**Chatbot using Amazon Lex and store bot response in DynamoDB**

**Step 1: Create a DynamoDB Table**

1. Go to AWS Console and ensure you're in us-east-1 (N. Virginia).
2. Navigate to **DynamoDB** from the **Services** menu.
3. Click **Create Table**.
4. Set the table name to **myBotResponse**.
5. Set the Partition Key as **Intent** with type **String**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Leave all other options as default and click **Create Table**.
2. Your table will be created within 2-3 minutes.

**Note: Ignore if there is any warning message.**

**Step 2: Insert Data into DynamoDB**

1. Click **Explore Items** on the left panel.
2. Select the table myBotResponse.
3. Click on **Create Item**.
4. Add attribute **Intent** with value **welcome**.
5. Add a new attribute named **Message** as a **List**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Insert **two strings** into the list:

A screenshot of a computer

AI-generated content may be incorrect.

* + Hi, what would you like to do?
  + Hello, How can I help you today?

A screenshot of a computer

AI-generated content may be incorrect.

1. Click **Create Item**.

**Step 3: Create a Lambda Function**

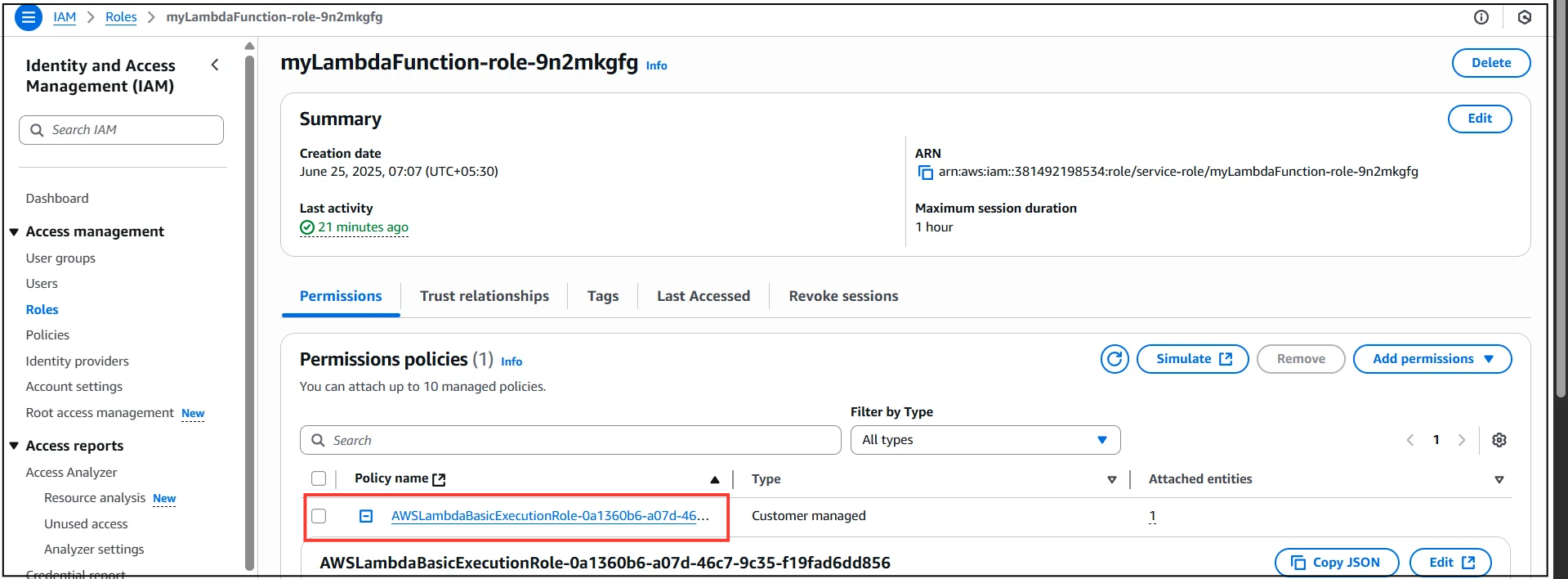
1. Go to **Services > Lambda**.
2. Click on **Create Function**.
3. Choose **Author from scratch**.
4. Set function name: **myLambdaFunction.**
5. Select **Node.js 18.x** as the runtime.
6. Permissions: Click on the **change default execution role** and choose Create a new role with basic Lambda permissions**.**
7. Click **Create Function**.
8. **Configure Lambda Execution Role for DynamoDB Access**

