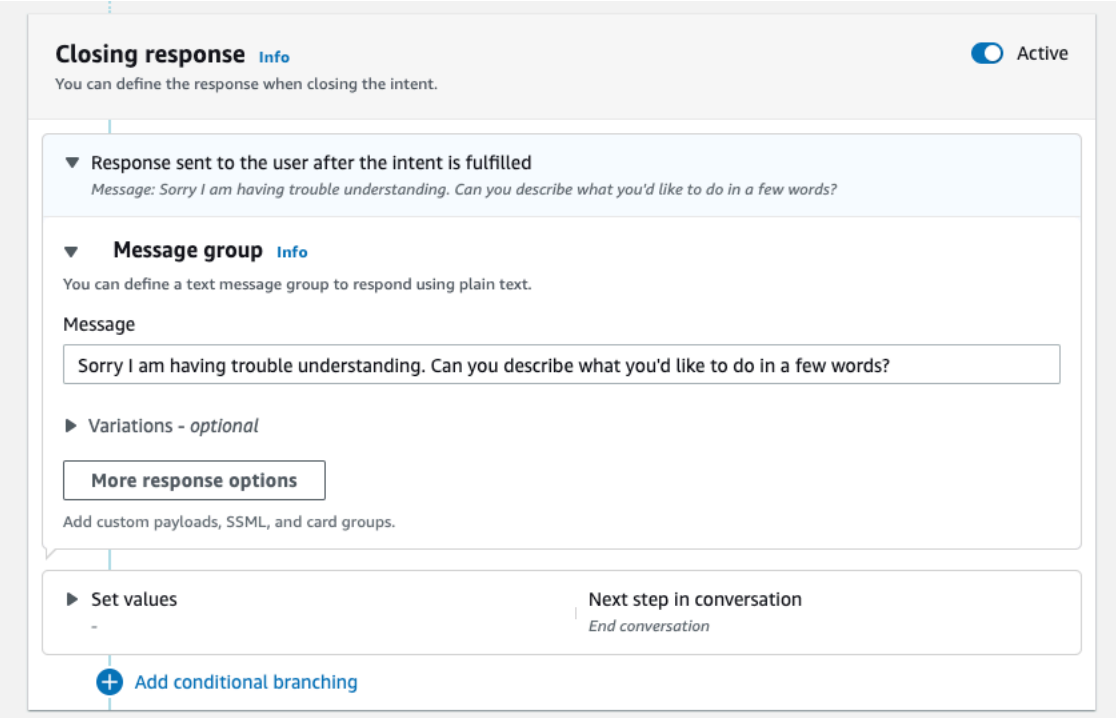


Configure the Fallbackintent

The default message from the **FallbackIntent** is fairly simple - we need to update this so that the user is given a better message, which should be to ask them to re-phrase their question in a better way. Click on the **FallbackIntent** item and scroll down to **Closing responses**.

Expand the speech section for ***Response sent to the user after the intent is fulfilled***, and in the ***Message*** field add the following text:

Sorry I am having trouble understanding. Can you describe what you'd like to do in a few words?



Hit the **Save intent** button, and click **Build**, once successfully built, then move to the next section.

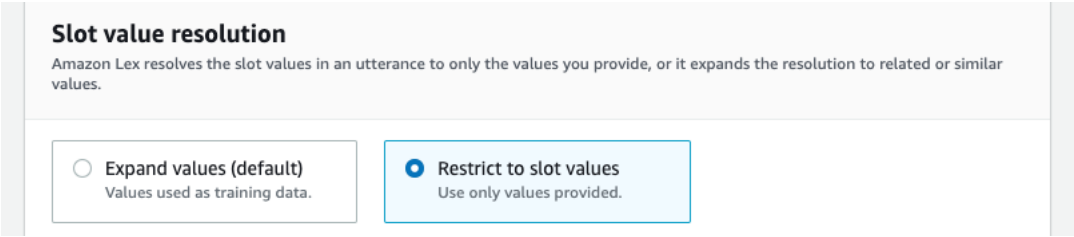
Create the HotelReservation Intent

Create a custom slot types

Now we are going to make this more powerful by adding the **HotelReservation** intent. This allows a user to make a hotel room reservation. We're going to support four different room types: **Standard, Queen, King and Double**. We can do this in one custom slot.

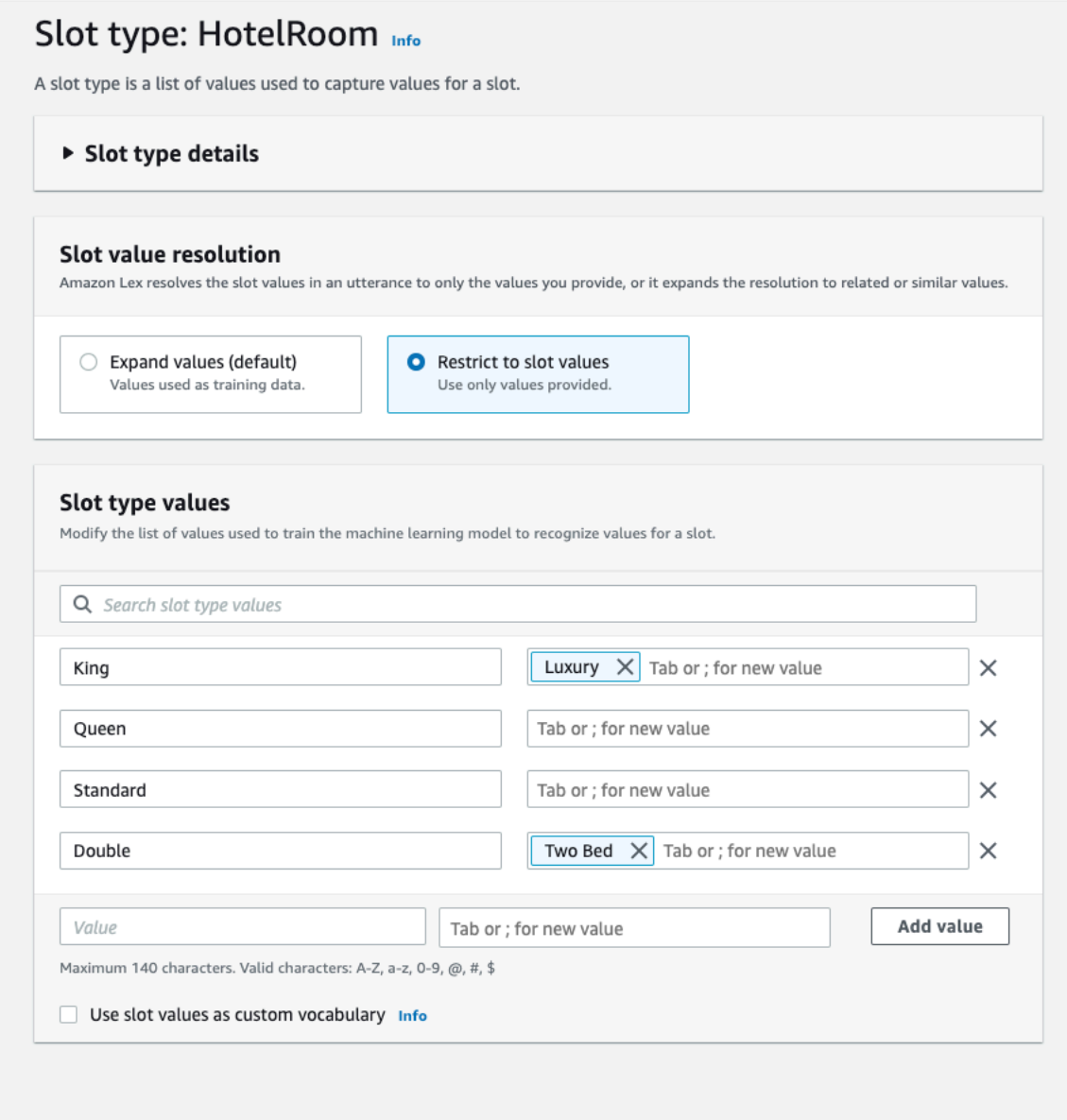
On the Amazon Lex Console, head to the **Slot types** panel. Click on the **Add slot type** button, and select the **Add blank slot type** - in the dialog that pops up please enter **HotelRoom** and click the **Add** button.

This will bring up a large **Slot types editor** panel, and the first thing to change is the **Slot value resolution**, which you want to set to **Restrict to slot values**.



This setting ensures that the only valid values possible are the ones that you supply in the next step. If you choose the **Expand values (default)** option then Amazon Lex will still honor the values that you supply, but if Amazon Lex's ML models see different values being given for these slots, then it will over time expand the resolution of slots to those values.

You can now add each of the hotel room types into the **Slot type values** panel - put each on one of the ***Value*** fields and click the **Add value** button.



There are also synonyms for King and Double value. The idea here is now when a user enters any of these synonyms then Amazon Lex will correctly map it back to the correct value, which means you don't have to write a back-end integration code to convert multiple possible variations of this slot.

Now click on the **Save slot type** button and your first custom slot is saved, and it is available for any intents within this bot language.

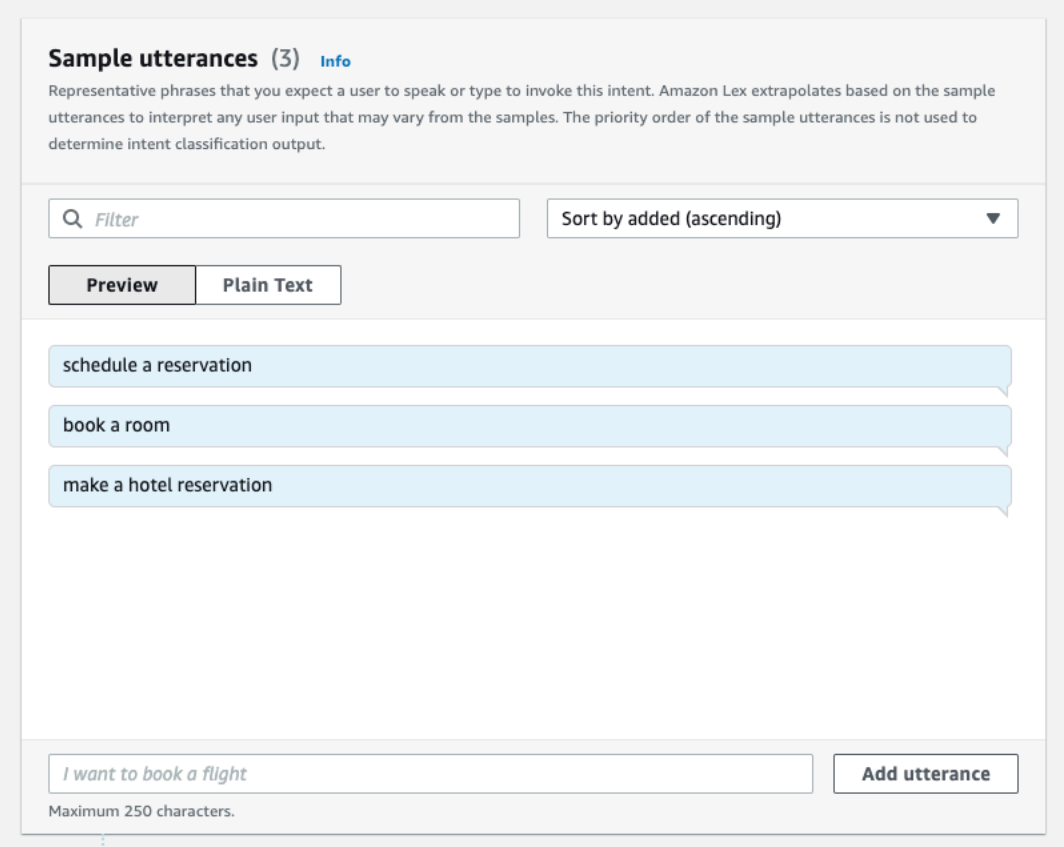
Create the HotelReservation intent

We are now ready to create the intent to make hotel reservation, the process here will be very similar to the **Greeting** intent.

Under **HotelBot** on the left-hand menu, click on **Intents**, then click the **Add intent** button, select ***Add empty intent***, and for the ***Intent Name*** please enter **HotelReservation**, then click the **Add** button. Once the **Intent editor** panel appears, enter the following description in the **Intent details** pane: Intent to make hotel reservation

Scroll down to **Sample utterances**. switch to ***Plain Text*** and paste in the following utterances. Once entered, click the *Preview* button, then click on the **Save intent** button.

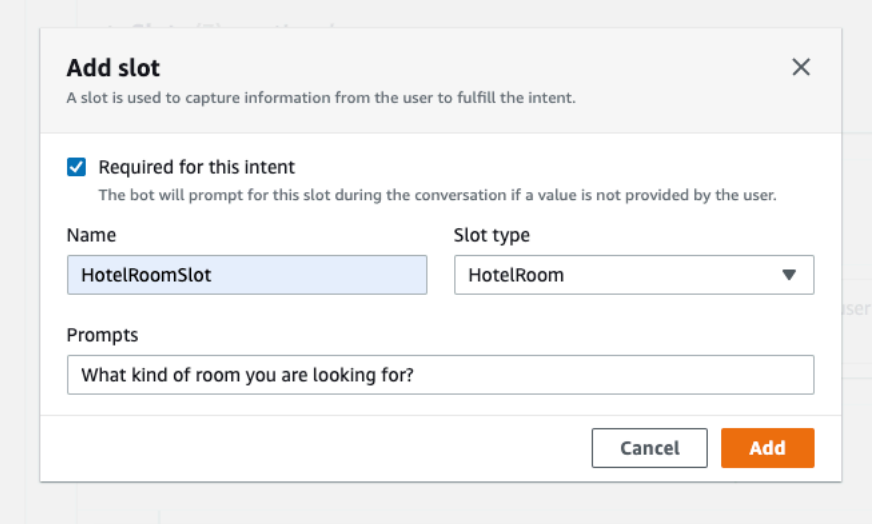
schedule a reservation
book a room
make a hotel reservation



Next you will scroll down to the **Slots** section. On this section, you will need to create couple slots for the bot to fulfill the required information.

Click **Add Slot**, and name the slot **HotelRoomSlot**. On the slot type section, choose the **HotelRoom** slot that we created earlier. and enter the following for the prompt:

What kind of room you are looking for?



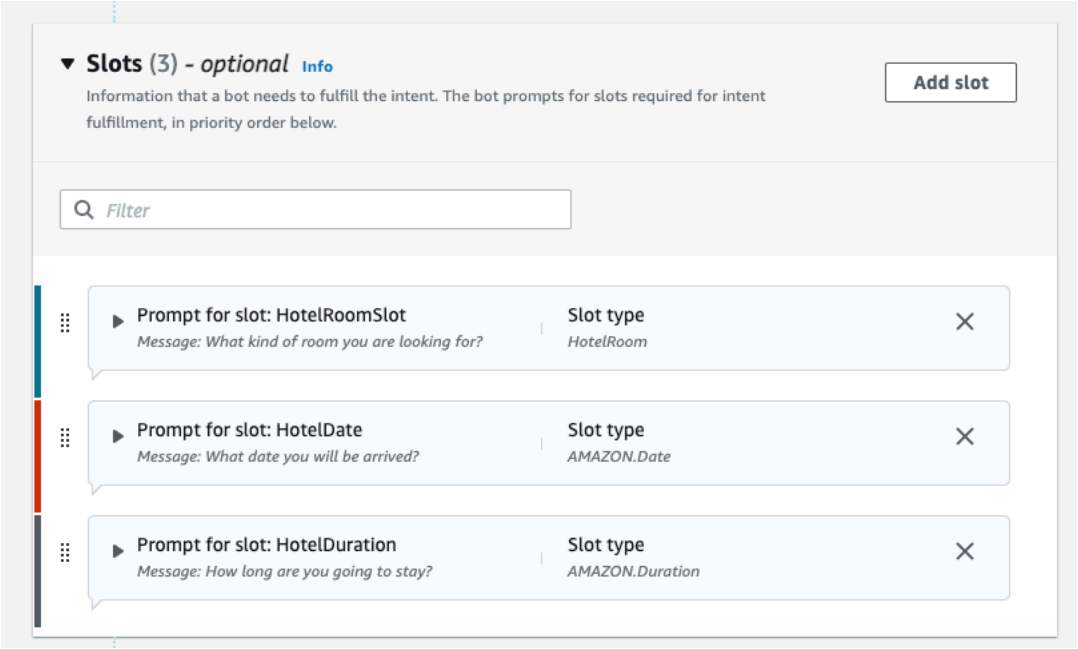
Now that we have our custom slot, we want to repeat this entire process, but this time we will use one of the built-in slot types. Start by clicking the **Add slot** button, and this time use these values and then click the **Add** button:

- **Name** - HotelDate
- **Slot type** - AMAZON.Date
- **Prompts** - What day will you arrive?

