



Documentation on AWS Lex & Kendra

Project Teach It

Cloud Computing for IT
Spring 2021

Dharmik Patel
U34190712

Multipurpose-Bot with Amazon Lex

Abstract:

I have built a bot that can serve the purpose of ordering meal as well as getting responses to numerous AWS FAQ's while ordering the food you want! This is done with the help of machine learning technologies of AWS viz. Lex and Kendra. I have created intents in Lex bot to OrderFood, OrderDessert and OrderDrinks as well as used a built-in intent based on AWS Kendra which serves as a powerful search query for fulfilling the requests for AWS FAQ's. This website for this bot is hosted on CloudFront using CloudFormation stack with the required data and project files stored in S3 bucket.

View project here: <https://d2fjjzevnf7r4w.cloudfront.net/index.html>

About Lex:

Amazon Lex is an AWS service for building conversational interfaces into applications using voice and text. With Amazon Lex, the same deep learning engine that powers Amazon Alexa is now available to any developer, enabling you to build sophisticated, natural language chatbots into your new and existing applications. Amazon Lex provides the deep functionality and flexibility of natural language understanding (NLU) and automatic speech recognition (ASR) to enable you to build highly engaging user experiences with lifelike, conversational interactions and create new categories of products. [1]

Bots with Lex:

Amazon Lex enables any developer to build conversational chatbots quickly. With Amazon Lex, no deep learning expertise is necessary—to create a bot, you just specify the basic conversation flow in the Amazon Lex console. Amazon Lex manages the dialogue and dynamically adjusts the responses in the conversation. Using the console, you can build, test, and publish your text or voice chatbot. You can then add the conversational interfaces to bots on mobile devices, web applications, and chat platforms (for example, Facebook Messenger). [2]

Benefits of Lex:

Simplicity, Democratized deep learning technologies, Seamless deployment and scaling, Built-in integration with the AWS platform, Cost-effectiveness

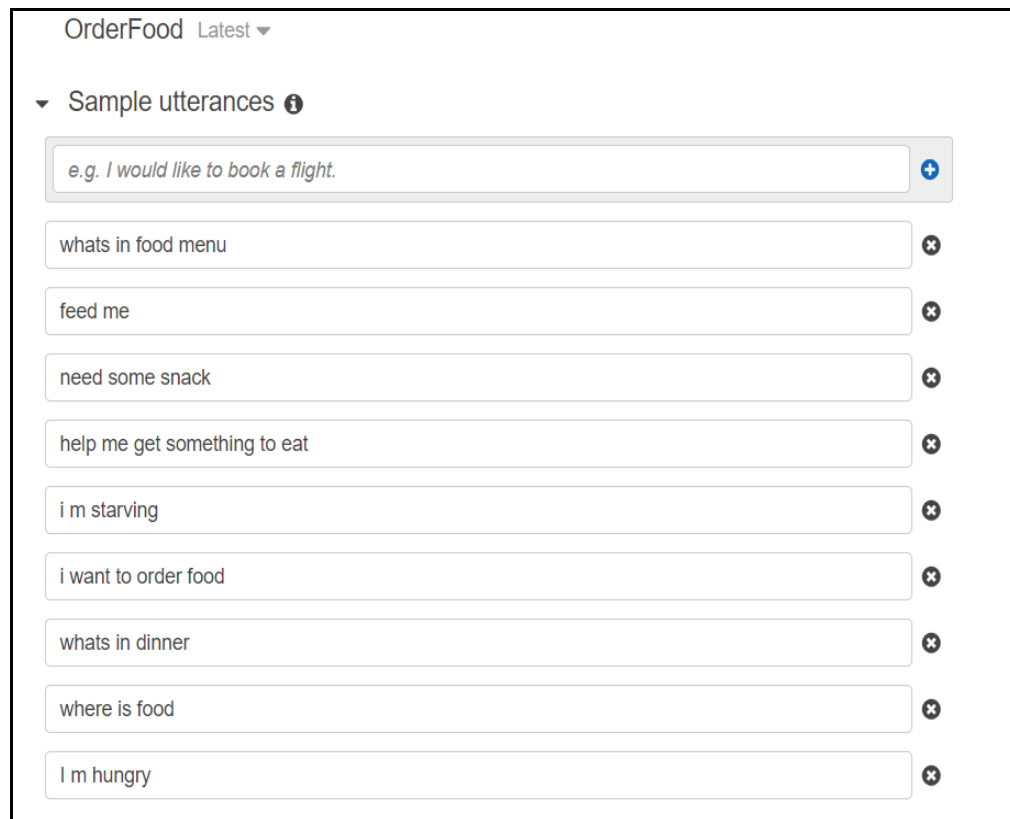
Creating a Bot:

A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities. Each bot must have a unique name within your account. [3]

1. Create a bot and configure it with one or more intents that you want to support. Configure the bot so it understands the user's goal (intent), engages in conversation with the user to elicit information, and fulfills the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.
3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on platforms such as mobile applications or messaging platforms such as Facebook Messenger.

Intent – An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For example, you might create a bot that orders food, dessert and drinks. For each intent, you provide the following required information:

- **Intent name**– A descriptive name for the intent. For example, **OrderFood** or **Order Drink**. Intent names must be unique within your account.
- **Sample utterances** – How a user might convey the intent. For example, a user might say "I am hungry" or "Quench my thirst".



