Programming Assignment: Machine Problem 3: AWS Lex & Lambda

Graded · 1/1 points

**Deadline**

Pass this assignment by Feb 21, 12:59 AM EST

InstructionsMy submissionsDiscussions

**MP3\_FAQs.pdf**PDF File

[Open file](https://d18ky98rnyall9.cloudfront.net/GEPWqx5bSauD1qseWxmrzg_a0538e3741ec41d4b5e75c5051f429f1_MP3_FAQs.pdf?Expires=1645488000&Signature=gz0jbGyK-Xc4QM6~8TrAiQSvV8MsoXRXpyNhLfdmO4g9s6ZqFkNfam1AhwtVVinFiKtUzLV20WBfQqHRWUYzUM4WWBUQ1eM5mjCfzaMj4L0wu21RiPhMKpwD4Xv2EsyRCScrvGraACzYmrGbvE~XkRfP~2B-iUKUXGHUPtBzAKo_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

**1. Overview**

In this MP, we build a chatbot that returns the shortest distance between two cities/nodes in a directed graph, where all edges weigh 1.

**2. Requirements**

You need a valid AWS account and work on **Lambda**, **API Gateway**, **DynamoDB**, **Cognito**, and **Lex**. Also, you need to be familiar with one of the following programming languages for implementing lambda: Python / Javascript / Java / Go. While we will attempt to support you irrespective of your chosen language, we can best assist with Python.

**Note**: we suggest you create all of the services in the zone**us-east-1** to prevent any unexpected issues from the autograder.

**3. Procedure**

**3.1. AWS Graph Creator Lambda Function:**

You need to write a program and create a POST REST API (using AWS API Gateway and Lambda) to take a graph and store it in the DynamoDB database. The following would be an example specification for a graph that your function should accept in the body of the POST request:

{"graph": "Chicago->Urbana,Urbana->Springfield,Chicago->Lafayette"}

Here, a directed edge goes from Chicago to Urbana, Urbana to Springfield, and Chicago to Lafayette. Your lambda function needs to parse this graph, compute the shortest distance using BFS (Breadth-First Search) between all vertices, and store this information in DynamoDB. If successful, return HTTP status code 200. Make sure you delete all items in the respective table of your database to avoid reading stale data.

Your solution should include a table in DynamoDB that will contain the source, destination, and distance attributes. While parsing the graph, your solution should populate this table with the locations and the distances between them. Your chatbot will retrieve these values in the next stage. The following AWS official documentation will help you get started with this:

<https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-develop.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.html>

You may also find this unofficial tutorial helpful: <https://medium.com/accenture-the-dock/serverless-api-with-aws-and-python-tutorial-3dff032628a7>

Check out the [code examples](https://boto3.amazonaws.com/v1/documentation/api/latest/guide/examples.html) of AWS SDK for Python (Boto3) to create, configure, and manage AWS services, you will use this library in this and other MPs.

**Note**: The autograder uses the Python [requests](https://requests.readthedocs.io/en/master/) library to send a POST request. Once you have set up the API and Lambda functions, check if you can successfully update the contents of DynamoDB after sending a sample POST request through the requests library.

[Cloudwatch](https://docs.aws.amazon.com/lambda/latest/dg/python-logging.html) is a valuable tool that lets you observe logs from your lambda function.

**3.2 AWS Lex:**

**Important Note: Please use and follow AWS Lex V1 document and console to do the MP. Otherwise, you might get errors from the autograder.**

In this step, you need to create a chatbot using AWS Lex. Lex is an AWS service for building conversational interfaces for interactive voice and text applications. An excellent way to get started with Lex is to read Amazon's official documentation:

<https://docs.aws.amazon.com/lex/latest/dg/ex1-sch-appt.html>

You need to create a chatbot that can decipher the name of the two cities from text and provide the distance. The interaction with the chatbot will be of the following type:

1. User: "What is the distance from Chicago to Springfield?"

Reply from chatbot: "2"

2. User: "I need to find the distance between two cities?"

Reply from chatbot: "Source?"

User: "Chicago"

Reply from chatbot: "Destination?"

User: "Urbana"

Reply from chatbot: "1"

The autograder uses the above utterances to prompt lex, so please make sure you set the utterances correctly. For slot types, please choose AMAZON.US\_CITY.

Give your bot an [alias](https://docs.aws.amazon.com/connect/latest/adminguide/tutorial1-lex-bot-publish.html) before you publish it.

**3.3 Lex with Lambda:**

You need to link the previously created chatbot to an AWS Lambda function which, when triggered, retrieves the shortest distance between two nodes in the graph from the database and returns the result to the chatbot. You can link a lambda function to a chatbot intent by selecting the fulfillment tab and the appropriate lambda function.

You can learn more about the input and response format from AWS Lex to Lambda from:

<https://docs.aws.amazon.com/lex/latest/dg/lambda-input-response-format.html>

For testing, you can pass in a random graph using the API created in 3.1 and interact with Lex to see if the end-to-end flow works. After you build the bot, you need to publish it and give it an alias name. Please note down the alias name you provide as it needs to be passed to the autograder.

Here is a helpful unofficial tutorial that you may find beneficial: <https://chatbotsmagazine.com/quick-start-develop-a-chat-bot-with-aws-lex-lambda-part-1-b6f7c80ebba6>

**3.4 Deploying Lex:**

You need to deploy your Lex bot to make it publicly accessible. There are many ways to do this. We will use AWS Cognito Identity Pool in this assignment. The "Setup Amazon Cognito" section from the following link shows how you can do this.

<https://aws.amazon.com/blogs/machine-learning/greetings-visitor-engage-your-web-users-with-amazon-lex/>

After following the tutorial, you should have an identity pool id which our autograder will use to verify the functionality of your Lex.

**4. Checklist**

Before submitting the assignment, it would be good to check if you have successfully configured the following services.

* Two Lambda functions: one for handling graph and DynamoDB, and one for Lex
* DynamoDB table configuration
* Lex configuration
* Cognito identity pool configuration

**5. Submission**

Add the necessary info in the test.py file attached below in the payload section. If all the test cases pass, you will see your grade on Coursera.

**MP Files:**