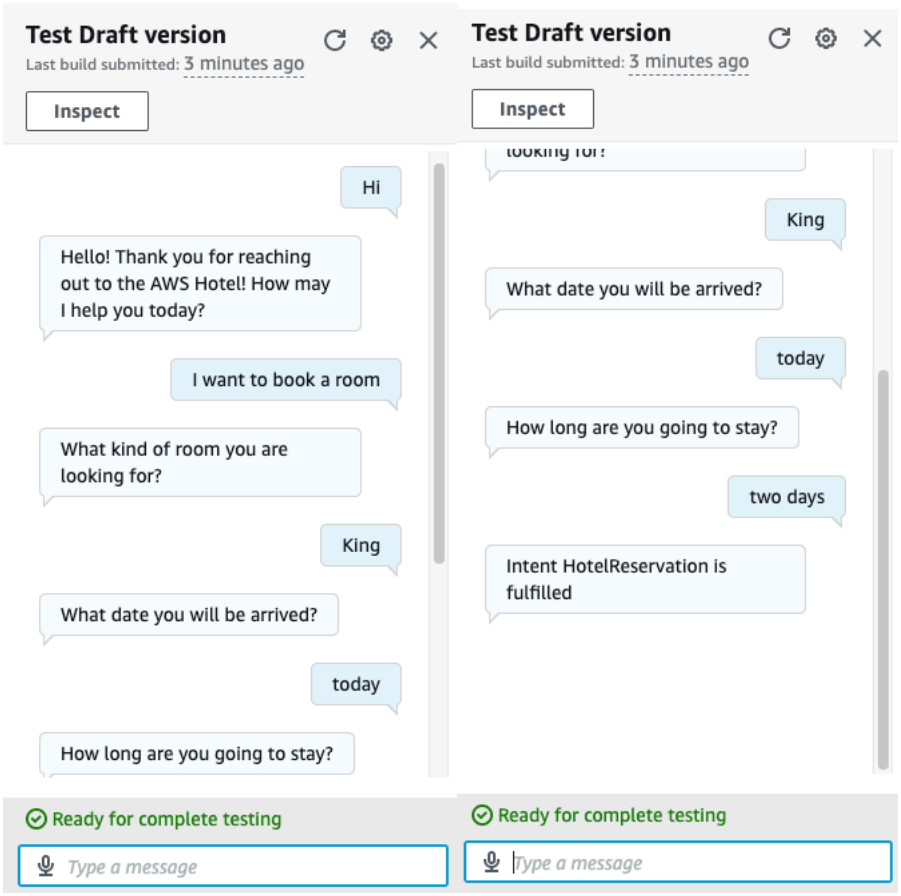
Then we will repeat the same process to create another one:

- **Name** - HotelDuration
- **Slot type** - AMAZON.Duration
- **Prompts** - How long are you going to stay?

You should now see the following slots in the system. You are able to open up each on to edit the value.



Now you can go through the save / build / test cycle and let's try this out, use one of the utterances such as “Make hotel reservation”. Amazon Lex will then prompt you in order. It does this using the prompts that you have defined in the intent slots, and the result looks like this.



In this example we have created a bot to collect information from the users, in the next section we will be creating a lambda function to integrate with Dynamodb, so once the chatbot captures the data, it will automatically push the data to the lambda function and store the data in the Amazon NoSQL database DynamoDB.

Create the bot Lambda function

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

In Amazon Lex V2 you now create a single Lambda function per language per chatbot, which must be able to support both types of Amazon Lex interaction:

1. **Initialization/Validation** - Lambda is called at every turn of the conversation. This allows you to initialize, validate or override slot data values.
2. **Fulfillment** - Lambda is called to alter all slots that have been elicited and any confirmation messages that have been okayed. This is the time for Lambda to deliver "the result" of the Amazon Lex intent.

The Lambda function must also be able handle the required logic for each intent. Each time Amazon Lex calls a Lambda function it will include everything that the bot currently knows about the session - this includes, but is not limited to, all elicited slot values, the current user utterance, the intent and slot currently being elicited, and information from some prior intent triggers during this user session.

Creating the Lambda function

For this Workshop we are using a simple python file - please download the code at the following link to your local computer, as we will need it later in this section.

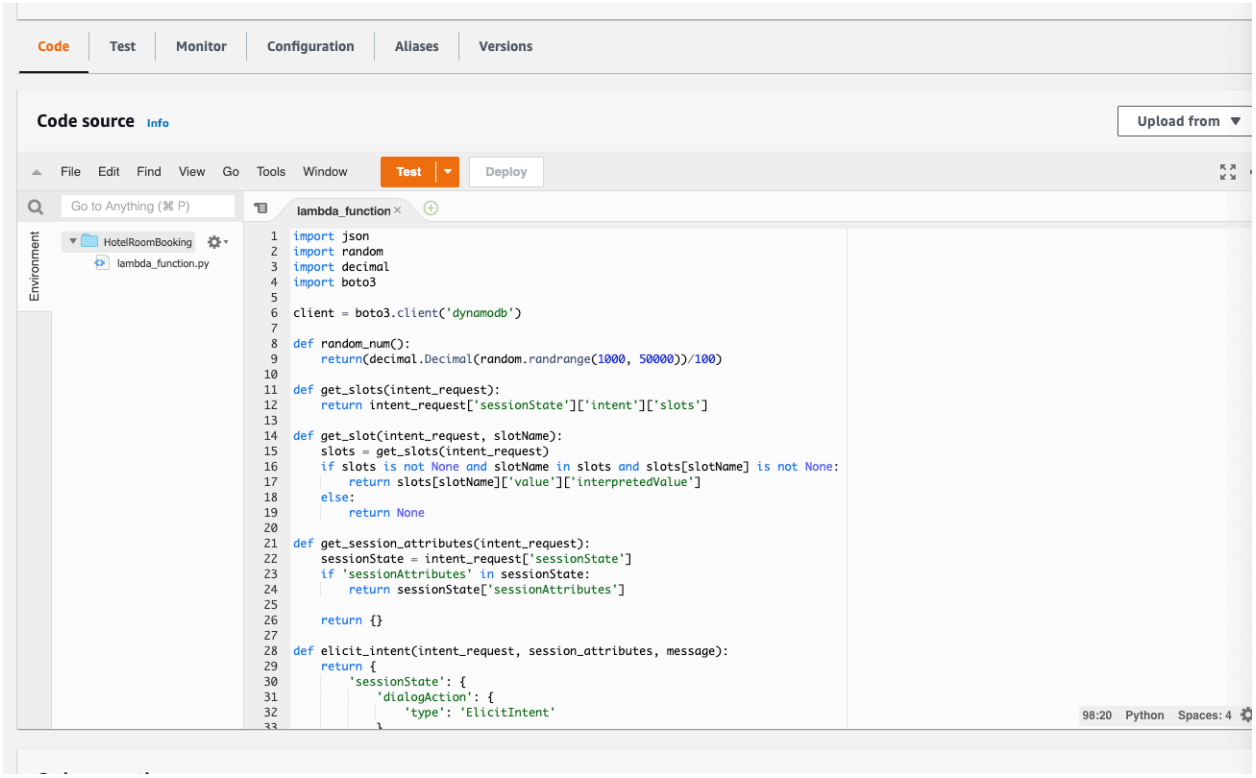
https://github.com/jkwnamazon/aws-nlp-2022-lex.git

Navigate within the AWS Console to the AWS Lambda service - type in **Lambda** in the console search bar and select **Lambda - Run Code without Thinking about Servers.** On the Lambda page click on the **Create function** button to begin the creation of your function.

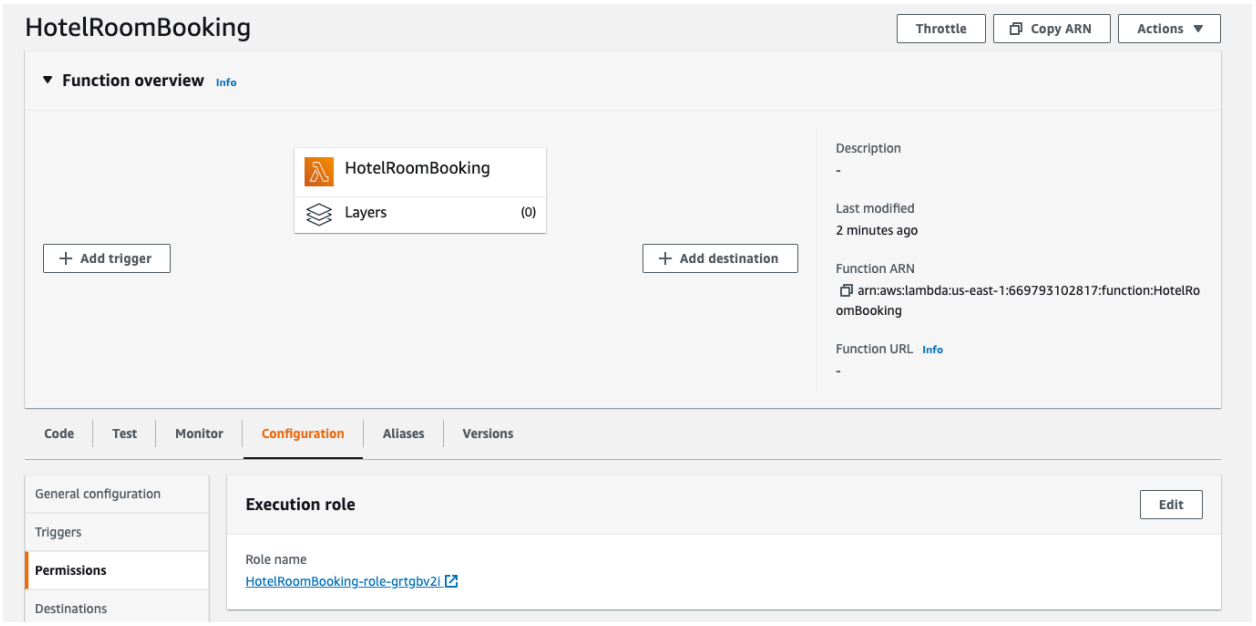
We only need to set a few parameters on this screen, so enter the following pieces of data and click on the **Create function** button on the bottom-right of the screen

- Select **Author from scratch** on the radio-button bar
- **Function name** - ***HotelRoomBooking***
- **Runtime** - Python 3.9

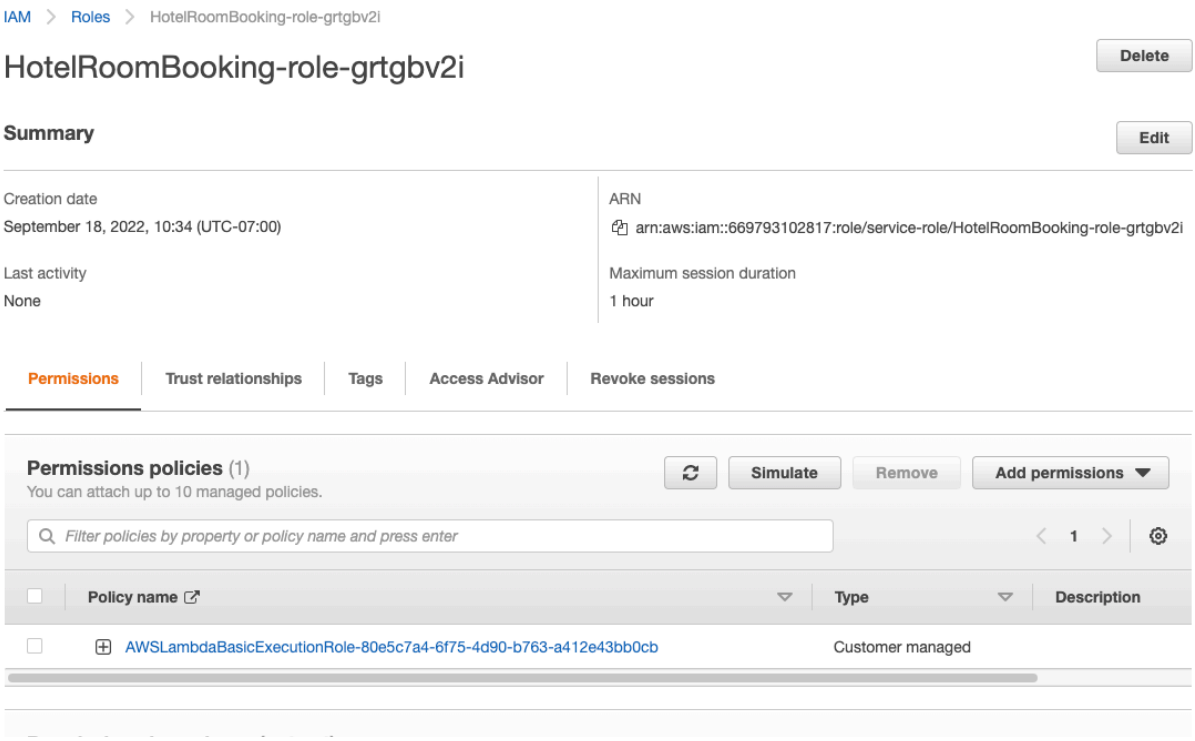
Leave everything else by default and click **Create Function**.



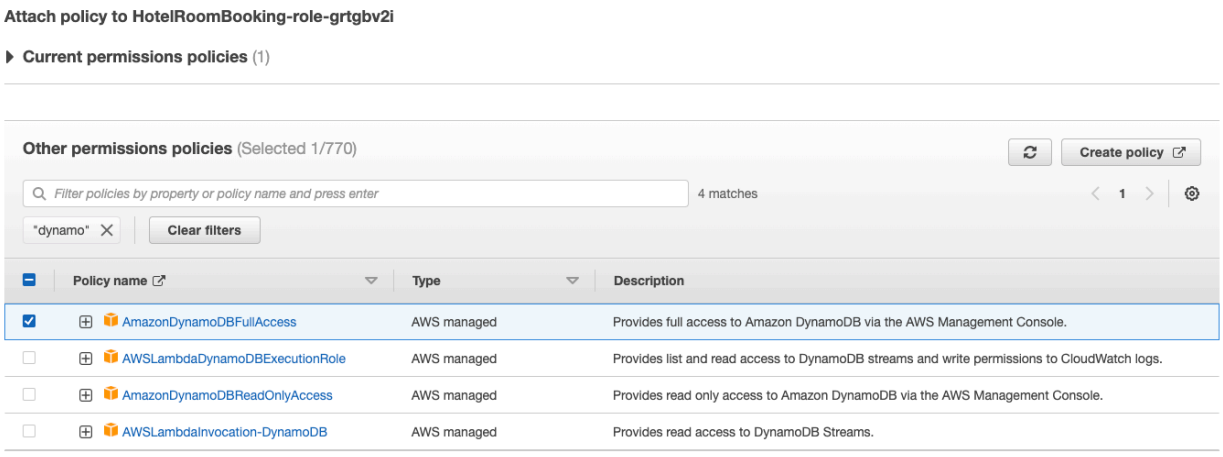
Now delete the default code, copy and paste the code from the python file from Github repo to the lambda function. Once you done, click **Deploy**.



Now click the **Configuration** section and go to the **Permission**, click on the HotelRoomBooking-role-xxxxxxx.



Now you are in the IAM Role console, you will need to attach a DynamoDB access to this IAM Role, so the lambda function could be executed and store the data to the DynamoDB database. Click **Add permissions** and **Attach policies**.

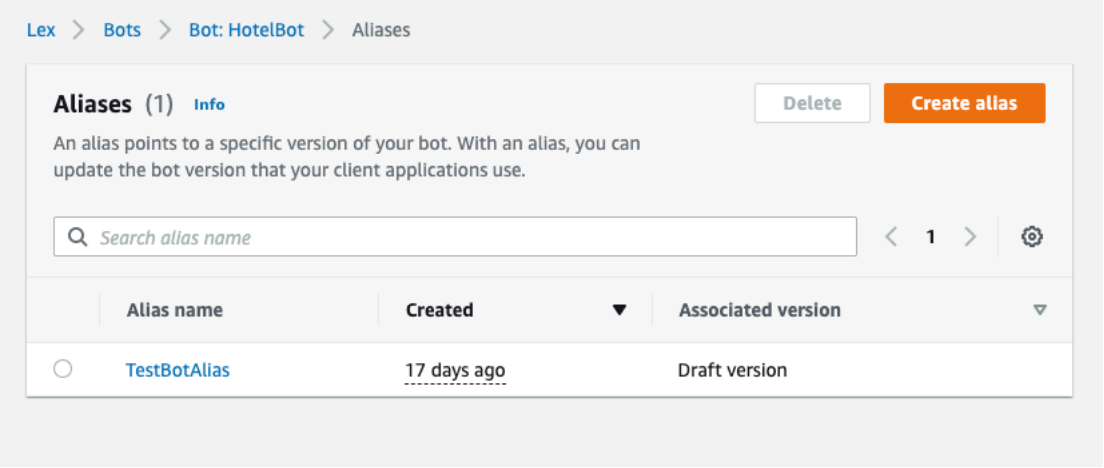


Search DynamoDB on the search bar, choose the **AmazonDynamoDBFullAccess** and click **Attach policies.** Now your IAM role has the DynamoDB write access.