The screenshot shows the Amazon Lex console interface for the 'OrderFood' intent. At the top, the intent name 'OrderFood' is displayed with a 'Latest' dropdown menu. Below this, a section titled 'Sample utterances' with an information icon is expanded. It features a text input field containing the example 'e.g. I would like to book a flight.' with a blue plus icon to its right. Below this field is a list of ten sample utterances, each in a rounded rectangular box with a delete icon (an 'x' in a circle) to its right. The utterances are: 'whats in food menu', 'feed me', 'need some snack', 'help me get something to eat', 'i m starving', 'i want to order food', 'whats in dinner', 'where is food', and 'I m hungry'.

Figure1: Sample utterances

- **How to fulfill the intent** – How you want to fulfill the intent after the user provides the necessary information (for example, place order with a local pizza shop). It is recommended that you create a Lambda function to fulfill the intent. You can optionally configure the intent so Amazon Lex simply returns the information back to the client application to do the necessary fulfillment.

In addition to custom intents such as ordering a pizza, Amazon Lex also provides built-in intents to quickly set up your bot.

Slot – An intent can require zero or more slots or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all *required* slots before Amazon Lex can fulfill the intent. [3]

For example, the OrderFood intent requires slot such as DropOffLocation. In the intent configuration, you add these slots. For each slot, you provide slot type and a prompt for Amazon Lex to send to the client to elicit data from the user. A user can reply with a slot value that includes additional words, such as "at the garage" or "in the front of door" Amazon Lex can still understand the intended slot value.

Slot type – Each slot has a type. You can create your custom slot types or use built-in slot types. Each slot type must have a unique name within your account. For example, you might create and use the following slot types for the `OrderFood` intent:

The screenshot shows the 'Add slot type' dialog box. It has a title bar with the text 'Add slot type' and a close button (X). The main content area is divided into three sections. The first section, 'Slot type name', has a text input field containing 'DropOffLocation'. The second section, 'Description', has a text input field containing 'e.g. Available car types'. The third section, 'Slot Resolution', has two radio buttons: 'Expand Values' (selected) and 'Restrict to Slot values and Synonyms'. Below this is a 'Value' section with a list of values: 'doorstep', 'the lobby', 'garage', and 'frontdoor'. Each value is in a text input field with a delete button (X) to its right. At the bottom of the dialog are three buttons: 'Cancel', 'Save slot type', and 'Add slot to intent'.

Figure 2: Slot type

Amazon Lex also provides built-in slot types. For example, `AMAZON.NUMBER` is a built-in slot type that you can use for the quantity of food ordered.

Attaching slot to intent, reorder and Set Confirmation prompt:

I have used two built-in slot types `AMAZON.Food` for validating an input for the what type of food prompt and `AMAZON.FoodEstablishment` for validating the foodoutlet name prompt. As shown above I have created a custom `DropOffLocation` slot and attached here for the food delivery location prompt.

The confirmation prompt will once again prompt the user to confirm the whole order and let the bot know if “yes” or “no” type of value for placing the order and it will respond according to that input.

Slots

Priority	Required	Name	Slot type	Version	Prompt
		e.g. Location	e.g. AMAZON.US_...		e.g. What city?
1.	<input checked="" type="checkbox"/>	Food	AMAZON.Food	Built-in	What would you like to order?
2.	<input checked="" type="checkbox"/>	Restaurant	AMAZON.FoodEst...	Built-in	From where do you want to order?
3.	<input checked="" type="checkbox"/>	DropOffLoc	DropOffLocation	1	Where do you want your food?

Confirmation prompt

☒ Confirmation prompt

Confirm

Are you sure you want to order {Food} from {Restaurant}?

Cancel (if the user says "no")

No problem. Let me know whenever you are hungry again!

Figure 3: Slots & Confirmation prompt

Response - Types of Messages

A message can be a prompt or a statement.

- A *prompt* is typically a question and expects a user response.
- A *statement* is informational. It doesn't expect a response.

You can set the format of the messages. The format can be one of the following:

- PlainText—The message is in plain UTF-8 text.
- SSML—The message is Speech Synthesis Markup Language (SSML).
- CustomPayload—The message is in a custom format that you specified.

Figure 4 shows one of the response message group I have created for intent OrderFood.

Response

Preview

Message

Custom Markup

One of these messages will be presented at random.

e.g. Thank you. Your {Drink_Name} has been ordered.

Thanks, I hope your {Food} to be delicious.

Order placed! Get ready for your {Food}

Figure 4: Response Message

Response card – A *response card* contains a set of appropriate responses to a prompt. Use response cards to simplify interactions for your users and increase your bot's accuracy by reducing typographical errors in text interactions. You can send a response card for each prompt that Amazon Lex sends to your client application. You can use response cards with Facebook Messenger, Slack, Twilio, and your own client applications.[3]

You can define a response card for the following prompts:

- Conclusion statement
- Confirmation prompt
- Follow-up prompt
- Rejection statement
- Slot type utterances

☒ Enable response card

Card 1 Preview as: Facebook 🗑️

Image URL*

Title*

Subtitle*

Button title* ✕

Button value*

Button title ✕

Button value

Button title ✕

Button value

+ Add Card

Figure 5: Response Card

The response card set in figure 5 will look in the application as figure 6:

