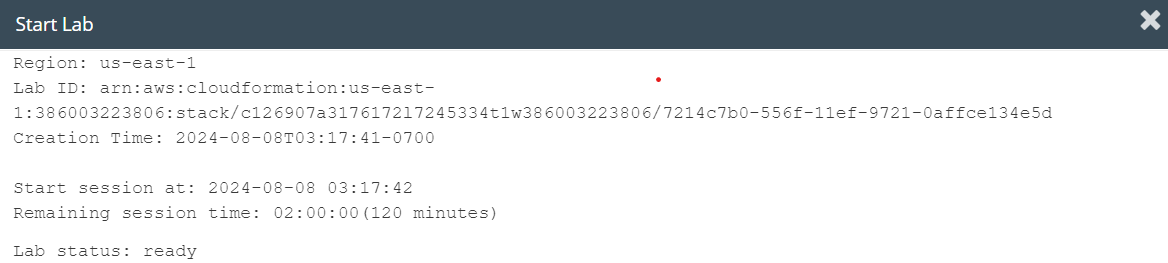
**LAB – 9**

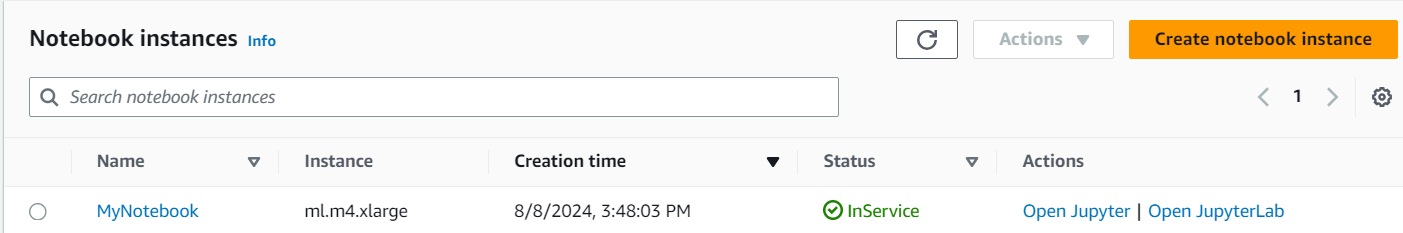
Objective: Guided Lab: Facial Recognition

Procedure:

1. Accessing the AWS Management Console

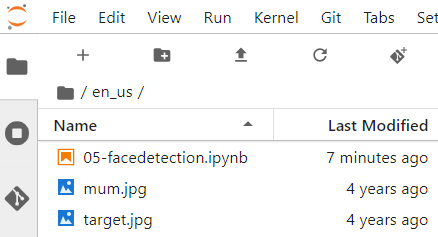


**Task 1: Accessing a notebook instance in Amazon SageMaker**

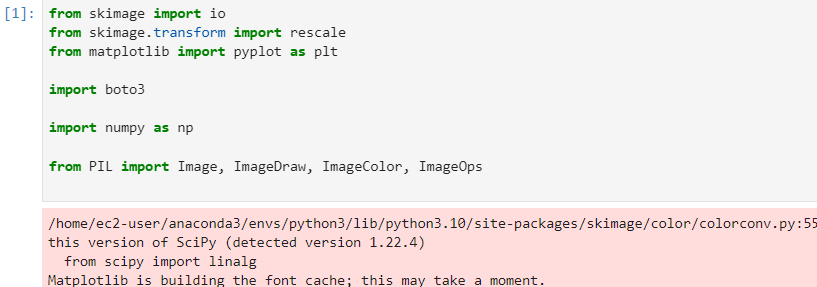


**Task 2: Opening a notebook in your notebook instance**

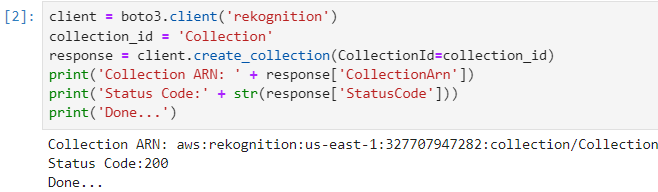
1. In your JupyterLab environment, go to the file browser in the left pane and locate the *en\_us/05-facedetection.ipynb* file & open it.



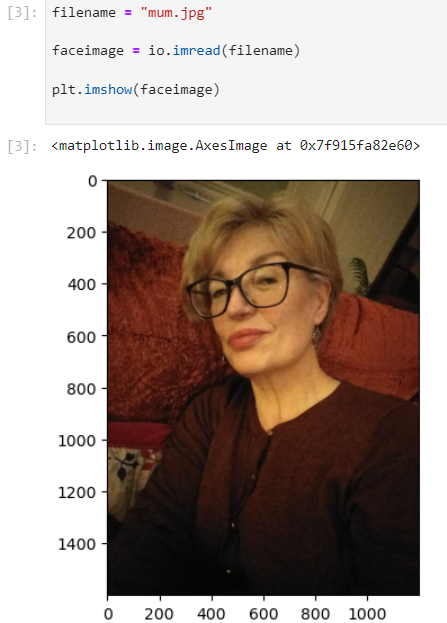
1. Follow the instructions in the notebook
   1. Importing Python packages