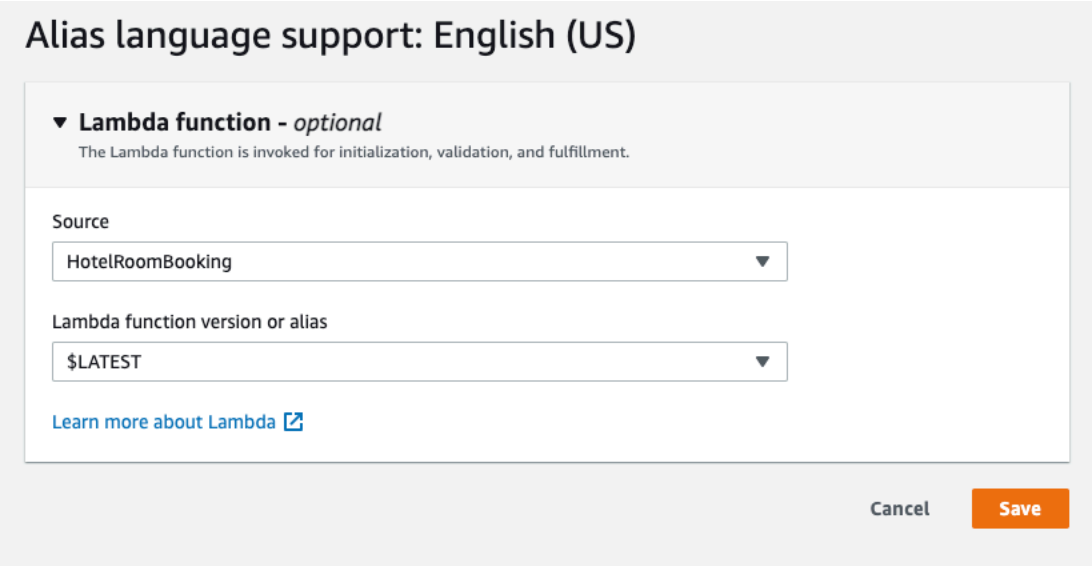
Configure bot Language configuration for Lamdba

Now head back to the Amazon Lex Console, Lambda functions are defined on a per language basis, but they can be changed to different functions as your Amazon Lex bot evolves over time. Hence, you actually associate a Lambda function with an alias of your bot - this essentially represents a version of your bot. You will have noticed previous that your bot is the Draft version, and that is the version that we want to associate this with.

On the Amazon Lex console click on the **HotelBot,** then on the left-hand menu select the **Aliases** link, which will bring up a full list of aliases defined for your bot.



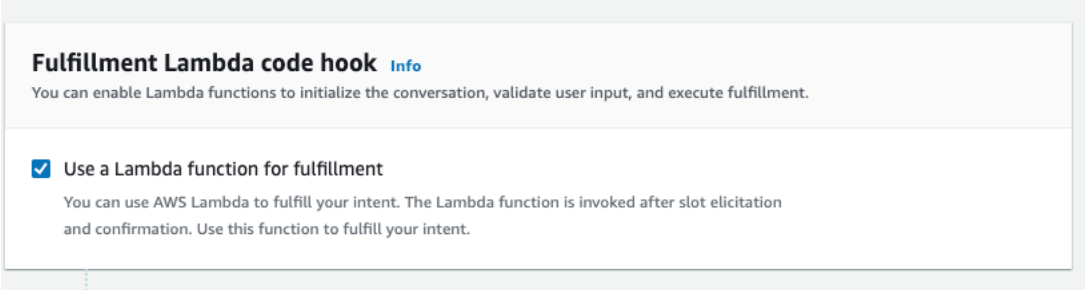
From here you choose the default **TestBotAlias**, then from the *Languages* panel select **English (US)**. This will bring up a dialog that allows you to associate the Lambda function to this Language/version combination of your bot. Select your Lambda function **HotelRoomBooking** in the Source drop-down field, leave the ***Lambda function version or alias*** at the default **$LATEST**. Click the **Save** button.



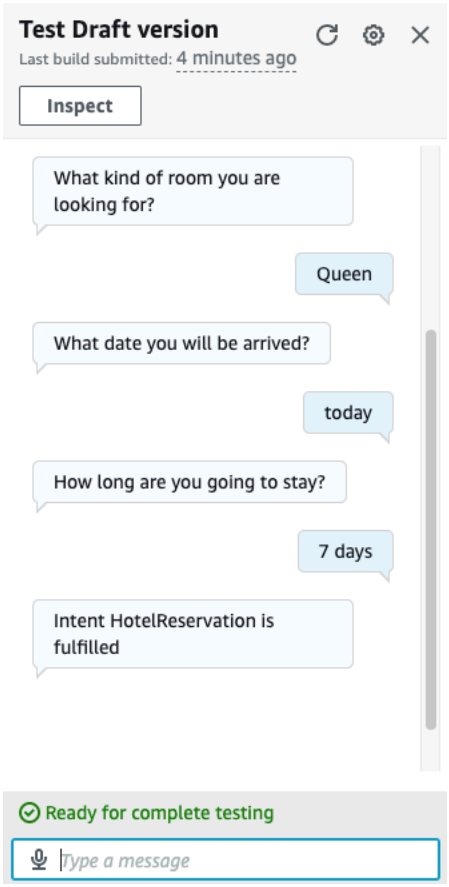
In Lambda you are able to create multiple versions of the same function, as well as aliased version names - your Amazon Lex bots can be associated with any of these specific versions, but if you use the *$LATEST* version then you will always use the most up-to-date version of the Lambda function - in development environments this is often the easiest way to work.

Enabling Lambda code hooks on the intents

The Lambda function is now ready to work on the ***HotelBot*** intents, but we still have to tell Amazon Lex which intents are to call the Lambda function and which tasks it should call upon to perform. Navigate to your intent by clicking on the **English (US)** link on the *Languages* pane, then on the far-left menu click on the *Intents* link to list your intents.
We will only associate the Lambda function with the ***HotelReservation*** intent - click on that intent and scroll down to the **Fulfillment** pane. Click on the arrow next to ***On successful fulfillment***, followed by the **Advanced options** button, which will bring up a long list of fulfillment options. Go to the one marked *Fulfillment Lambda code hook* and click the checkbox to enable a fulfillment Lambda for this intent, then click on the **Update options** button at the bottom of the panel.



Go through the save / build / test cycle again, and ask for make a hotel reservation - you should now see a result like the following test dialog.


Test Draft version
Last build submitted: 4 minutes ago
Inspect
What kind of room you are looking for?
Queen
What date you will be arrived?
today
How long are you going to stay?
7 days
Intent HotelReservation is fulfilled
Ready for complete testing
Type a message

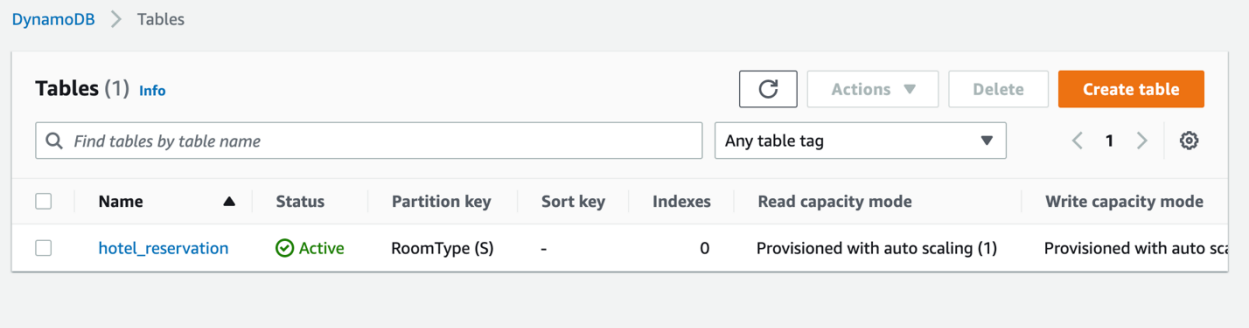
Create a DynamoDB database

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

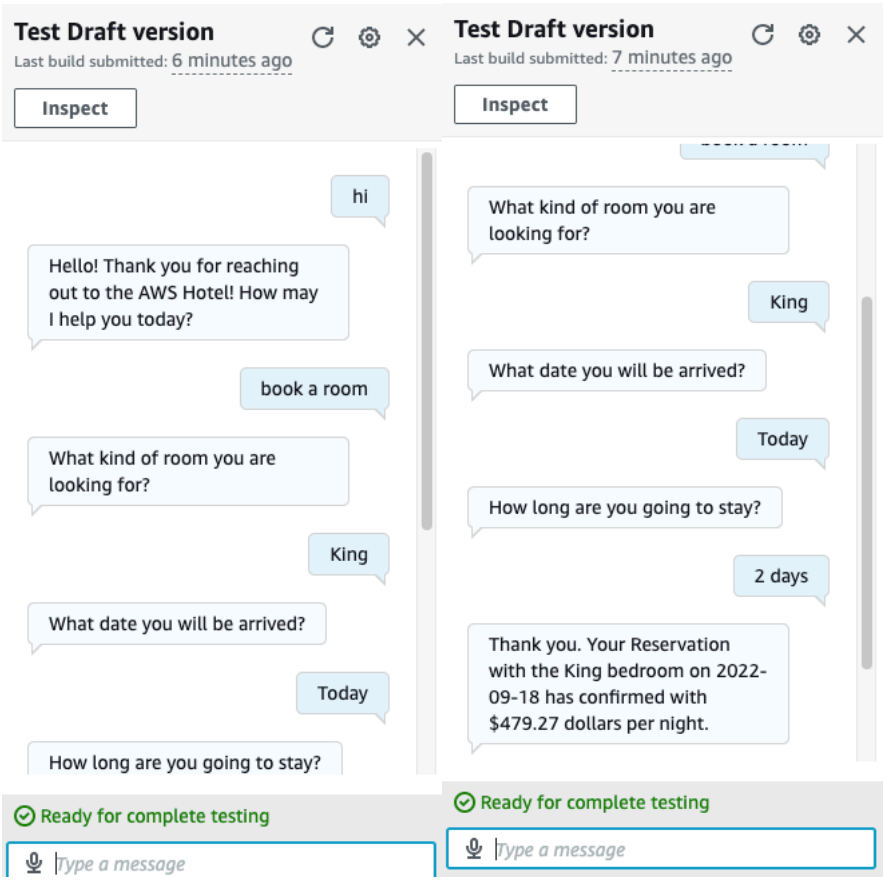
Head to the AWS console search bar, type DynamoDB and click it. Once you jump into the DynamoDB page, navigate to the **Tables** section.

Click **Create table**, enter the following information:
Table name: hotel_reservation
Partition key: RoomType

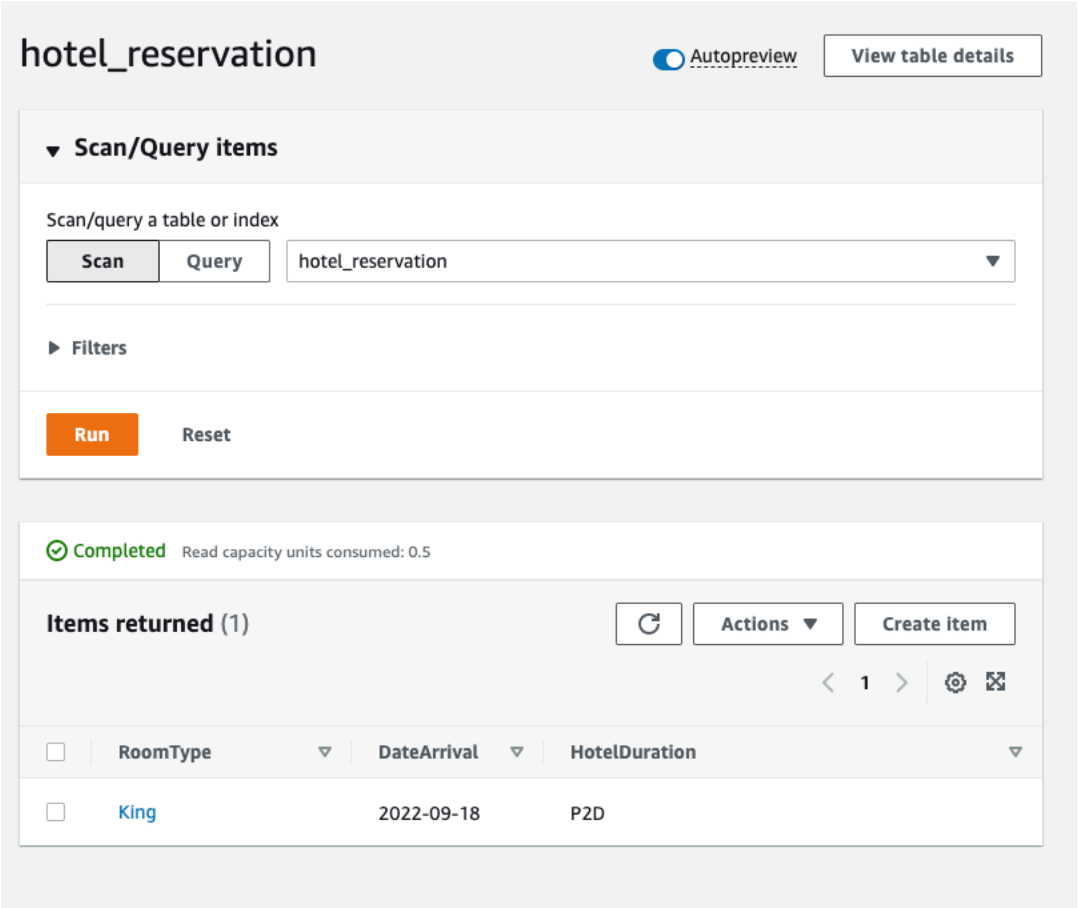
And select **String** for the partition key value. Then leave the Table setting as **default settings**. Scroll down and click **create table**.



The database table status should be ***Active*** within a minute. Please verify your table name and the partition key are exactly the same as the one provided on the instruction. This will be used by the lambda function later.



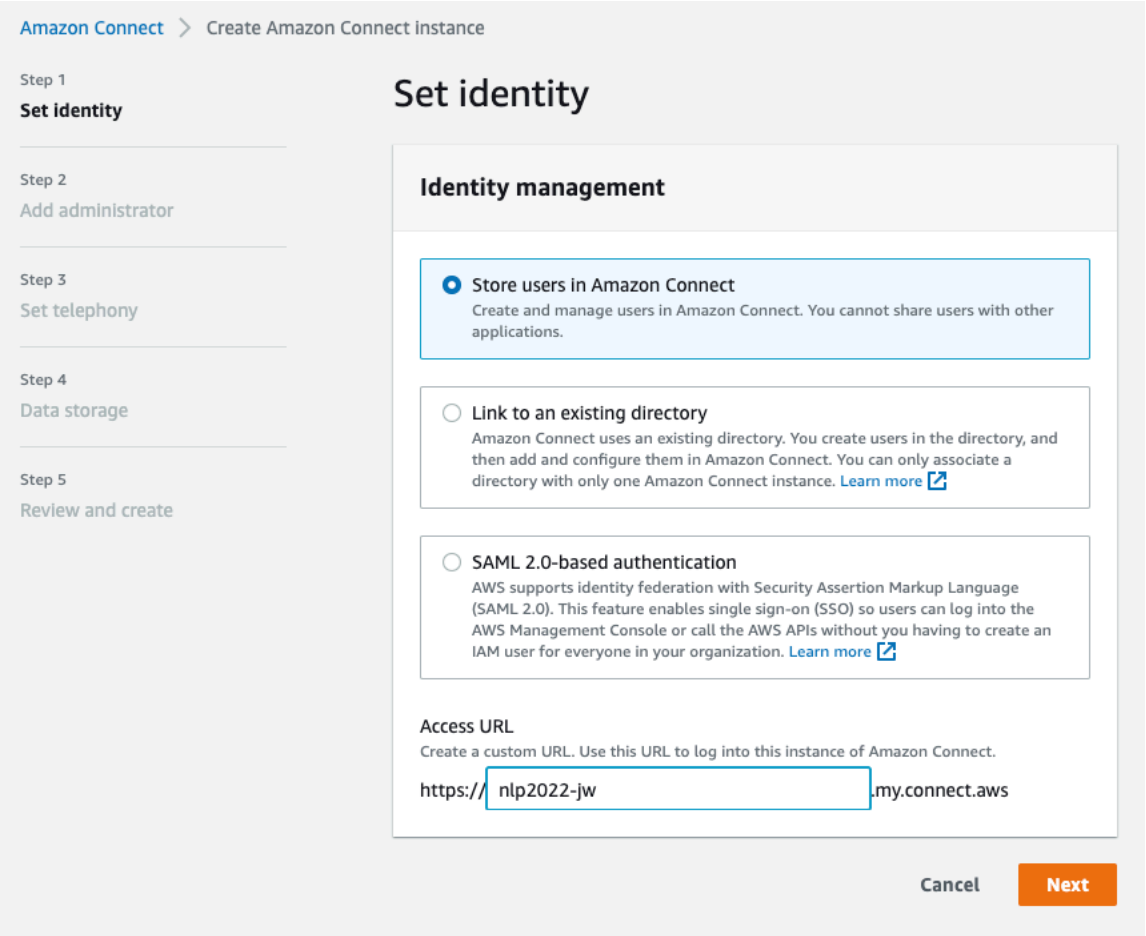
Now head back to the Amazon Lex V2 and test the **HotelBot** again. This time because the Lex triggered the lambda function, so you will see the confirmation message from the lambda function indicate the thank you note, confirmation details and hotel pricing.