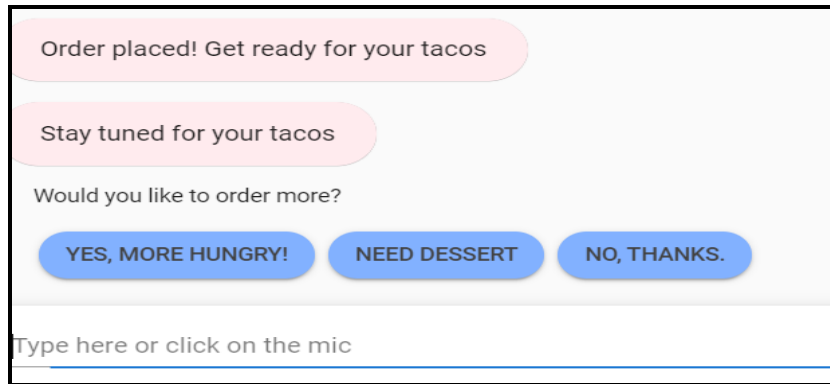


Figure 6

Wait for User reply: Response message for if button value of response card button set to “no”

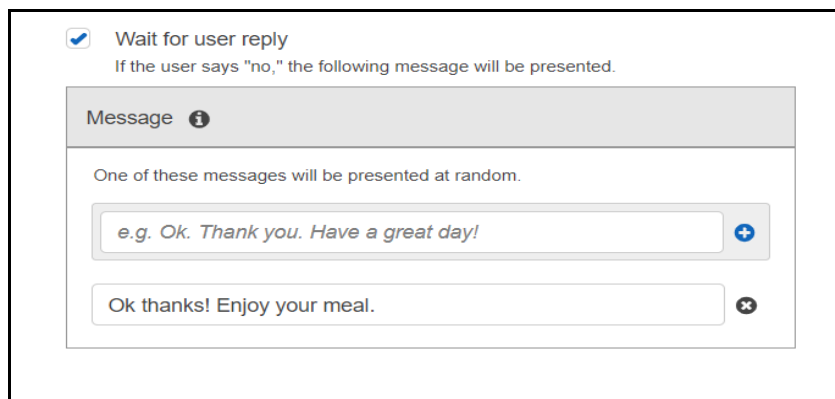


Figure 7: Message for User reply “no”

Error Handling – It is utilized when the bot cannot understand any input or it prompts for an input and the user is not able to provide it during prompt time.

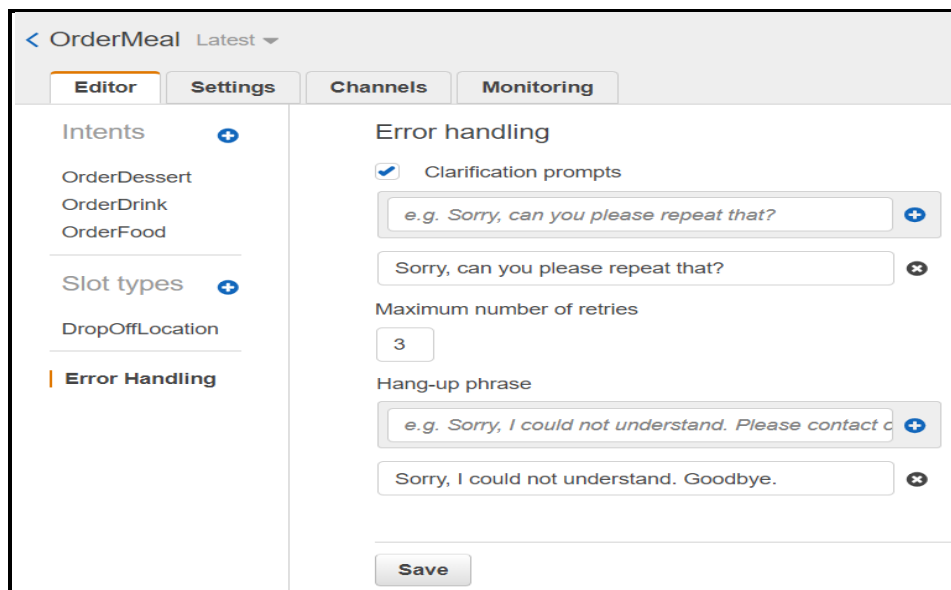


Figure 8: Error Handling

Amazon Lambda with Bots:

Amazon Lex provides pre-built integration with AWS Lambda, and you can easily integrate with many other services on the AWS platform, including Amazon Cognito, AWS Mobile Hub, Amazon CloudWatch, and Amazon DynamoDB. Integration with Lambda provides bots access to pre-built serverless enterprise connectors to link to data in SaaS applications, such as Salesforce, HubSpot, or Marketo. [4]

You can configure your Amazon Lex bot to invoke a Lambda function as a code hook. The code hook can serve multiple purposes:

- Customizes the user interaction—For example, when Joe asks for available pizza toppings, you can use prior knowledge of Joe's choices to display a subset of toppings.
- Validates the user's input—Suppose that Jen wants to pick up flowers after hours. You can validate the time that Jen input and send an appropriate response.
- Fulfills the user's intent—After Joe provides all of the information for his pizza order, Amazon Lex can invoke a Lambda function to place the order with a local pizzeria.

When you configure an intent, you specify Lambda functions as code hooks in the following places:

- Dialog code hook for initialization and validation—This Lambda function is invoked on each user input, assuming Amazon Lex understood the user intent.
- Fulfillment code hook—This Lambda function is invoked after the user provides all of the slot data required to fulfill the intent.

Experimenting with Lambda:

Create lambda blueprint Booktrip: For demonstration purposes I have used an inbuilt lambda function which is made for lex using python programming.