* Once your Lambda function is created:
* Navigate to the **Configuration** tab of your Lambda function (from the Lambda function page).
* Click on **Permissions** in the left sidebar.
* Under **Execution role**, click on the **role name** link to open the IAM role in a new tab.

A close-up of a white background

AI-generated content may be incorrect.

* On the IAM role page, scroll down to the **Permissions policies** section.
* Expand the existing policy (likely named something like AWSLambdaBasicExecutionRole-xxxx) by clicking the small arrow next to it.



* Click **Edit Policy**.

9. Modify Policy to Allow DynamoDB Access

* In the visual editor, switch to the **JSON** tab (top right corner of the policy editor).
* Replace the existing policy JSON with the one provided below (or add only the DynamoDB permissions block if you're extending an existing policy):

|  |
| --- |
| {  "Version": "2012-10-17",  "Statement": [  {  "Action": [  "logs:Describe\*",  "logs:Get\*",  "logs:List\*",  "logs:StartQuery",  "logs:StopQuery",  "logs:TestMetricFilter",  "logs:FilterLogEvents"  ],  "Effect": "Allow",  "Resource": "\*",  "Condition": {  "StringEquals": {  "aws:RequestedRegion": "us-east-1"  }  }  },  {  "Action": [  "dynamodb:Get\*",  "dynamodb:List\*",  "dynamodb:Describe\*",  "dynamodb:CreateTable",  "dynamodb:DeleteTable",  "dynamodb:Scan"  ],  "Effect": "Allow",  "Resource": "\*",  "Condition": {  "StringEquals": {  "aws:RequestedRegion": "us-east-1"  }  }  }  ]  } |

Click **Review policy** and then click **Save changes**.

10. Return to Lambda Console

* Go back to your Lambda function's page.
* Your function now has the necessary permissions to read items from the myBotResponse table in DynamoDB.

**Step 4: Configure Lambda Function Code**

1. In the Lambda function console, scroll down to the **Code** section.
2. Replace any existing code with the logic to read from DynamoDB and respond based on the Intent.

|  |
| --- |
| import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  import { DynamoDBDocumentClient, GetCommand } from "@aws-sdk/lib-dynamodb";  // Initialize DynamoDB client for us-east-1  const client = new DynamoDBClient({ region: "us-east-1" });  const docClient = DynamoDBDocumentClient.from(client);  const table = "myBotResponse";  export const handler = async (event) => {  console.log("Received event:", JSON.stringify(event, null, 2));  // Support both Lex V2 and V1 formats  const intent = event.sessionState?.intent?.name || event.currentIntent?.name || "unknown";  console.log("Extracted intent:", intent);  if (intent === "welcome") {  const param = {  TableName: table,  Key: { Intent: intent }  };  try {  const data = await docClient.send(new GetCommand(param));  console.log("DynamoDB GetCommand result:", JSON.stringify(data, null, 2));  let message = "No messages available.";  if (data.Item) {  const msg = data.Item.Message;  if (Array.isArray(msg) && msg.length > 0) {  const randomIndex = Math.floor(Math.random() \* msg.length);  message = msg[randomIndex];  } else if (typeof msg === "string") {  message = msg;  }  }  // Respond using Lex V2 format (compatible with both)  return {  sessionState: {  dialogAction: {  type: "Close"  },  intent: {  name: intent,  state: "Fulfilled"  }  },  messages: [  {  contentType: "PlainText",  content: message  }  ]  };  } catch (err) {  console.error("Error fetching message from DynamoDB:", err);  return {  sessionState: {  dialogAction: {  type: "Close"  },  intent: {  name: intent,  state: "Failed"  }  },  messages: [  {  contentType: "PlainText",  content: "An error occurred while fetching the response."  }  ]  };  }  }  // Handle unsupported or unknown intents  return {  sessionState: {  dialogAction: {  type: "Close"  },  intent: {  name: intent,  state: "Failed"  }  },  messages: [  {  contentType: "PlainText",  content: "Sorry, I couldn't process your request."  }  ]  };  }; |