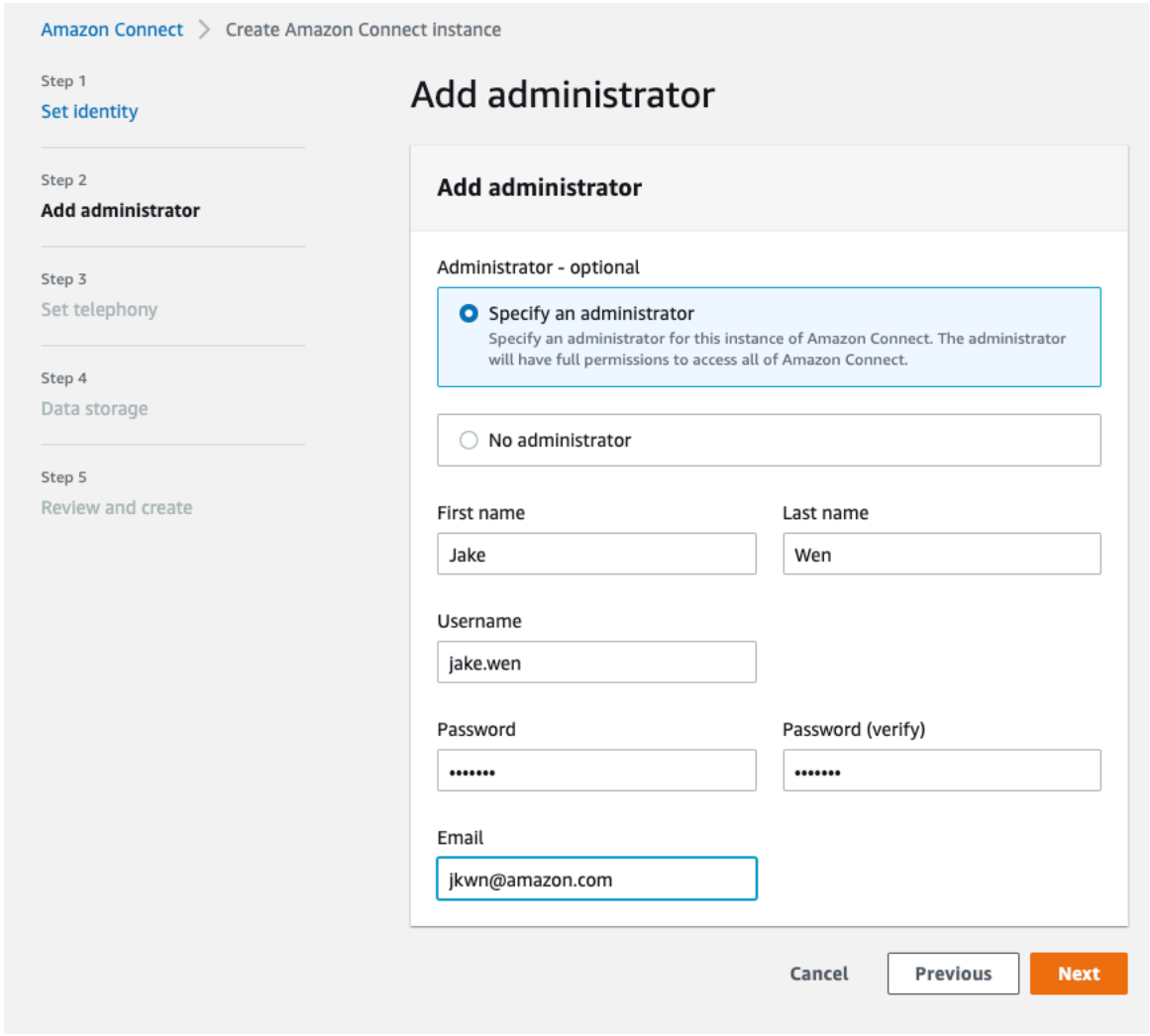
Great! You are almost there! Now you will head back to the DynamoDB database, click the database's name to get into the database. Then click the ***Explore table items*** option. Now you will see your reservation data stored in the DynamoDB table.

Create an Amazon Connect Instance

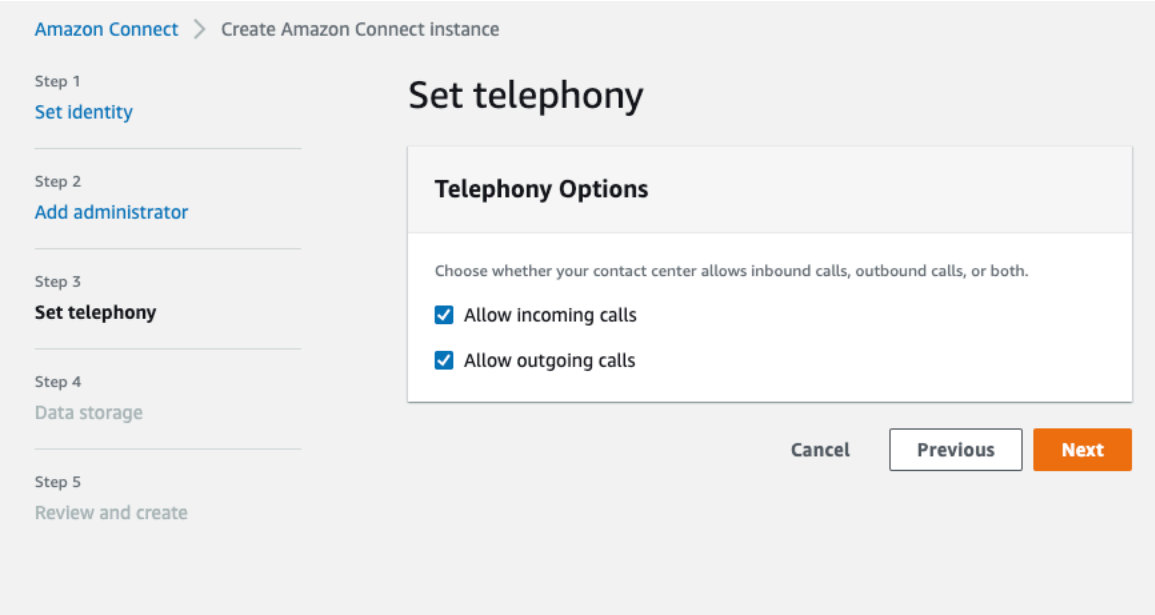
On the top of the AWS console, search **Amazon Connect** on the search bar and click it. Now you will be presented with an Amazon Connect introduction screen. Select **Create instance**.



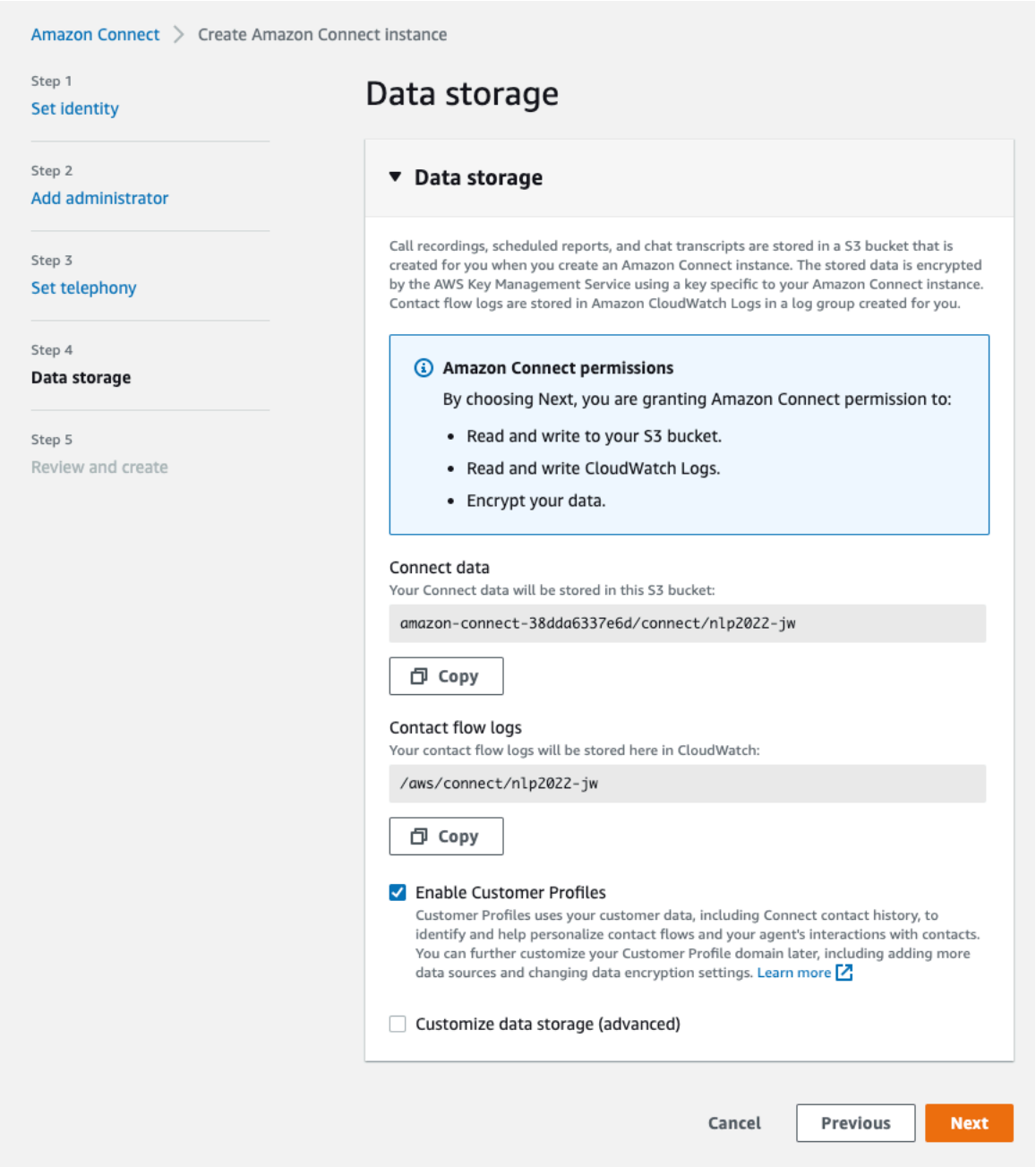
For **Set identity** – Feel free to enter a unique instance name and choose **Next**.



Add administrator - Enter details for creating an administrator user for your new instance. Choose **Next** once done.

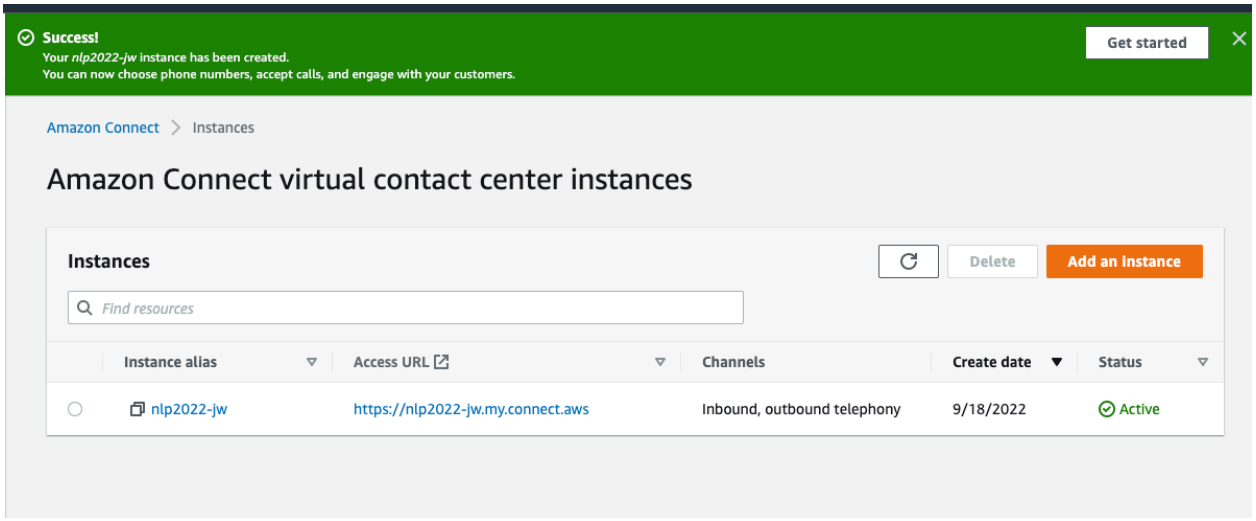


Set telephony - Keep the default options selected to allow both incoming and outgoing calls for your instance. Choose **Next**.

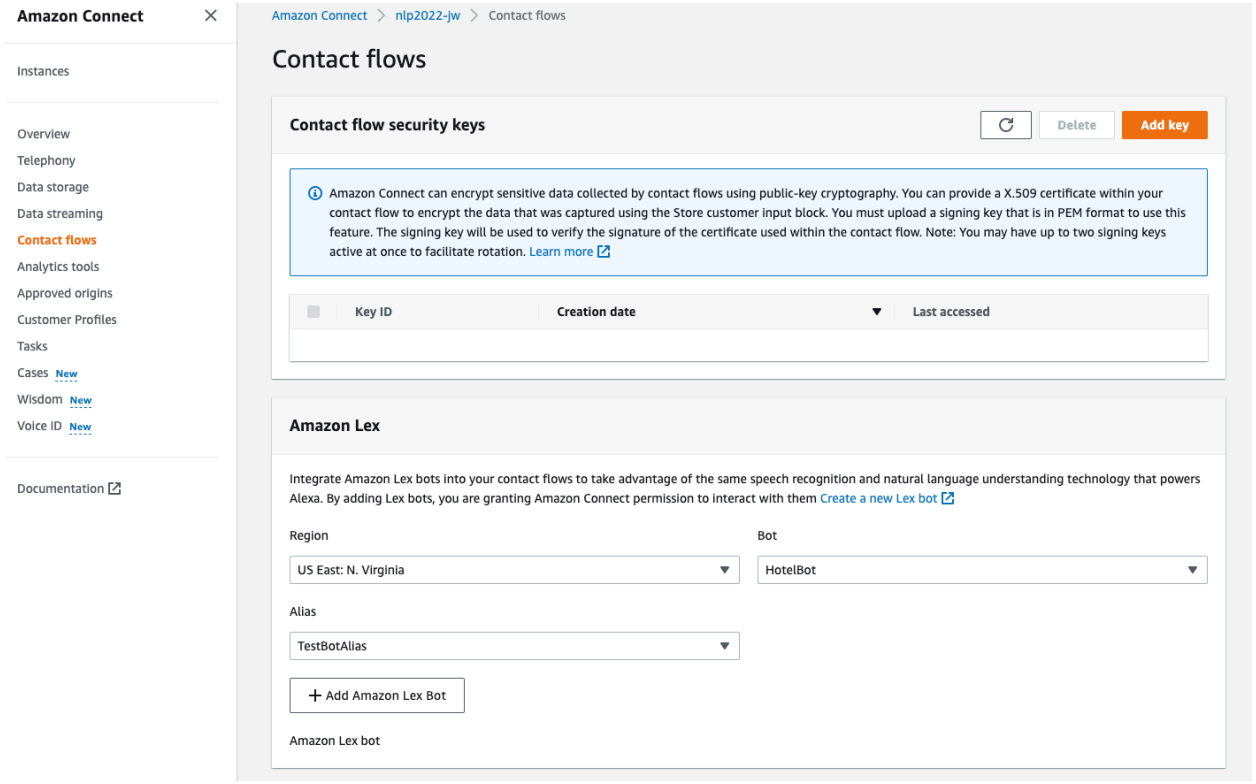


Data storage - Keep the default options selected for data storage. Choose **Next**.

Review and create - Review all settings and information before choosing **Create instance** to create your instance. The access URL for your instance is also shown on this screen.



