Guided Lab: Natural Language Processing

Lab overview

Natural language processing (NLP) is a common use case for machine learning (ML).

In this lab, you will learn how to use Amazon Lex to create a chatbot that users can interact with to create a dental appointment.

The high-level steps you will perform are:

* Create and test a bot by using the Amazon Lex *ScheduleAppointment* blueprint
* Create and test an AWS Lambda function to perform initiation, validation, and fulfillment tasks
* Configure the *MakeAppointment* intent to use your Lambda function as a code hook
* Test the bot in the console and through a static webpage that is hosted on Amazon Simple Storage Service (Amazon S3)

Objectives

After completing this lab, you will be able to:

* Create and test a bot by using Amazon Lex
* Create a Lambda function and configure it to work with Amazon Lex
* Create a static webpage in Amazon S3 to host the bot
* Interact with the bot in the webpage

Prerequisites

This lab requires:

* Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS, or Linux (Ubuntu, SUSE, or Red Hat)
* For Microsoft Windows users: Administrator access to the computer
* An internet browser such as Chrome, Firefox, or IE9 (previous versions of Internet Explorer are not supported)

Duration

This lab requires **60** minutes to complete. The lab will remain active for **180** minutes.

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

A **Start Lab** panel opens, which displays the lab status.

1. Wait until you see the message *Lab status: ready*, then close the **Start Lab** panel by choosing the **X**.
2. At the top of these instructions, choose AWS

This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

**Tip**: If a new browser tab does not open, there will typically be a banner or icon at the top of your browser that indicates that your browser is preventing the website from opening pop-up windows. Select the banner or icon and then choose **Allow pop ups**.

1. Arrange the **AWS Management Console** browser tab so that it displays next to these instructions. Ideally, you should be able to see both browser tabs at the same time, which can make it easier to follow the lab steps.

Scenario summary

The following diagram illustrates the environment that you will build in this lab:

Task 1: Creating a bot with Amazon Lex

In this task, you will create a bot by using the AWS Management Console.

To create a bot with Amazon Lex:

1. On the AWS Management Console, on the **Services** menu, choose **Amazon Lex**.
2. On the **Amazon Lex** page, choose **Get Started**.
3. In the left navigation pane, choose **Return to the V1 console**.
4. Choose **Create**.
5. On the **Create your bot** page, choose the **ScheduleAppointment** blueprint.
6. For the **Language** section, select **English (US)**.
7. For the **COPPA** section, select **No**.
8. Choose **Create**.

Task 2: Testing your bot

Now, you will test your bot by using the test window in the console.

1. When you see the status *Ready Build complete*, test your bot by entering the following values:

* I would like to make an appointment
* A root canal
* 5/1/2020
* 4:00 PM
* Yes

You should see the following confirmation.

!

Task 3: Creating an AWS Lambda function

1. On the AWS Management Console, on the **Services** menu, choose **Lambda**.
2. Choose **Create a function**.
3. Choose the **Use a blueprint** tab.
4. In the search box, filter for Amazon Lex blueprints by entering Lex.
5. Open the **Make an appointment with Lex** blueprint by selecting it.
6. For the **Function name**, enter MakeAppointmentCodeHook.
7. For the **Execution role**, select **Create a new role from AWS policy templates**.
8. For the **Role name**, enter myLexrole.

Take a few minutes to review the Python code in the function.

1. Choose **Create function**.

When your function is ready, it should be in the **Lambda Designer** window.

!

Task 4: Testing your Lambda function

1. On the **Designer** page, choose **Test**.
2. On the **Test event** page, for the name of the event, enter MyMakeAppointmentEvent.
3. Choose **Save**.
4. Choose **Test**.

You see the message: *Execution result: succeeded*. You can examine the details of the test by expanding the **Details** section.

Task 5: Updating the intent of your bot

Each bot you create in Amazon Lex has an *intent*. The intent is an action that the bot will fulfill. In this task, you will update the intent to use the AWS Lambda function that you created.

1. On the AWS Management Console, on the **Services** menu, choose **Amazon Lex**. In the left navigation pane, choose **Return to the V1 console**.
2. From the **Bots** list, select the **ScheduleAppointment** bot that you created in Task 1.
3. Expand the **Lambda initialization and validation** section.
4. Select the **Initialization and validation code hook** option.
5. From the dropdown menu, select **MakeAppointmentCodeHook**.

You should get a message that you are going to give Amazon Lex permission to invoke your Lambda function.

1. Choose **OK**.

In addition to adding the code for initializing the bot, you must also add code to fulfill the request.

1. In the **Fulfillment section**, choose the **AWS Lambda function** radio button.
2. From the dropdown list of Lambda functions, select **MakeAppointmentCodeHook**.
3. Choose **Save Intent**.

Task 6: Building and testing your bot

Now, you will test your bot to make sure that it uses the Lambda function.

1. Select **Build**.

You are prompted with a message that says that you can continue to edit your bot while it builds.

1. Select **Build**.

After the build is complete, you should receive a confirmation that your bot build was successful.

1. Test your bot by entering the following details.

* Enter Make an appointment
* Select **root canal**.
* Select one of the times that are displayed. If you select any of the available times and do not get a prompt to confirm the appointment, continue to the next step.
* Confirm the appointment by selecting **yes**.