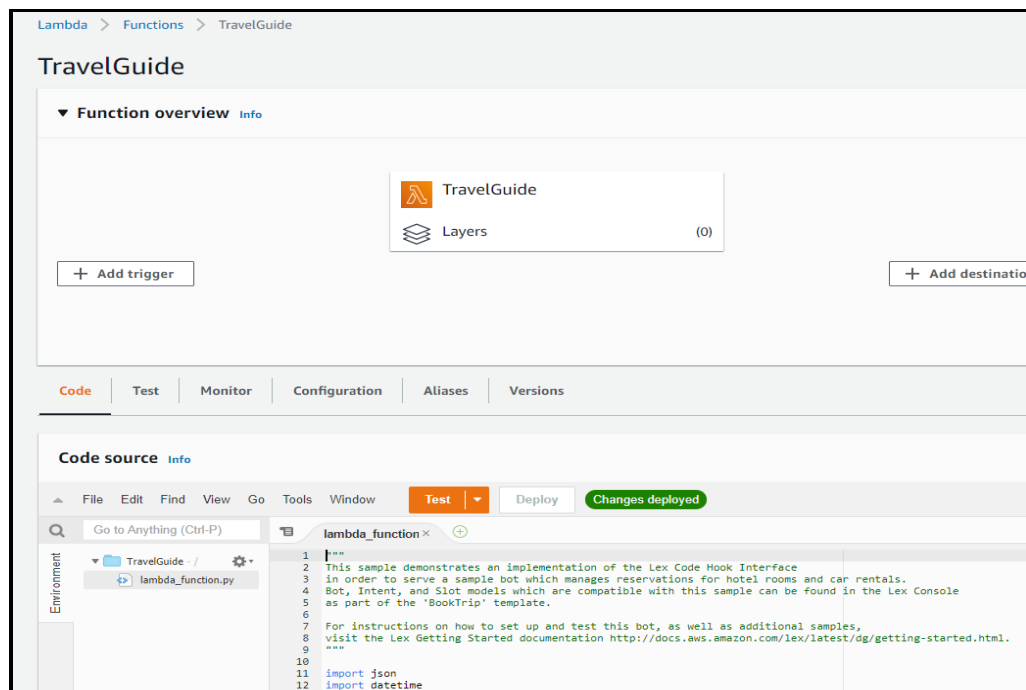


Figure 9: Built-in Lambda for book-trip

Create template bot Booktrip – Also I have used an inbuilt template “Booktrip” for creating a bot that has two built-in intents BookCar and BookHotel as below.

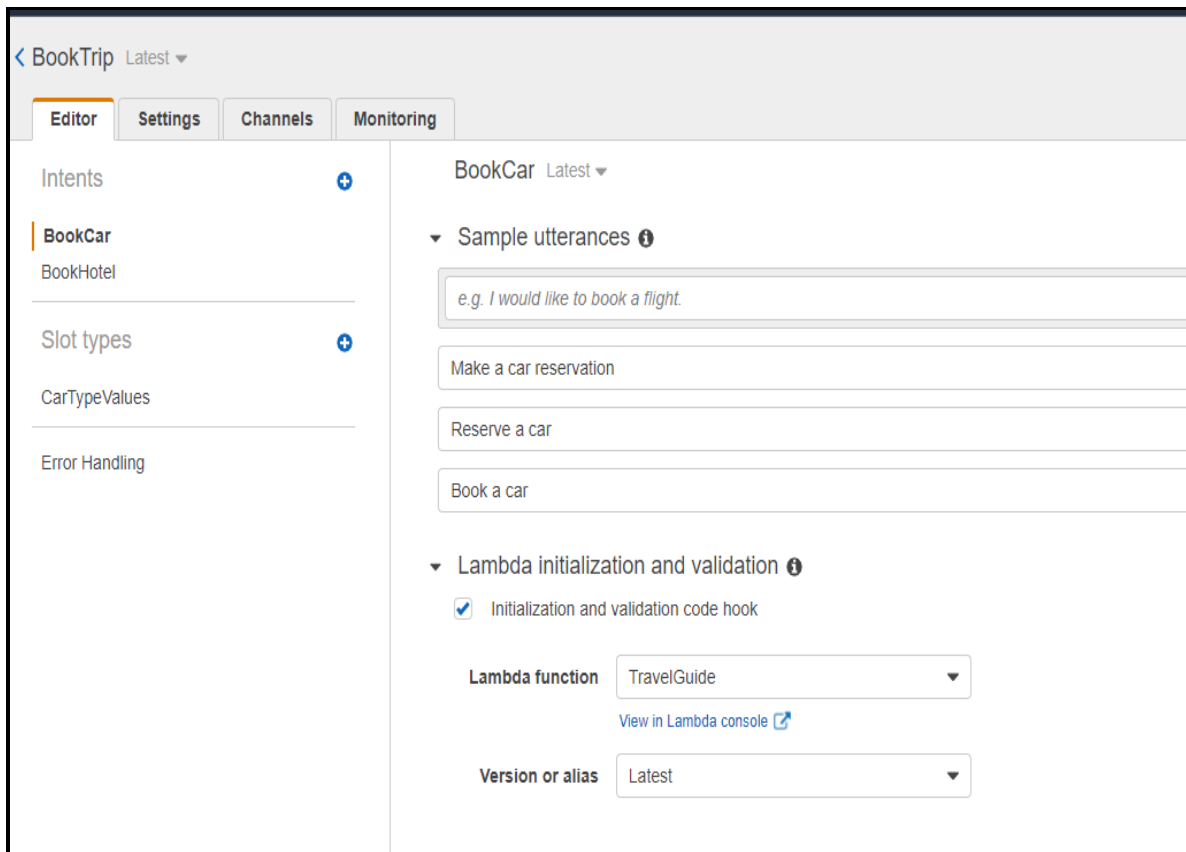


Figure 10: Template bot Book-trip

Lambda initialization – To invoke lambda from lex intent you have to check the Lambda initialization and provide lambda function you want to invoke as below.