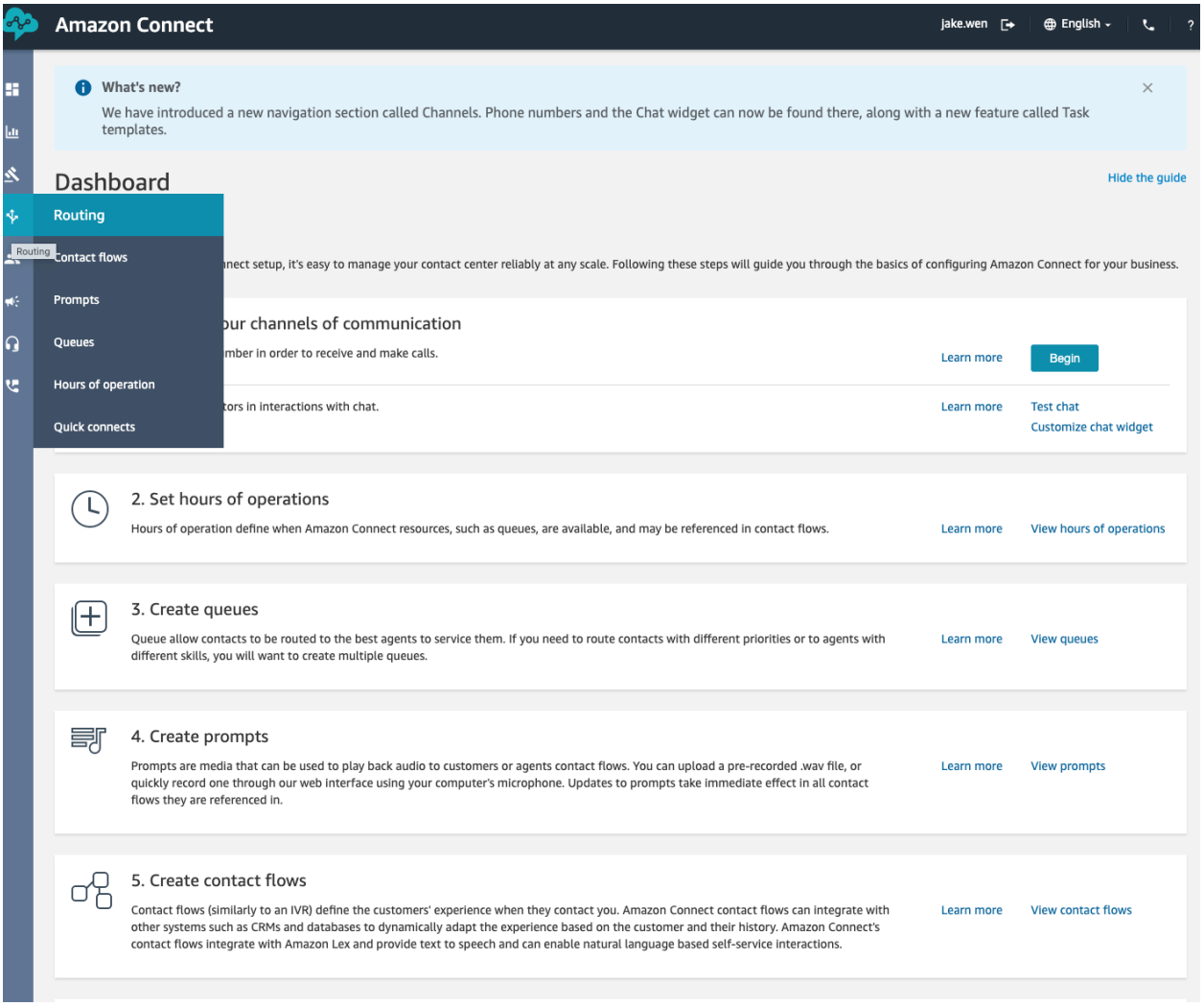
Before you login in to the Amazon Connect page, click the **instance alias** and go to the **Contact flows** option.



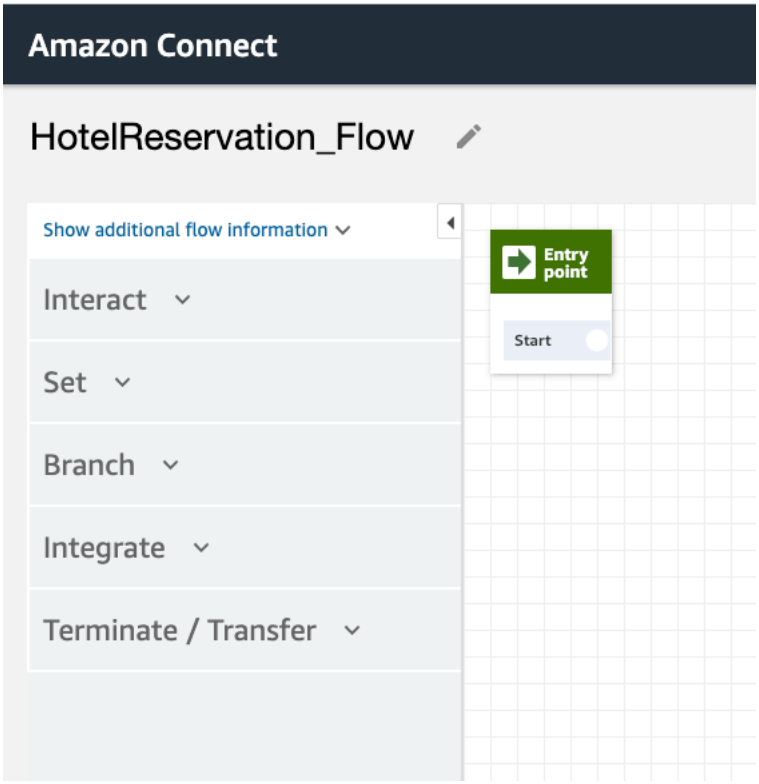
On this page, choose the HotelBot we just created, picked the TestBotAlias and click **Add Amazon Lex Bot.**

Build a contact flow to use Lex Bot

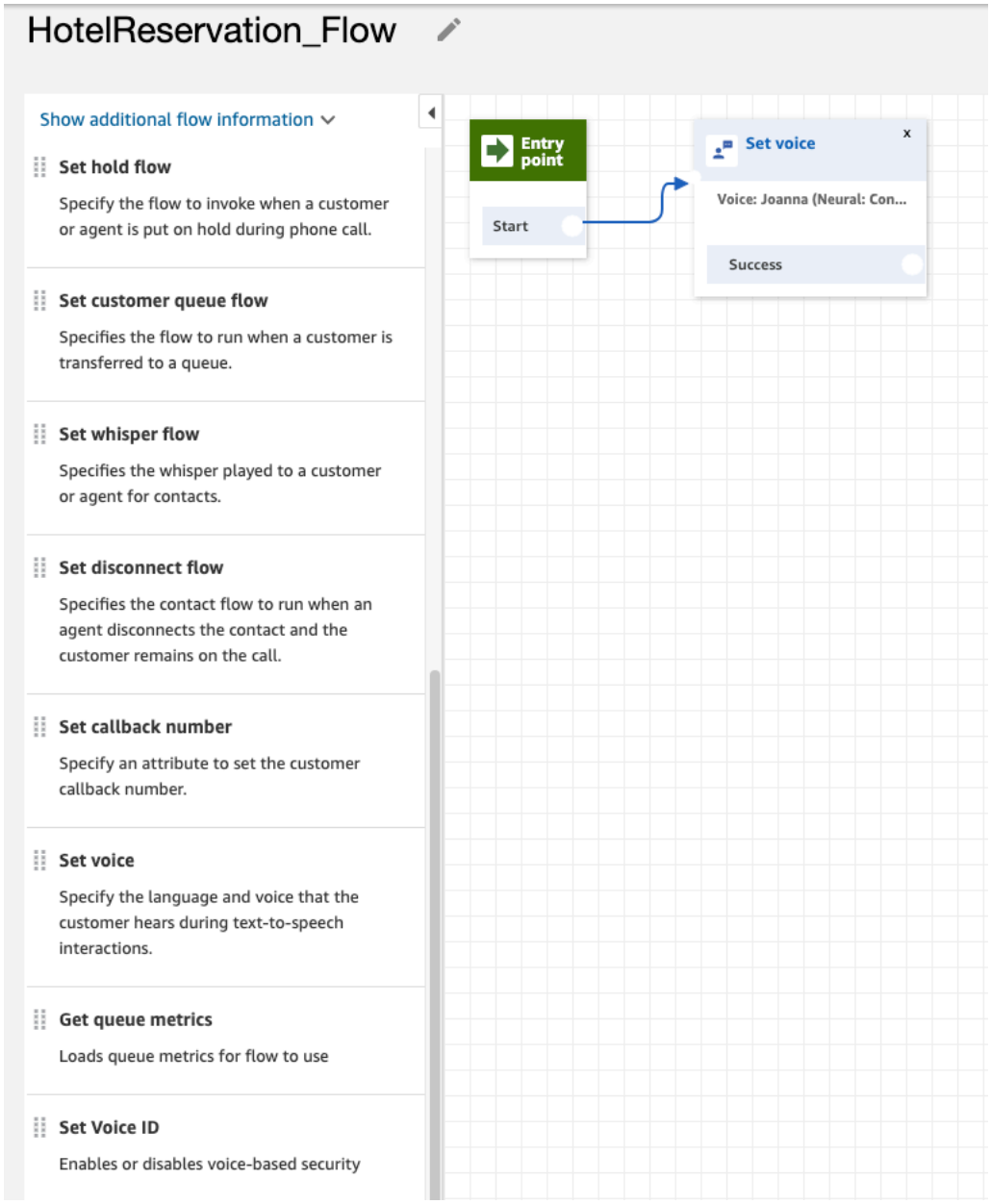
Open the Amazon Connect console and click the **Access URL,** enter your username and password we just created a moment ago to login to the Amazon Connect Console.



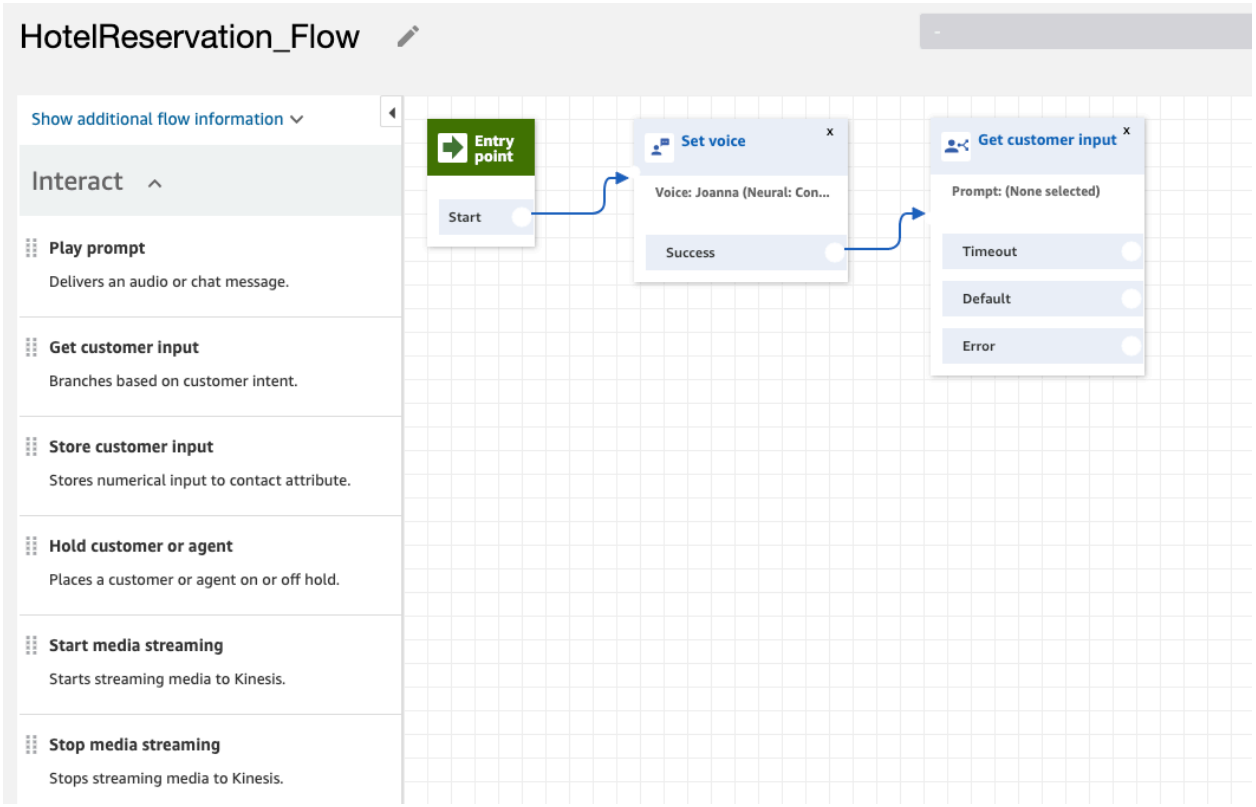
Once you login, click the **Contact Flows** option on the left pane under **Routing**. Then Choose on **Create contact flow** on the top right.



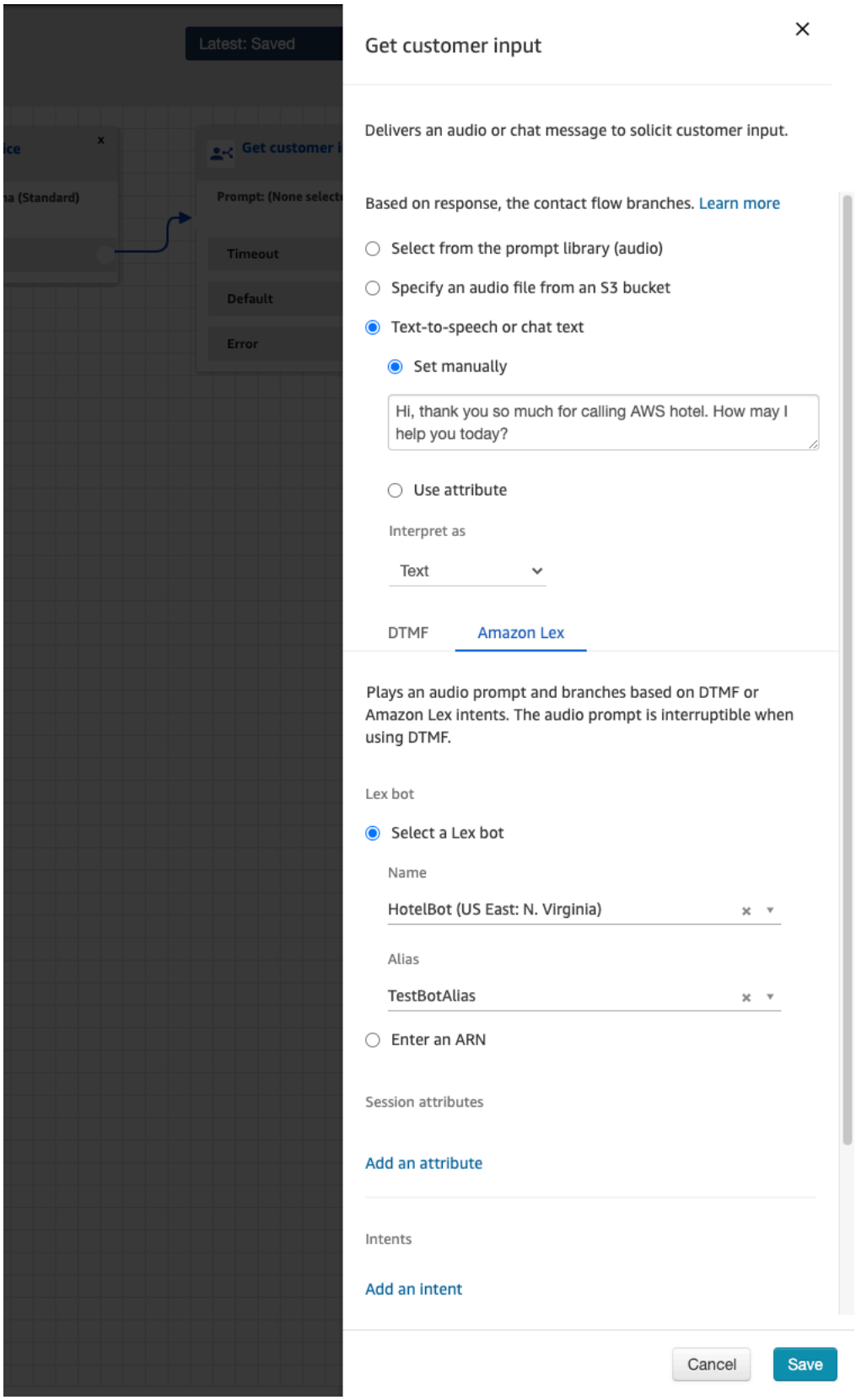
We have to give a name to this contact flow, now enter the name HotelReservation_Flow on the top left side.



In the ***Set*** section, bring a **Set voice** block onto the design pane and connect it to the Start output from the Entry point block.



