Machine Problem 3: AWS Lex V2 & Lambda

In this week, we focus on the topic of Serverless ...

6 分鐘的閱讀時間 · 查看原始頁面

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 v2**. 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. In each lambda function call, make sure you delete all items first in the respective table of your database before storing the new graph 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 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 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 is a valuable tool that lets you observe logs from your lambda function.

3.2 AWS Lex:

Important Note: Please use and follow the AWS Lex V2 document and console to do the MP. Otherwise, you might get errors from the autograder.

In this step, you need to create a chatbot with support for English 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/lexv2/latest/dg/building-bots.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.City.

Give your bot an Alias and save the alias ID. By default, Amazon Lex v2 creates an alias called TestBotAlias.

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.

The following article shows how the lex bot can be hooked up with the lambda function. You can also learn more about the input and response format from AWS Lex to Lambda in this article:

https://docs.aws.amazon.com/lexv2/latest/dg/lambda.html

https://docs.aws.amazon.com/lexv2/latest/dg/lambda-attach-console.html

Make sure to enable the Lambda code hook for Fulfillment, not for Initialization and Validation. Once you have created the bot and hooked it up with the lambda function, you need to create a bot version and a bot alias. Then you will have to associate the bot version with the bot alias.

You can see the following articles to learn more about versions and aliases for a lex bot:

https://docs.aws.amazon.com/lexv2/latest/dg/versions-aliases.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 building the bot, you need to give it an alias name. Please note down your bot ID in bot details and alias ID in alias details as they will be passed to the autograder.

Here is a helpful official tutorial with a step-by-step example that you may find beneficial. Appendix B towards the end of the article shows how to hook up lambda to your lex bot. https://aws.amazon.com/blogs/machine-learning/creating-a-bankingbot-on-amazon-lex-v2-console-with-support-for-english-and-spanish/

3.4 Deploying Lex:

You need to deploy your Lex bot to make it publicly accessible. There are numerous approaches that can be taken. We will use AWS Cognito Identity Pool in this assignment. The "Setup Amazon Cognito" section of the following link demonstrates how you can do this. Pay attention to the needed IAM roles in this section. Note that the "Build an Amazon Lex bot" section works with Lex v1, which isn't

used in the MP3, but the Cognito Identity Pool works with both Lex v1 and Lex v2.

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. (Optional) Lambda function Cloudwatch Debugging Logs

To view logs using the Lambda console

1. Open the Functions page of the Lambda console.
2. Choose a function.
3. Choose **Monitor**.
4. Choose **View logs in CloudWatch**.

For the AWS CLI usages, please refer to the official document.

5. Checklist

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

6. 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:

7. FAQs and Resources

1. Differences between Lex v1 and v2:

https://docs.aws.amazon.com/lexv2/latest/dg/understanding-new-flows.html

2. What about the situation of both source and destination as the same city. For example: source: Chicago, destination: Chicago? Should we return -1?

You can return 0. Treat the graph as directed, and return -1 for invalid pair, 0 for identical source and destination.

3. For BFS, do you know if those paths are directional or bidirectional?

The graph is directional. You can return -1 for source - destination pair.

4. A generic confusion- The graph doesn't need to be completely connected.

Please read the assignment instructions on how to handle this case.

5. Problem with GraphAPI Submission url:

It’s the API gateway uRL, no “arn-” just URL used to POST graph.

6. Test.py issue:

Leaving out 'https://' in 'test.py' for 'graphAPI'. Add it, test.py will pass.

7. Use the Lambda console and Postman to test your lambda functions before running test.py.

8. If you use a Test event, then the double quotes in the body value have to be properly escaped. Ex:"body": "{\"graph\": \"city100\"}"API should return status code: 200

9. test.py expects POST response code to be 200.

10. If your lambda function is taking longer than 3 seconds to execute, it might give timeout errors. The default lambda timeout setting is 3 seconds.

11. Make sure you grant necessary Read/Write permissions in the inline policy for DynamoDB.

12. When adding a trigger to the first lambda function as an API Gateway, choose ‘Security’ as “Open” .

13. In Lex, make sure to separate the slot names with a whitespace character.

14. When adding the slot, select the checkbox for “Required for this intent” option.

15. Write both your prompts/utterances on different lines, “Preview” them to see they look

correct.

16. You can find the “Fulfillment Lambda code hook” menu under the “Fulfillment Advanced Options”.

17. Make sure to test your Lex bot. Go to the new intent you create, open the intent editor and there you’ll find a test button.

18. CloudWatch is indeed your friend, try to debug using the logs.

19. AccessDeniedException - dynamoTable.scan(ProjectionExpression='Distance')['Items']

Try to give full access permission and check once

20. MP3 IdentityPool error:

Change the location from us-west-2 to us-east-1 (Basically the region autograder). It might fix.

Resources

1. Graph (Directed graph)

https://www.python.org/doc/essays/graphs/

https://medium.com/@yasufumy/algorithm-breadth-first-search-408297a075c9

2. Use requestUnicorn.js as example for your Lambda function - Step 3 -

https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/module-3/

3. Writing to DynamoDB

https://boto3.amazonaws.com/v1/documentation/api/latest/guide/dynamodb.html

4. Allow Lambda to execute DynamoDB

https://docs.aws.amazon.com/lambda/latest/dg/with-ddb-example.html

5. You can quickly test your bot without any Lambda function (try this first if you are new to Lex)

For quick testing, you can also create a simple Lambda handler. Lex-Lambda expects request/response JSON in a certain format. Refer here:

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