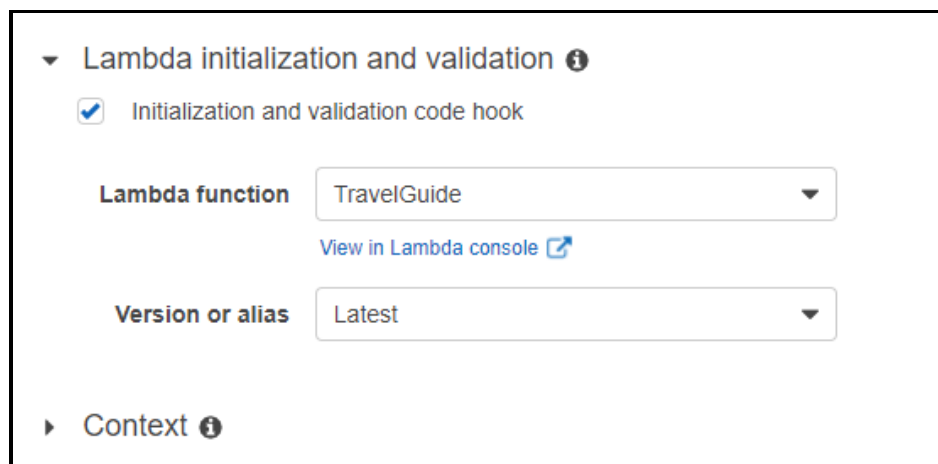


Figure 11: Lambda initialization and validation

Lambda fulfillment – This setting provides you to control the fulfillment of parameters to be handled by client or lambda function.

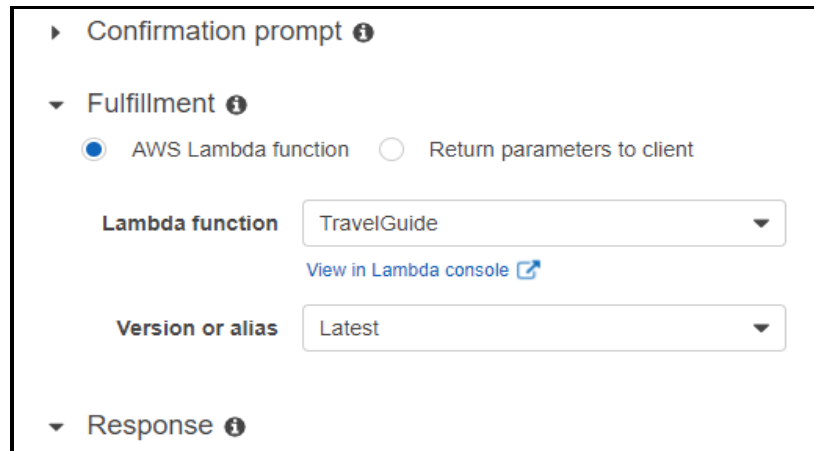
The image shows a configuration panel for Lambda fulfillment. It has three main sections: 'Confirmation prompt' (expanded), 'Fulfillment' (expanded), and 'Response' (collapsed). In the 'Fulfillment' section, there are two radio buttons: 'AWS Lambda function' (selected) and 'Return parameters to client'. Below these are two dropdown menus: 'Lambda function' with 'TravelGuide' selected, and 'Version or alias' with 'Latest' selected. A link 'View in Lambda console' with an external link icon is positioned between the two dropdowns.

Figure 12: Lambda Fulfillment

Observation: Before and after using lambda for this bot it was observed that invoking lambda provided better **validation, active sessions and session sharing between intents**.

About Kendra:

Amazon Kendra is a highly accurate intelligent search service that enables your users to search unstructured data using natural language. It returns specific answers to questions, giving users an experience that's close to interacting with a human expert. It is highly scalable and capable of meeting performance demands, tightly integrated with other AWS services such as Amazon Lex, and offers enterprise-grade security.[5]

Amazon Kendra users can ask the following types of questions, or queries:

- **Factoid questions** — Simple who, what, when, or where questions, such as *Who is on duty today?* or *Where is the nearest service center to me?* Factoid questions have fact-based answers that can be returned in the form of a single word or phrase. The precise answer, however, must be explicitly stated in the ingested text content.
- **Descriptive questions** — Questions whose answer could be a sentence, passage, or an entire document. For example, *How do I connect my Echo Plus to my network?* or *How do I get tax benefits for lower income families?*
- **Keyword searches** — Questions where the intent and scope are not clear. For example, *keynote address*. As 'address' can often have several meanings, Amazon Kendra can infer the user's intent behind the search query to return relevant information aligned with the user's intended meaning. Amazon Kendra uses deep learning models to handle this kind of query.