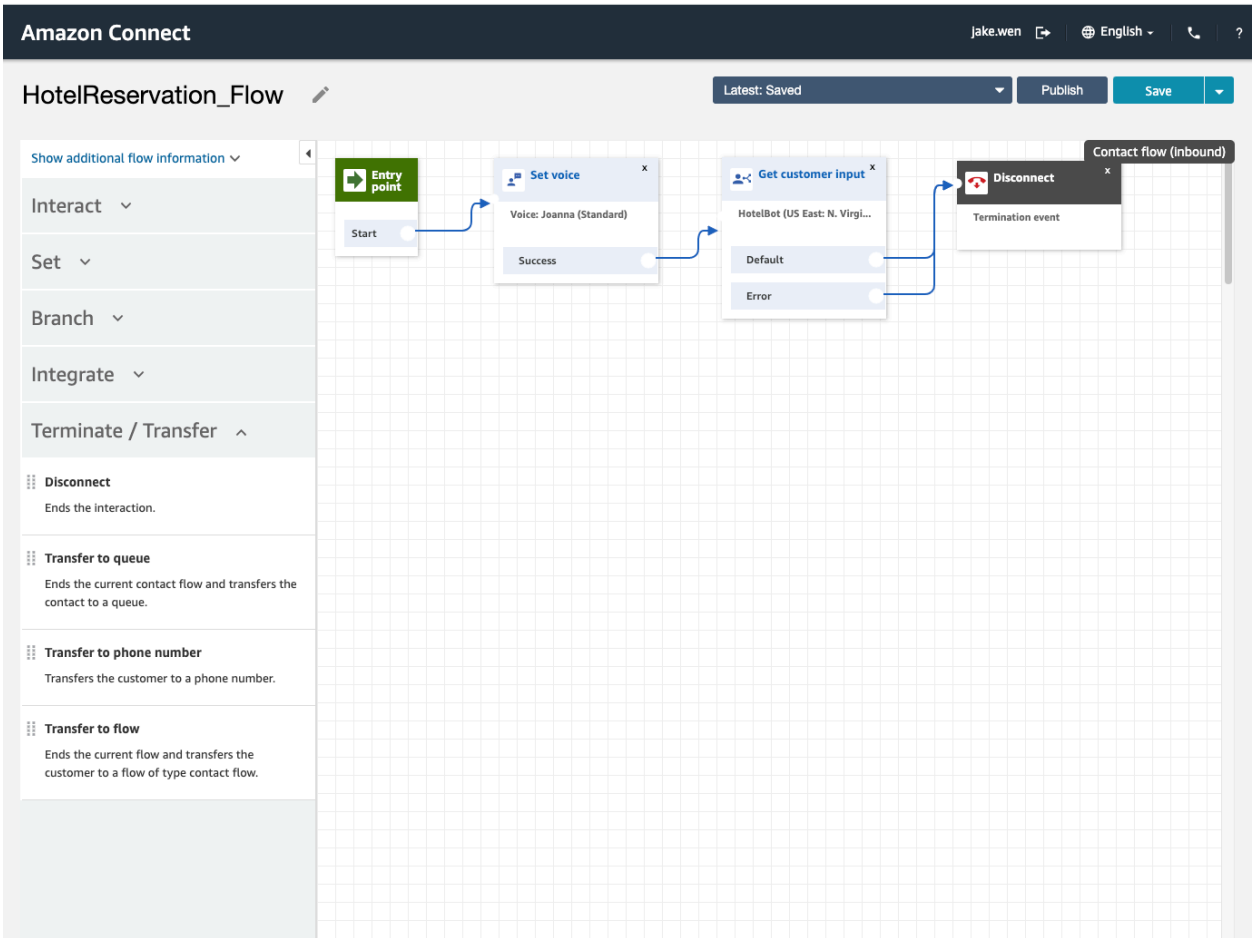
Next in the **Interact** section, bring a **Get customer input** block onto the design pane and click the block.



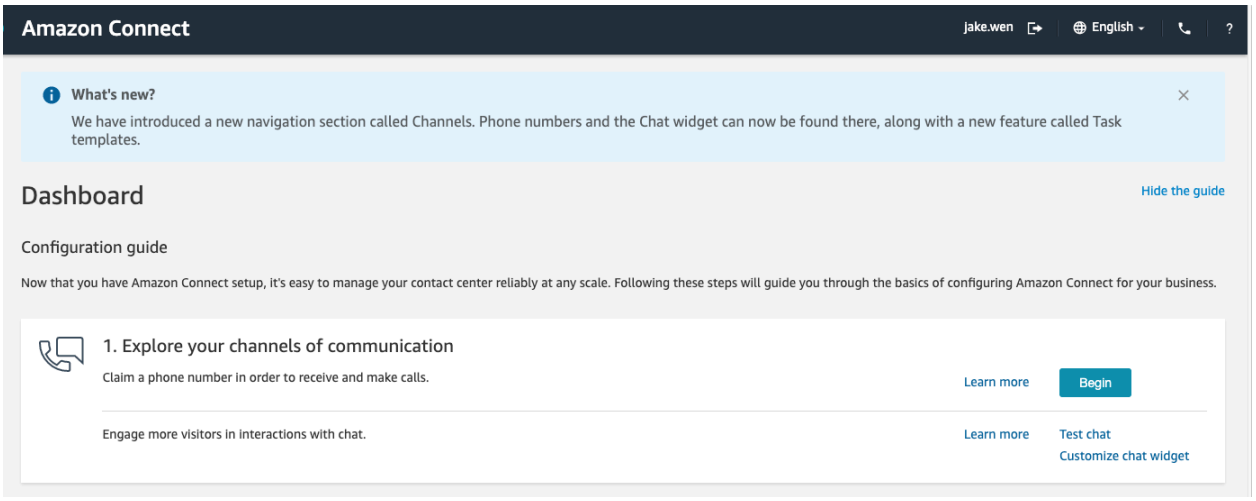
Choose the Text-to-speech or chat text option, and enter the following greeting message:

Hi, thank you so much for calling AWS hotel. How may I help you today?

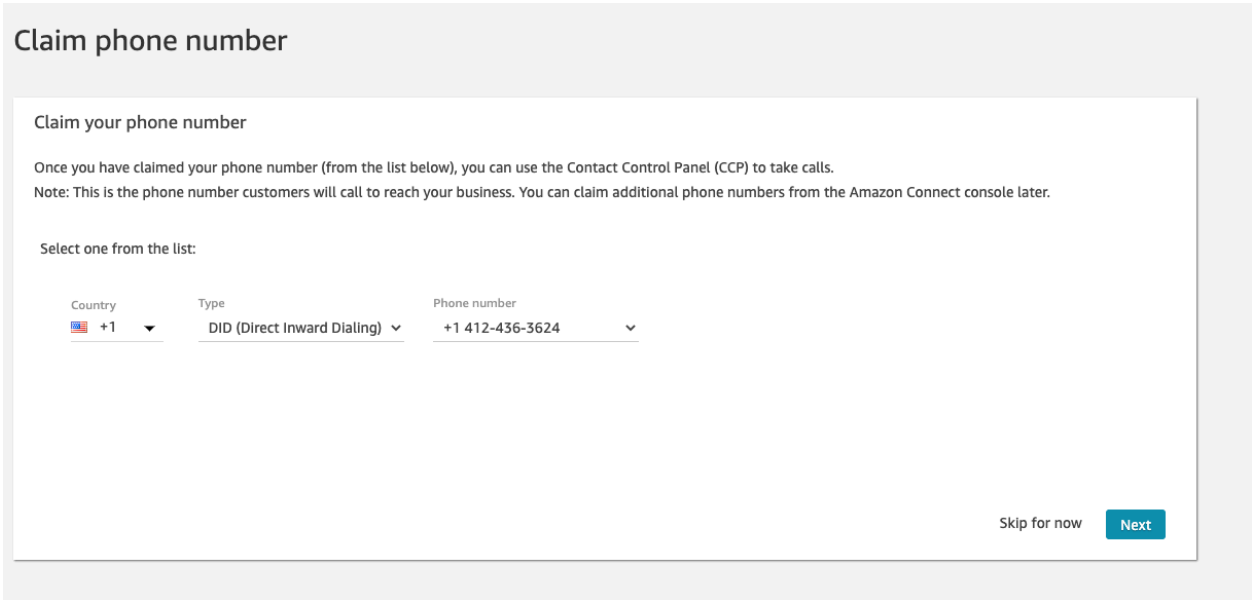
On the next section, choose Amazon Lex and select the HotelBot and the corresponding Alias. Then click **Save**.



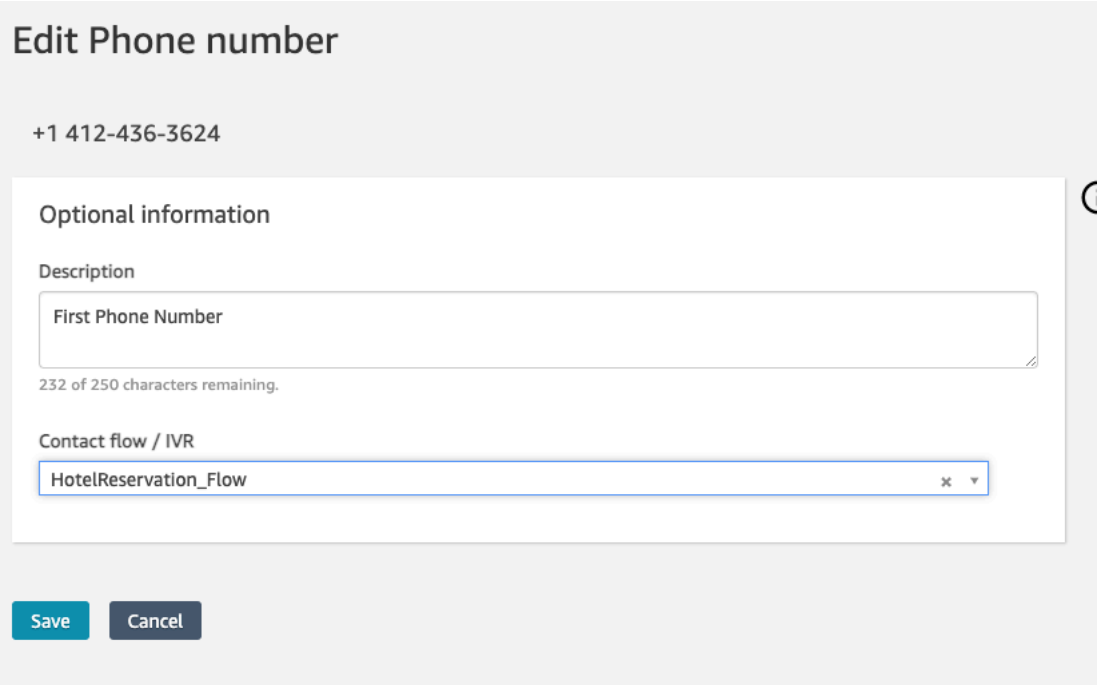
On the Terminate/ Transfer option, pick **Disconnect** and save. Now you have built your very first contact flow from scratch. You can click Publish now to make the contact flow live.



Now head back to the Amazon Connect home page and click **Begin** to claim a phone number.



For claim phone number, you can choose the country by preference. Then click **Next**.



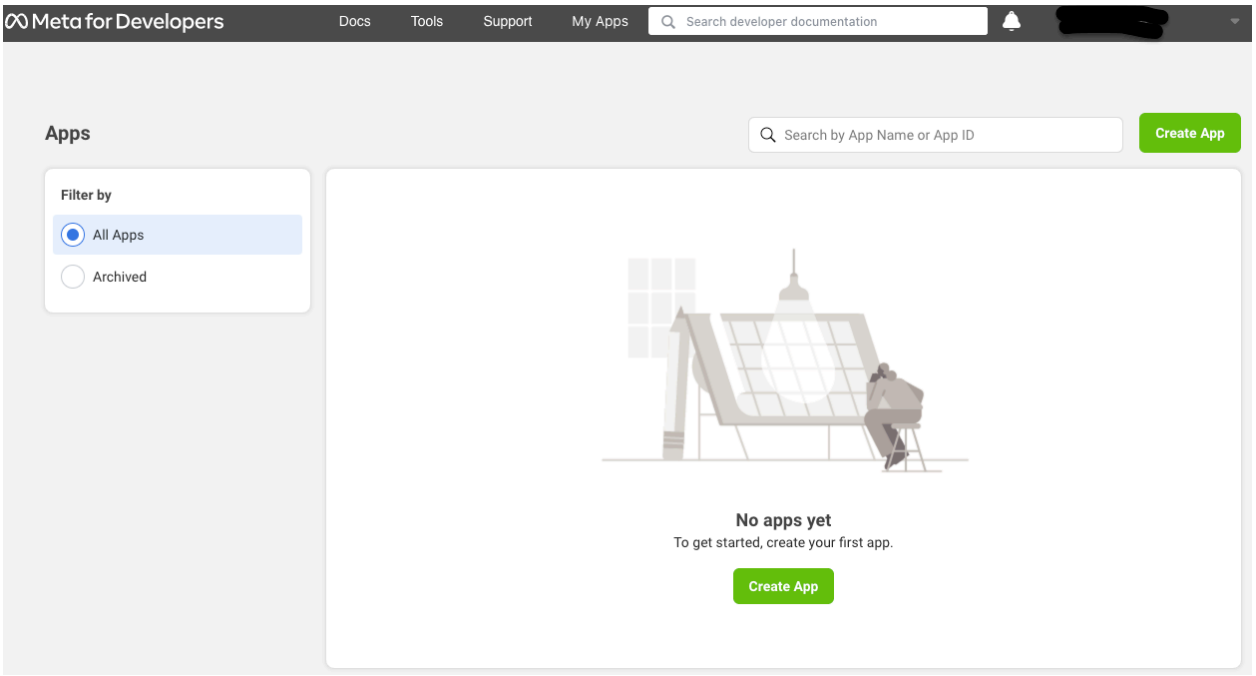
Once you successfully claimed a phone number, click the phone number into the **Edit Phone Number** page, attach the **HotelReservation_Flow** to this phone number and click **Save**.

Awesome! You made it! Now you can call the phone number you just claimed and try it out!

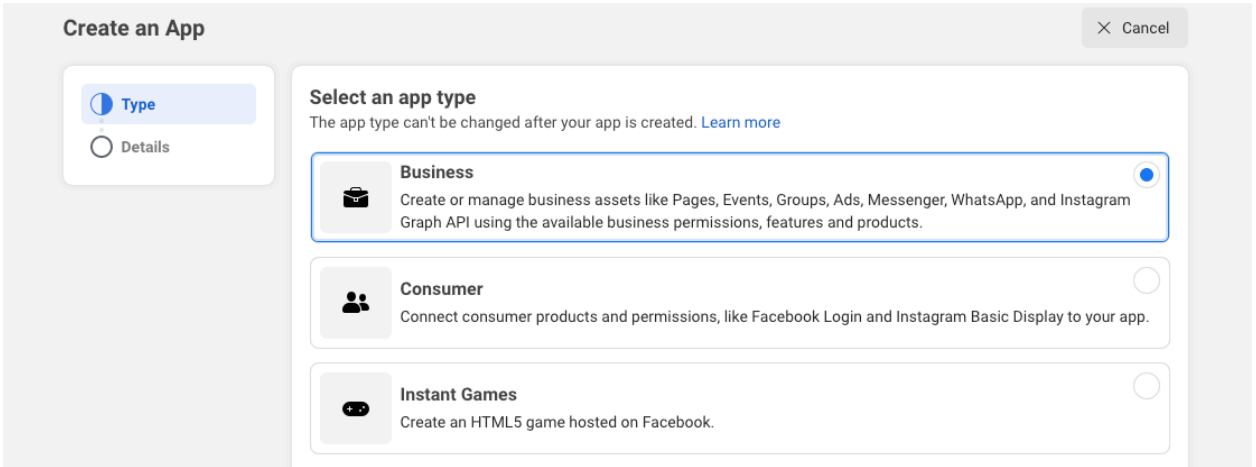
Optional: Integrate with Facebook Messager

If you have a Meta developer (Facebook developer) account and want to test it out the Lex integration with Facebook Messenger this is how.

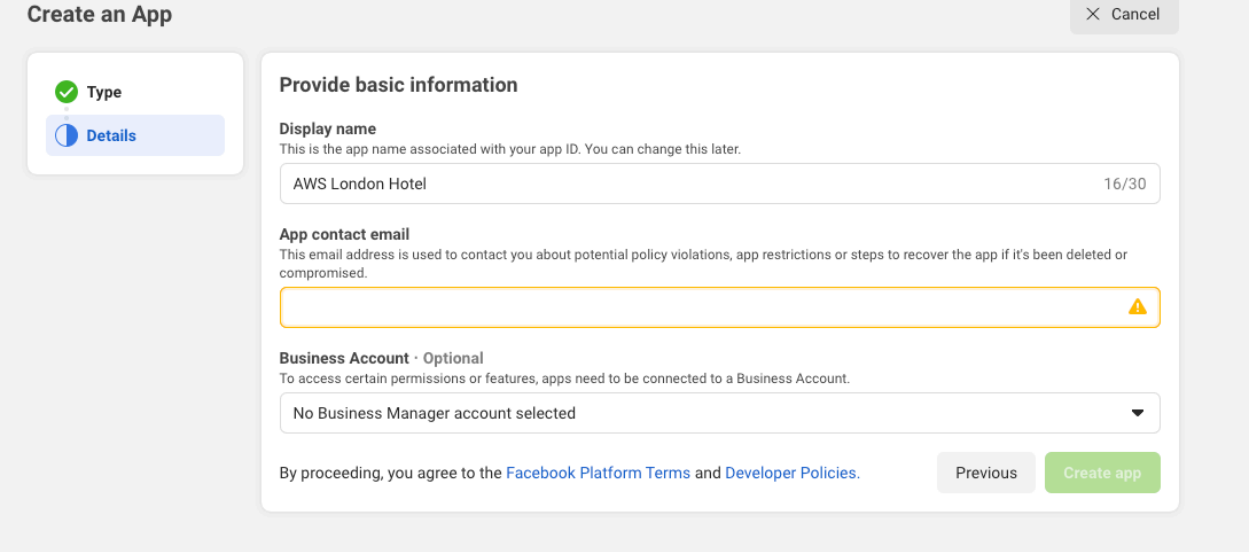
Visit this link: https://developers.facebook.com/ and click **Get Started** on the right upper corner. Fill in the required information and click complete registration.