**Step 5: Configure Test Event in Lambda**

1. Click the **Test** button in Lambda.
2. Click **Configure test event**.
3. Name the event **myTestFunction**.
4. Replace the default test event JSON with a sample input containing the intent welcome.

|  |
| --- |
| {  "bot": {  "alias": "null",  "name": "mySampleChatBot",  "version": "$LATEST"  },  "currentIntent": {  "confirmationStatus": "none",  "name": "welcome",  "slots": {}  }  } |

1. Click **Save**, then **Test**.

A screenshot of a computer

AI-generated content may be incorrect.

A white background with red dots

AI-generated content may be incorrect.

**Step 6: Create a Bot in Amazon Lex**

1. Go to **Services > Amazon Lex**.
2. Click **Get Started**, then click **Create Bot**.
3. Choose **Traditional** bot.
4. Select **Create a blank bot**.

A screenshot of a computer

AI-generated content may be incorrect.

1. Set Bot Name: **mySampleChatBot**.
2. Under **IAM Permissions**, choose an create a role with basic permissions.
3. Select **English (US)** as the language.
4. Set COPPA to **No**.
5. Click **Done**.

**Step 7: Create an Intent**

1. You will be redirected to the **Intent Creation** page.
2. Name the intent welcome.
3. Under **Sample Utterances**, add:
   * Hello
   * Hi
   * Hai
4. Scroll to **Code Hooks** and enable Lambda for **initialization and validation**.
5. Click **Save Intent**.

A screenshot of a computer

AI-generated content may be incorrect.

1. You will be able to see the new Intent **welcome** under the **Intents** section on the left side.

**Step 8: Build the Bot**

1. Click the **Build** button at the top-right corner.
2. Wait for the build to complete successfully.

**Step 9: Grant Lex Permission to Invoke Lambda**

1. Go back to the Lambda function.
2. Go to the **Configuration** tab → **Permissions** section.
3. Scroll down to **Resource-based policy statements** and click **Add permissions**.
4. Set:
   * Service: **Other**
   * Principal: lex.amazonaws.com
   * Action: lambda:InvokeFunction
   * Source ARN: Paste the alias ARN of your Lex bot
5. **To form the Lex bot alias ARN:**
6. Use this format:

|  |
| --- |
| arn:aws:lex:<region>:<account-id>:bot/<bot-id>/alias/<alias-id> |

1. Confirm and save the permission.

**Step 10: Test Your Bot**

1. In Lex, click on the **Test** button on the top-right.
2. Go to **Settings** and link the Lambda function under the bot source.
3. Enter sample utterances like "Hello" to test the response.
4. Ensure the response is fetched from DynamoDB.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a chat

AI-generated content may be incorrect.

Note : You can get Lambda codes in this GitHub Repo also https://github.com/DeepikaSidda/AWS\_Lex\_Chatbot

**Conclusion:**

In this project, we successfully created an AWS Lambda function that integrates with Amazon Lex V2 to process user inputs and interact with a DynamoDB table. By customizing IAM roles and permissions, we ensured secure access to necessary AWS resources. This setup demonstrates how serverless technologies and AI services can be combined to build intelligent, scalable, and cost-effective conversational applications. This foundational structure can be further extended to include advanced business logic, analytics, and multi-channel integrations.