How it works? - Amazon Kendra provides an interface for indexing and searching documents. You can use Amazon Kendra to create an updatable index of documents of a variety of types, including plain text, HTML files, Microsoft Word documents, Microsoft PowerPoint presentations, and PDF files. It has a search API that you can use from a variety of client applications, such as websites or mobile applications. [5]

Amazon Kendra has the following components:

- The *index*, which provides a search API for client queries. You create the index from source documents.
- A *source repository*, which contains the documents to index.
- A *data source* that syncs the documents in your source repositories to an Amazon Kendra index. You can automatically synchronize a data source with an Amazon Kendra index so that new, updated, and deleted files in the source repository are updated in the index.
- A *document addition API*, that adds documents directly to the index.

To manage indexes and data sources, you can use the Amazon Kendra console or the API. You can create, update, and delete indexes. Deleting an index deletes all data sources and permanently deletes all of your document information from Amazon Kendra.

Test: You can test your Kendra search after creating index and providing data in FAQs with the help of **Search Console** provided in the Amazon Kendra menu.

Overview of Project:

Create Kendra index – Here in my project I have used Kendra for search query regarding AWS' FAQ's for which I have created an index "agentreply" as shown below. [6]

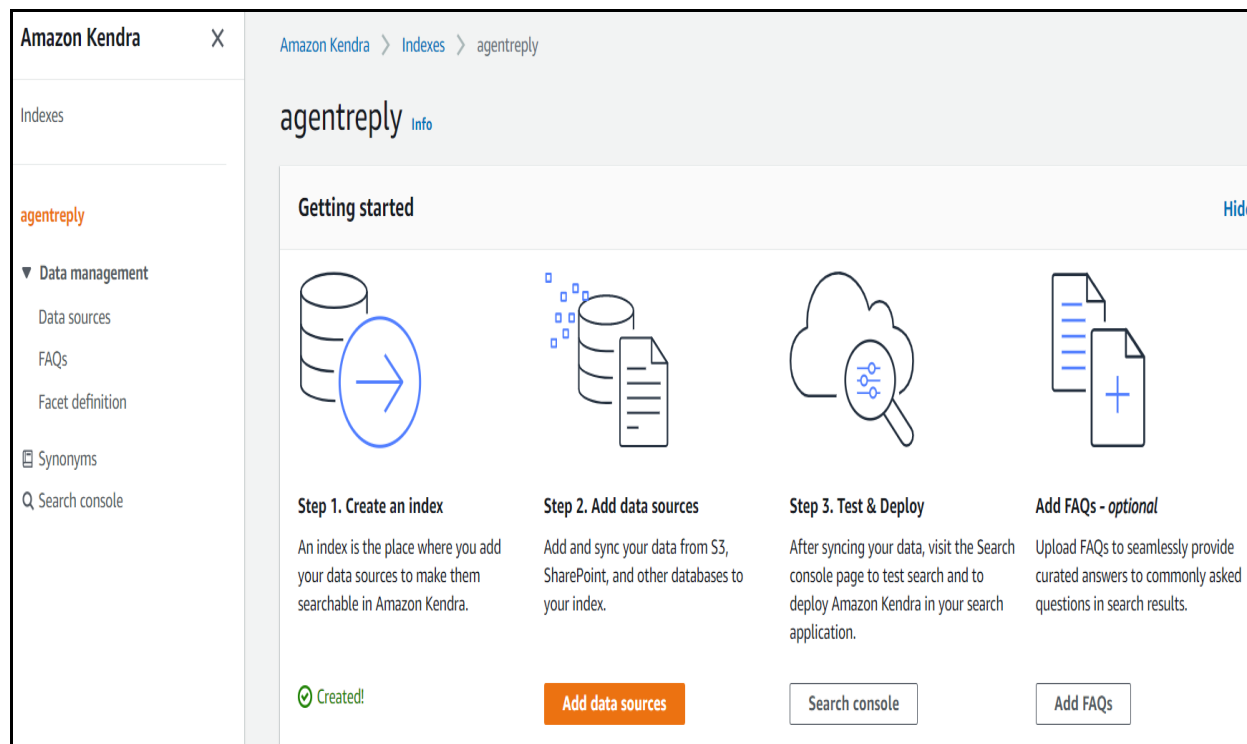


Figure 13: Template bot Book-trip

Create FAQs: I have used a .csv file which simply has two columns filled, one with questions and other with answers for that question and the Kendra indexes will automatically synchronize with the csv file stored in S3 bucket who's address(link) is provided in for FAQs here named as "projectguide".