Now that you have a working version of your bot, you must publish it so that you can test it by calling it from a webpage.

1. Choose **Publish**.
2. For your bot's alias, enter a unique name.
3. Choose **Publish**.

After the bot finishes publishing, you should get a confirmation notification. Make a note of the alias because you will need it in the next task.

Incorporating your bot into a webpage

Now that your bot is working, it's time to test it by running it from a webpage. The easiest way to do the test is to create a static webpage and host it on Amazon S3. The webpage will invoke the Amazon Lex API to load your bot. Visitors to the webpage can then interact with it.

Task 7: Setting up an Amazon Cognito identity pool

You will now set up a webpage for testing your appointment bot. This webpage will be hosted in Amazon S3 as a static webpage. To add security to this page, you must first set up an Amazon Cognito identity pool.

To set up the identity pool:

1. On the AWS Management Console, on the **Services** menu, choose **Cognito**.
2. In the **Amazon Cognito** navigation pane at the left, choose **Identity pools**.
3. Choose **Create identity pool**. Choose **Guest access**. Choose **Next**.
4. For **IAM role name** use myidentitypoolrole. Choose **Next**. For **Name** use myidentitypool.
5. Choose **Next**. Choose **Create identity pool**. Expand the **View Details** section.
6. Select the myidentitypool.
7. Under **Identity pool overview**.
8. Make a note of the **IdentityPoolID** because you will need this ID later in the lab.

Task 8: Modifying IAM roles to allow access to Amazon Lex

The webpage that hosts your bot must be allowed to access Amazon Lex. To do this, the two roles that were created in the Amazon Cognito identity pool need permissions to access Amazon Lex. You must configure AWS Identity and Access Management (IAM) to grant these permissions to the identity pool roles.

1. On the AWS Management Console, on the **Services** menu, choose **IAM**.
2. Choose **Roles**.
3. Search for **myidentitypoolrole**.
4. From the list of search results, select the role that you created.
5. Choose **Add permissions**.
6. Choose **Attach policies**.
7. In the search box, enter AmazonLex.
8. Select the **AmazonLexReadOnly** and **AmazonLexRunBotsOnly** policies.
9. Choose **Add permissions**.

Task 9: Creating an S3 bucket

Now that you set up the security permissions, you must create an S3 bucket to host your webpage.

1. First, download the following two webpage files and extract them to a local file directory.

* index.html
* error.html

The index.html file includes the script that will load your bot.

[Download the .zip file.](https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/CUR-TF-200-ACMLFO-1/lab-6/202011/en_us/lab6.zip)

1. On the AWS Management Console, on the **Services** menu, choose **S3**.
2. On the **Amazon S3** page, choose **Create bucket**.
3. Enter a name for your bucket. Because all S3 bucket names must be unique, try entering lexlab6 and six random letters.
4. Select **Create bucket**.
5. On the **Buckets** page, select the bucket that you just created.
6. On the **Bucket overview** page, select **Upload**.
7. On the **Upload** page, select **Add files**.
8. Browse to the location for the *index.html* and *error.html* files that you downloaded previously.
9. Select both files, and then select **Upload**.
10. After the upload is complete, return to the **Bucket overview** page by selecting **Close**.
11. On the **Properties** tab, scroll down to the **Static website hosting** section and select **Edit**.
12. Select **Enable**.
13. For the **Index document**, enter: index.html
14. For the error page, enter: error.html
15. Choose **Save changes**.

Task 10: Updating and testing the demonstration file

You must now update the demonstration HTML file so that it uses the Amazon Cognito identity pool that you created in Task 6.

Use a text editor to make the following changes to the HTML page.

1. On line 144 of the script, add the *IdentitypoolID* for the identity pool that you created in Task 6.
2. On lines 185 and 186 of the script, add the *botAlias* and *botName* for your bot.
3. Save your updated HTML page locally.
4. Choose the **Objects** tab for the bucket that you created in Task 9.
5. Choose **Upload** and upload the HTML file that you edited.
6. Choose **Close** to close the **Upload** page.
7. Choose the **Permissions** tab.
8. In the **Block public access** section, choose **Edit**.
9. Clear the following options:

* **Block *all* public access**

1. Choose **Save changes**.
2. In the confirmation dialog box, enter Confirm and then choose **Confirm**.
3. In the **Bucket policy** section, choose **Edit**.
4. To grant public read access to the webpage, copy the following bucket policy. In the **Bucket policy editor**, paste the policy.

{

  "Version": "2012-10-17",

  "Statement": [

      {

          "Sid": "PublicReadGetObject",

          "Effect": "Allow",

          "Principal": "\*",

          "Action": [

              "s3:GetObject"

          ],

          "Resource": [

              "arn:aws:s3:::example.com/\*"

          ]

      }

  ]

}

1. In the **Resource** part of the policy, edit the value of the Amazon Resource Name (ARN) by replacing *example.com* with the name of your bucket.
2. Choose **Save changes**. You will get a warning that the bucket has public access.
3. Choose the **Properties** tab.
4. Scroll down to the **Static website hosting** section.
5. At the bottom of the **Static website hosting** section, choose the URL.

Your webpage opens, and you can interact with your bot like you did in the AWS Management Console.