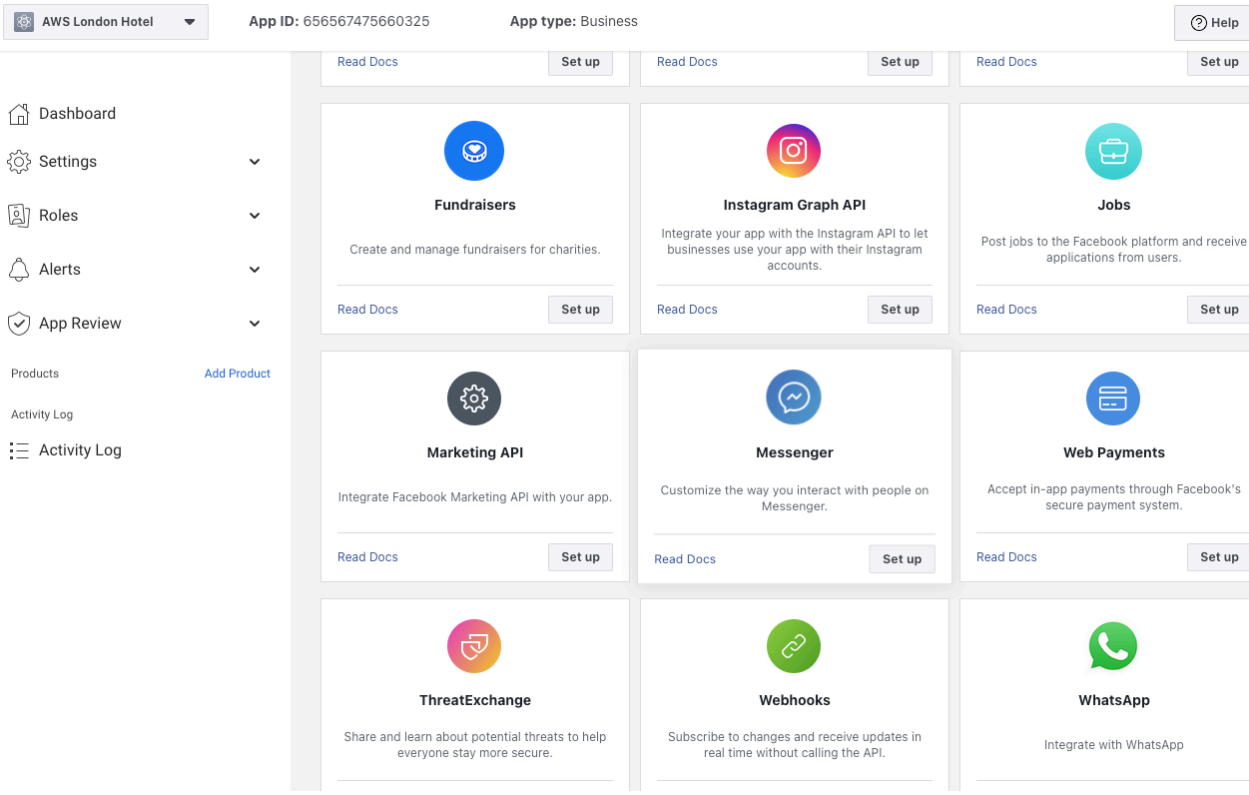
Once you completed the registration, click the **Create App** button.



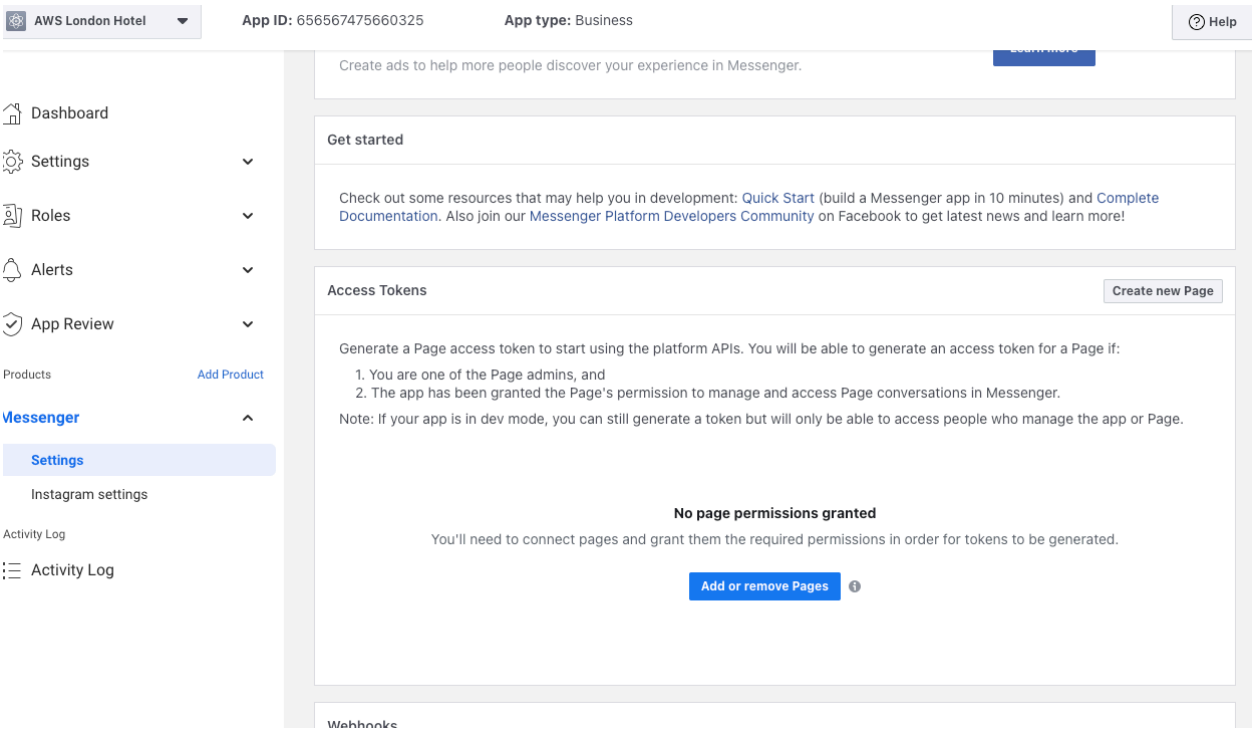
Then choose the app type, for this demo, we will just use Business.



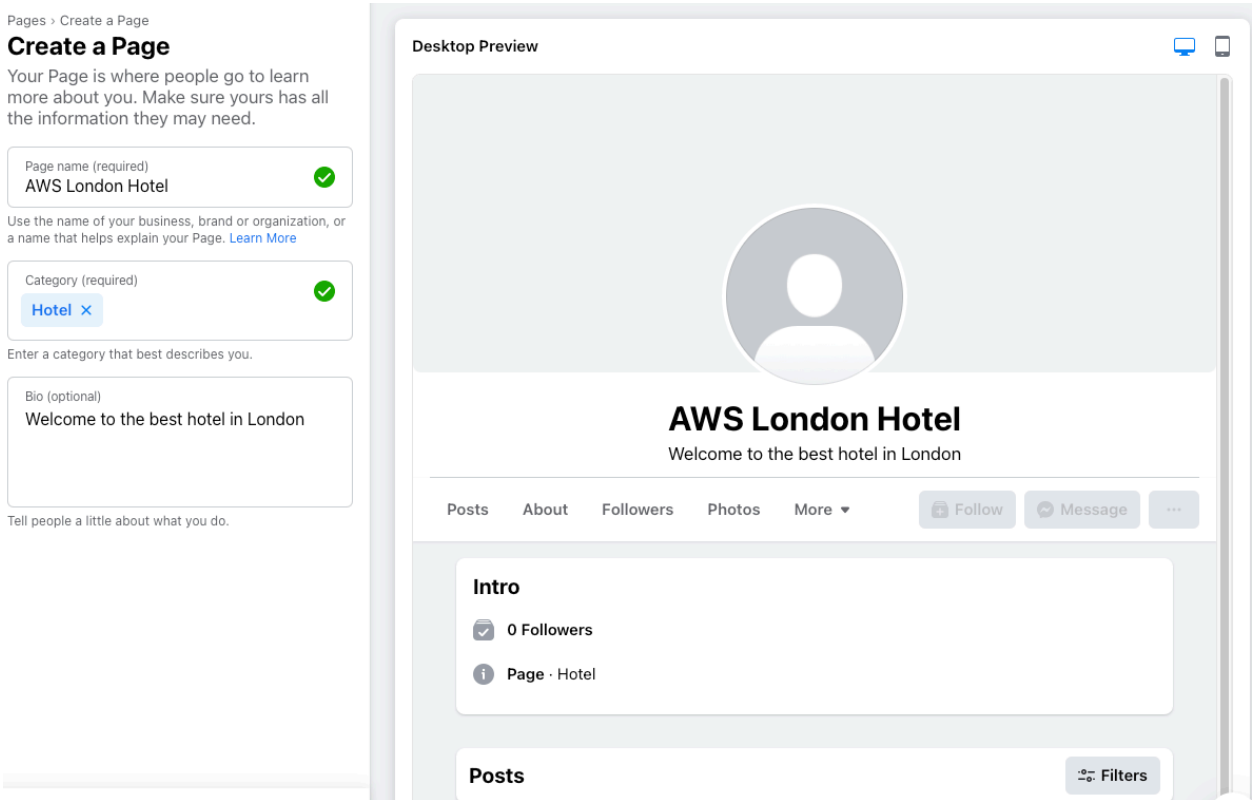
On the next page, put AWS London Hotel as the Display name and your personal email, then click Create app.



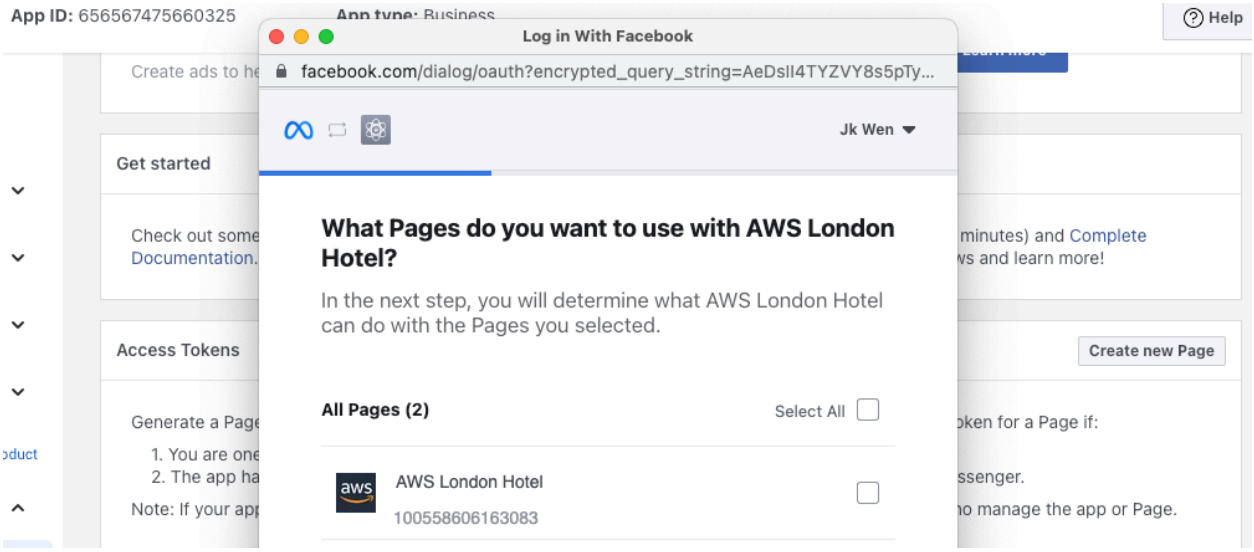
Once the app is created, you will land on this page. Click Set Up under the Messeger box.



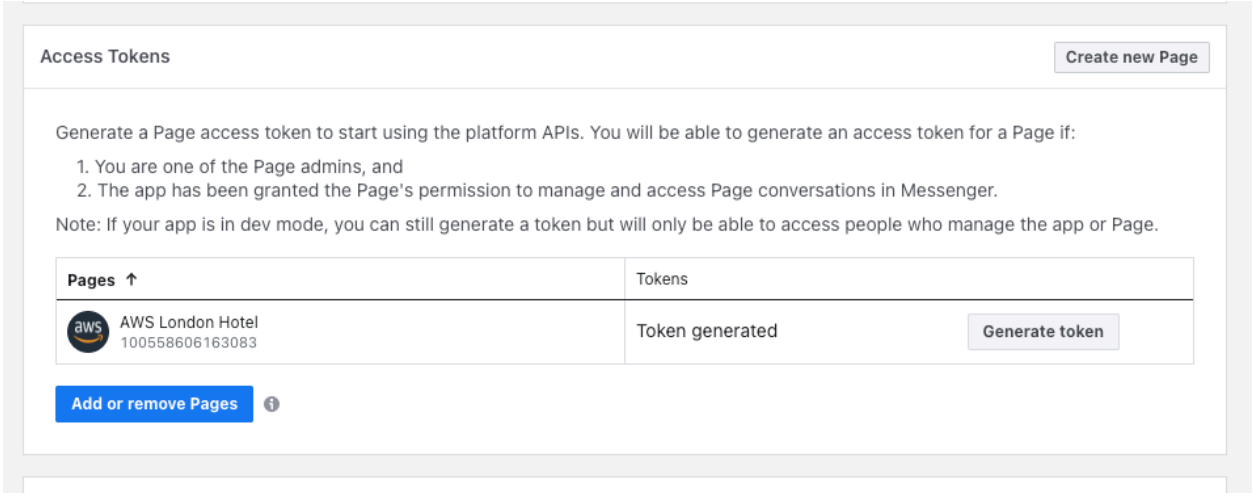
Click Create new Page.



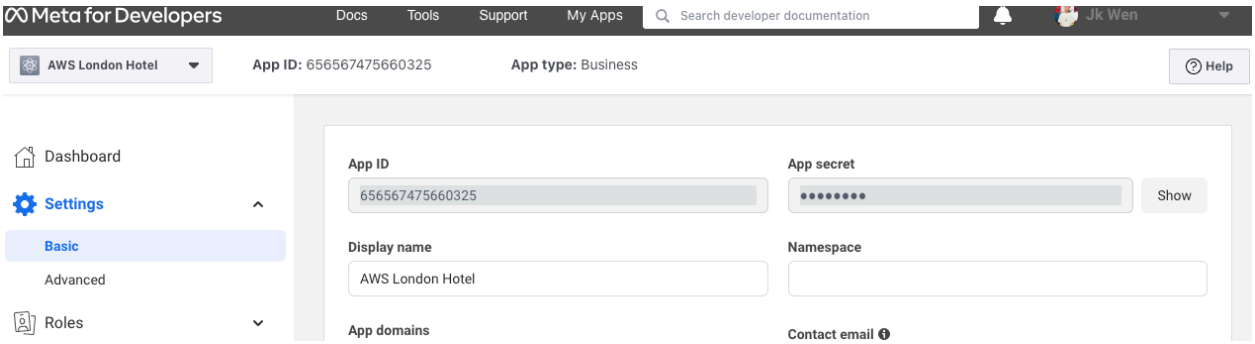
Customize your page with your personal preference.