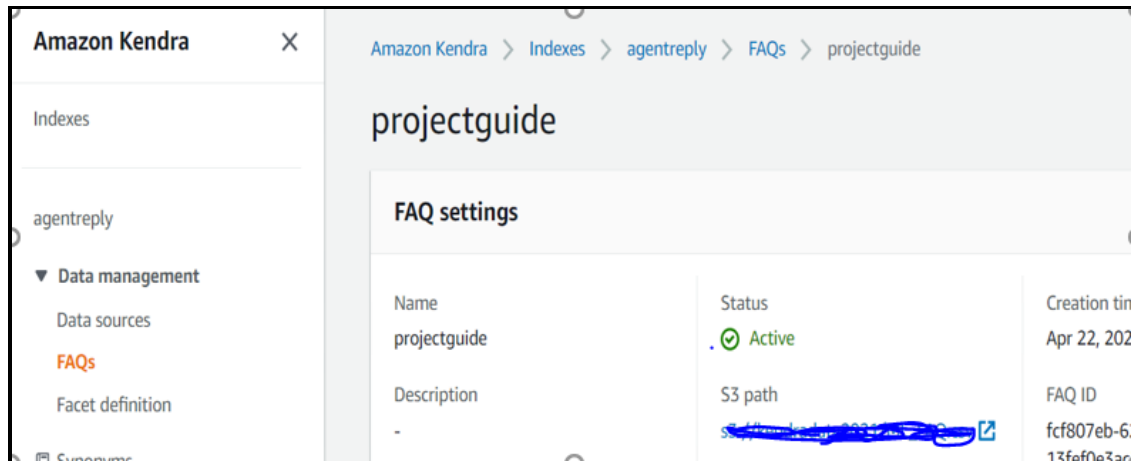


Figure 14: Template bot Book-trip

Create Lexbot – I have created custom bot with 3 custom intents to order food, dessert and drink as explained above. Also, I have used a built-in intent (**AMAZON.KendraSearchIntent**) that will search the FAQs for our bot requested inputs and return back to the bot and displayed on Bot-UI. In the response section of Kendra intent, I have displayed the message like this below which will return and fulfill the appropriate requests.

"I found an answer for the customer query: ((x-amz-lex:kendra-search-response-question_answer-question-1)) and the answer is ((x-amz-lex:kendra-search-response-question_answer-answer-1))" [7]

Cognito pool id - Amazon Cognito offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.[8] Thus I have created a User Pool Id for my account that will be used as a parameter in root stack of CloudFormation while publishing the bot.

Providing UI to our lex bot and publishing it via CloudFormation through AWS guide [9]

Here I have used the AWS Configured CloudFormation Stack for hosting the Lex Bot with JavaScript based user interface and I have set the parameters set to point to my existing bot "**AgentAssistBot**" created by following the steps as above as well as used my generated Cognito User Pool Id to grant access to AWS services. On clicking the Launch Stack button for hosting in the website in the region of your choice will open up a CloudFormation stack launch page with some prefilled parameter values and some empty cells. You will have to set the Bot Name, Bot Alias for your bot and the Cognito User Pool ID for your account. Launching this will result in forming a root stack and a nested stack that will be responsible for deploying your bot and hosting it online using CloudFront. Also, some buckets are created automatically in S3 where the project files are stored.

The stack deploys the following architecture. It is completely serverless - charges are based on your service usage.

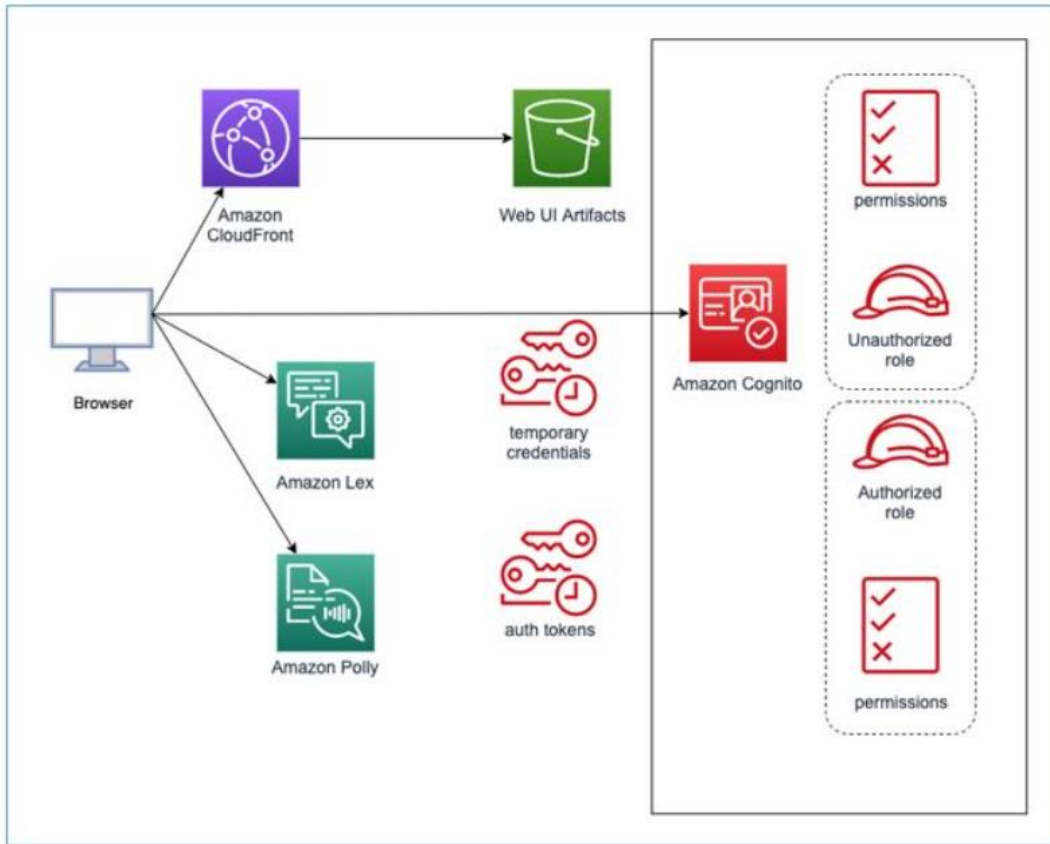
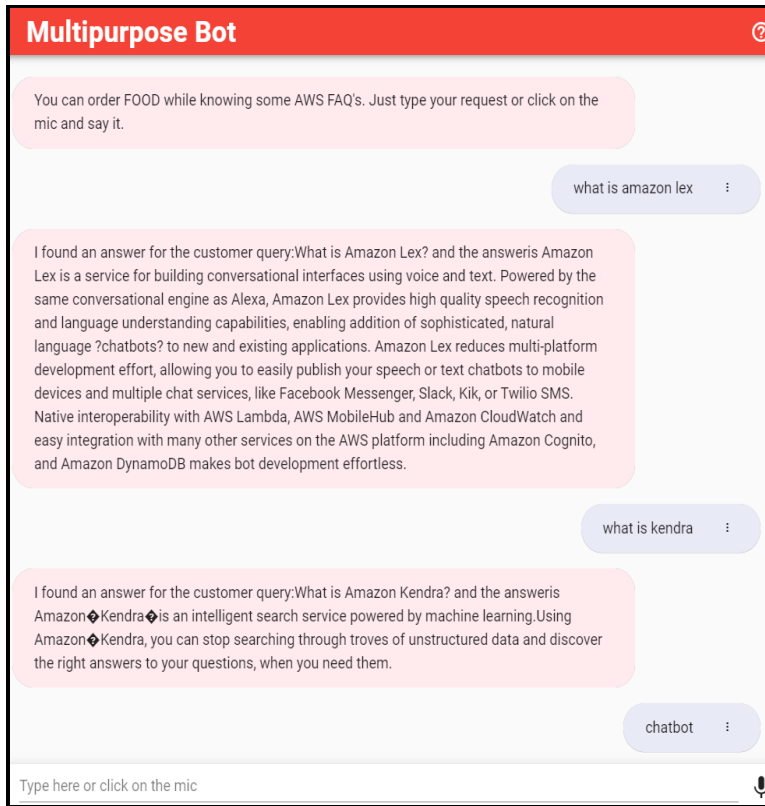


Figure 15: Bot Deployment Architecture

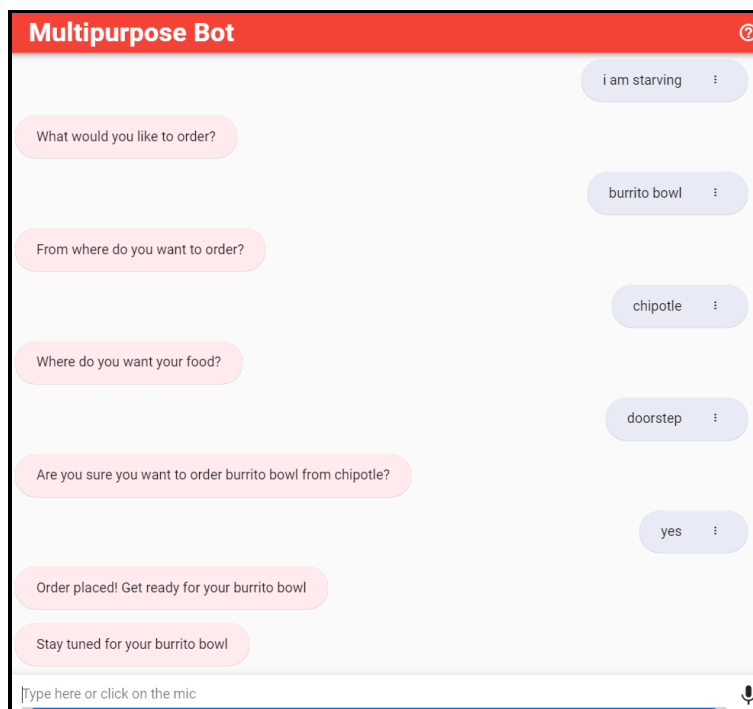
What will happen when you will interact with the bot?

Scenario1: If your query is regarding any AWS Machine learning FAQ's it will invoke the used inbuilt Kendra search query intent and the Kendra will call the index to return the answer for query in response to the question if found in FAQs that point to the .csv file dataset stored in S3 bucket. The response from Kendra to Lex will be captured in the response message of Kendra intent as stated above and which will ultimately show up on our JavaScript based Bot user interface given access to grant those services using generated Cognito pool ID from my AWS account.

Scenario2: If you will call the bot to order food for you then this query will invoke one of the OrderFood/OrderDrink/OrderDessert intents as per your request. The invoked intent would now act up and interact with you regarding your order and place the order after confirming the whole order with you. Here the fulfillment parameters are returned to the client which are displayed on our hosted CloudFront website.



Scenario 1: Kendra intent invoked



Scenario 2: OrderFood intent invoked

Try some example inputs on bot:

“What topics Dr.V taught in class ?”

“Who is the TA for class?”

“Who made this project?”

Any FAQ related to ML in AWS for e.g. “what is deep learning” or “what is deeplens”

“I am hungry” or “Need some snacks” or simply “order food”

“I have a sweet tooth” or “Need something sweet“

“I am thirsty” or “need something to drink”

“Turn on music” or “play my jam”

References:

- [1] <https://docs.aws.amazon.com/lex/index.html>
- [2] <https://docs.aws.amazon.com/lex/latest/dg/what-is.html>
- [3] <https://docs.aws.amazon.com/lex/latest/dg/how-it-works.html>
- [4] <https://docs.aws.amazon.com/lex/latest/dg/programming-model.html>
- [5] <https://docs.aws.amazon.com/kendra/latest/dg/what-is-kendra.html>
- [6] <https://docs.aws.amazon.com/kendra/latest/dg/create-index.html>
- [7] <https://docs.aws.amazon.com/lex/latest/dg/agent-step-3.html>
- [8] <https://console.aws.amazon.com/cognito/home>
- [9] <https://aws.amazon.com/blogs/machine-learning/deploy-a-web-ui-for-your-chatbot/>