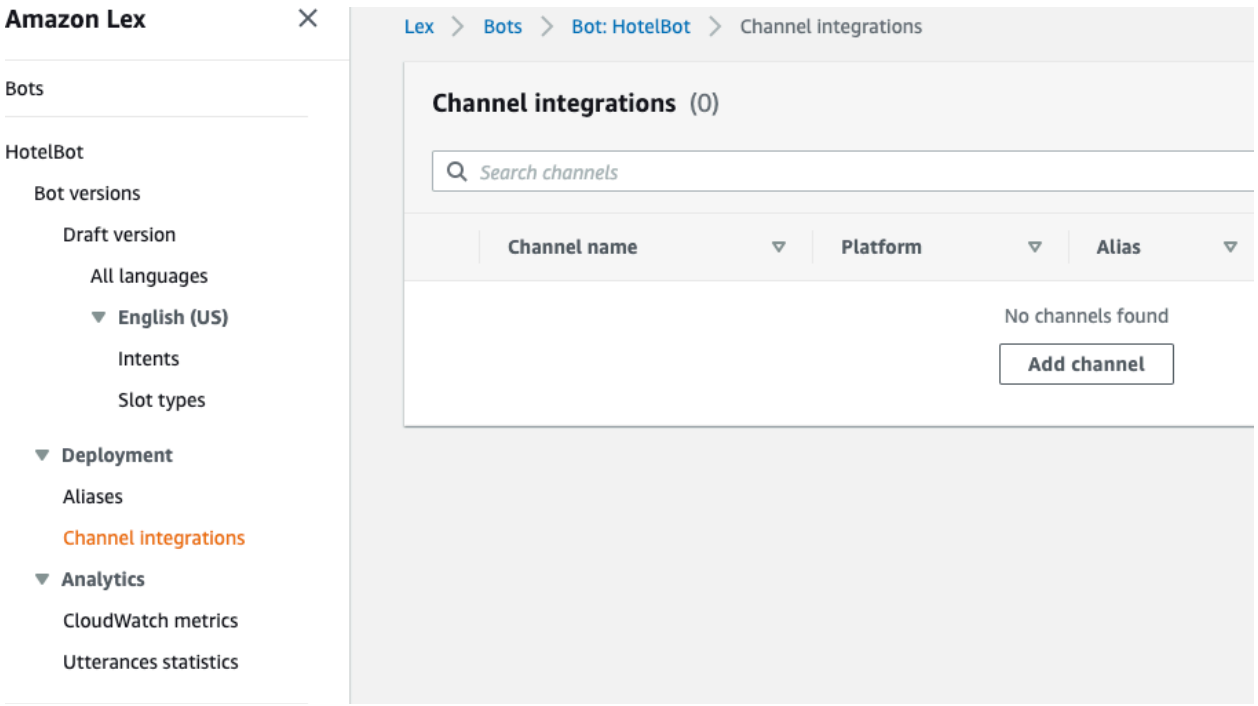
Head back and choose the **Add or remove Pages**, and pick the **AWS London Hotel**.



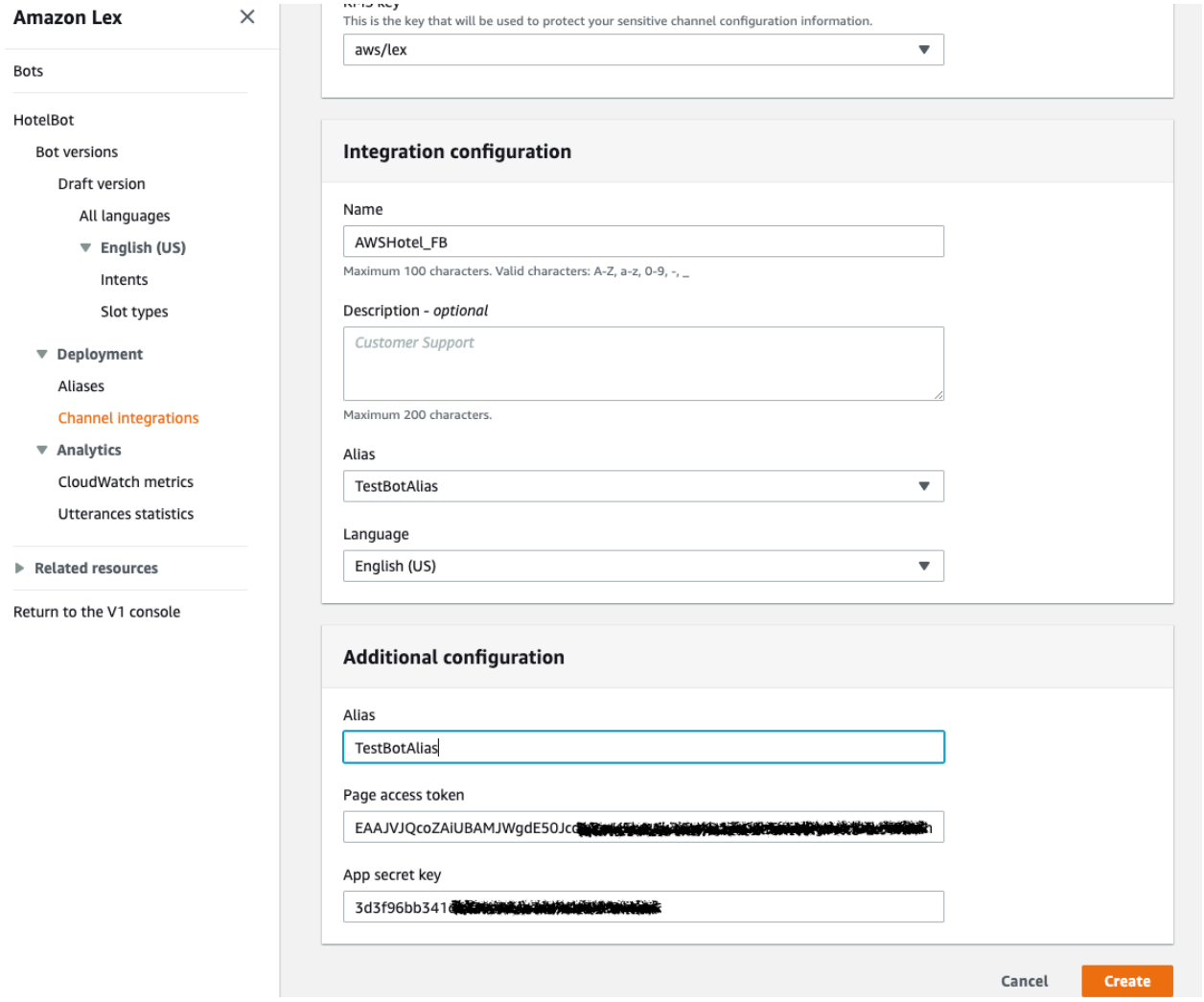
Click **Generate token**.



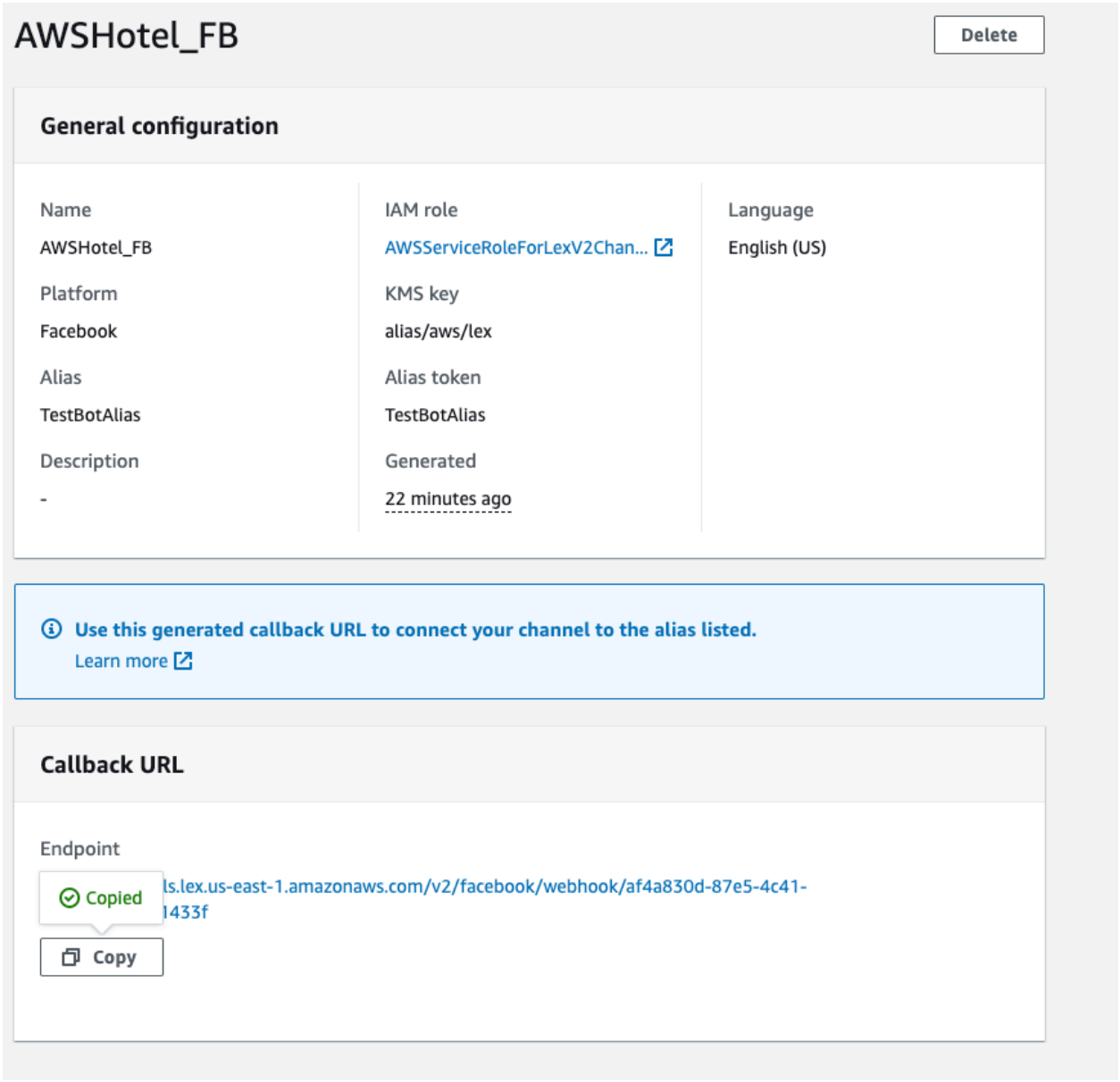
Copy the **App secret key** on the Basic section under Settings.



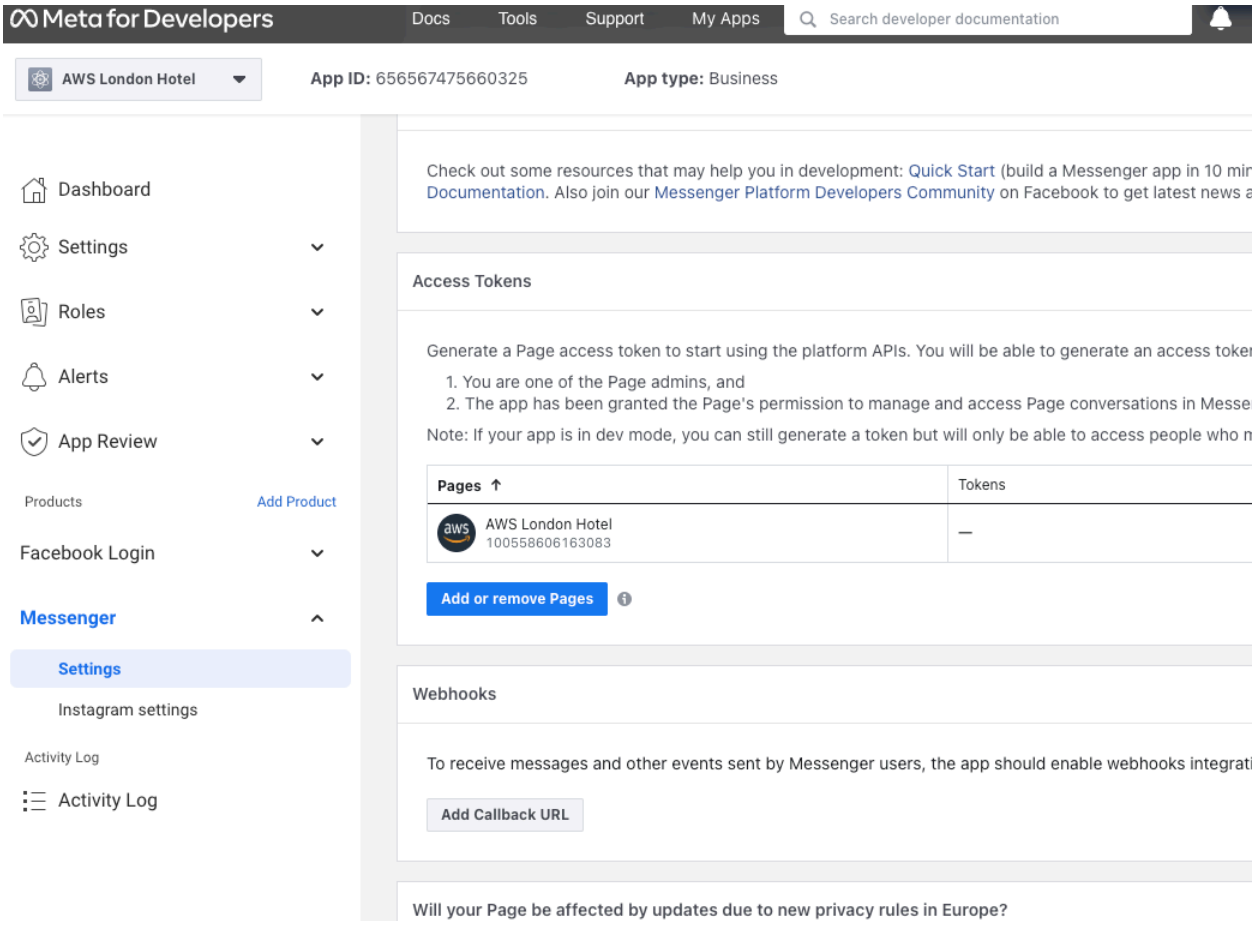
Navigate to **Channel integrations** section, and click **Add Channel.**



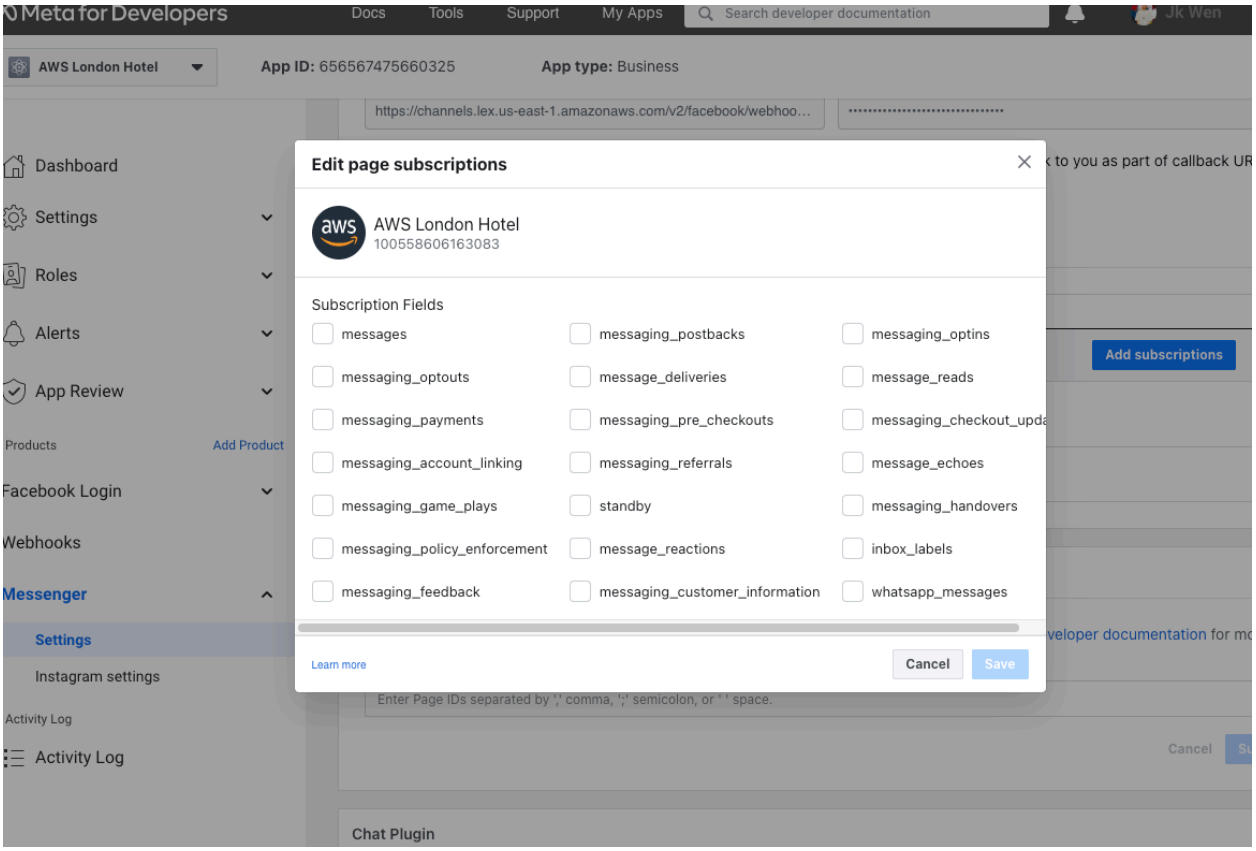
Paste the required information on the channel integration page and click create.



Now copy the endpoint URL.



Click **Add Callback URL** and paste the endpoint URL there.



Click **Add subscription** and check the **messages** box and **Save**. Now navigate to your newly created Facebook page.

Click the Messager feature on your Facebook page and test it out!