* 1. Creating a collection



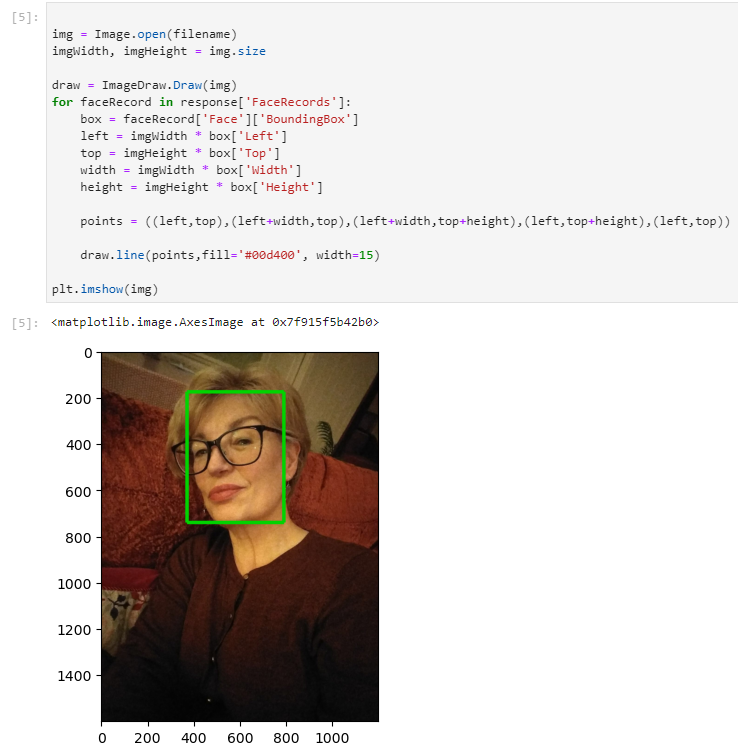
* 1. Uploading an image to search



* 1. Adding the image to the collection



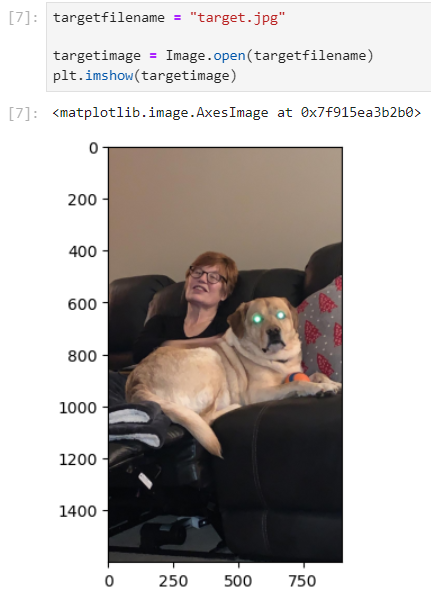
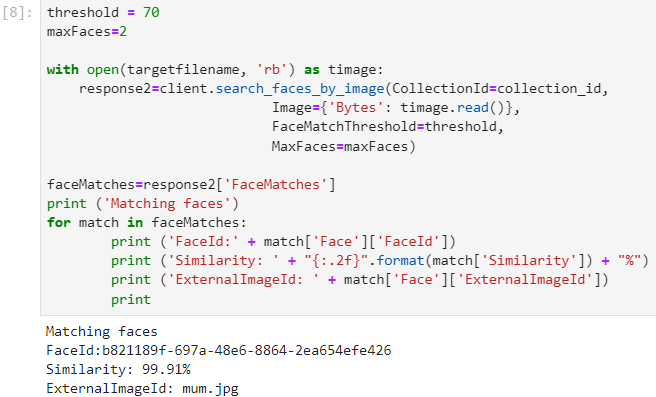
* 1. Viewing the bounding box for the detected face



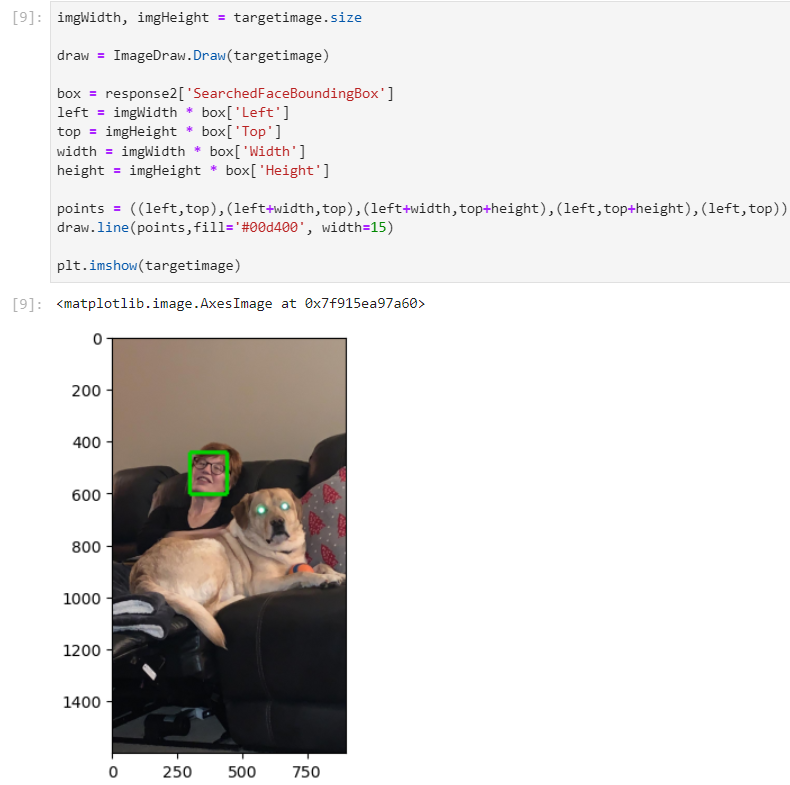
* 1. Listing faces in the collection



* 1. Finding a face by using the collection

* 1. Drawing a bounding box around the discovered face



* 1. Deleting the collection

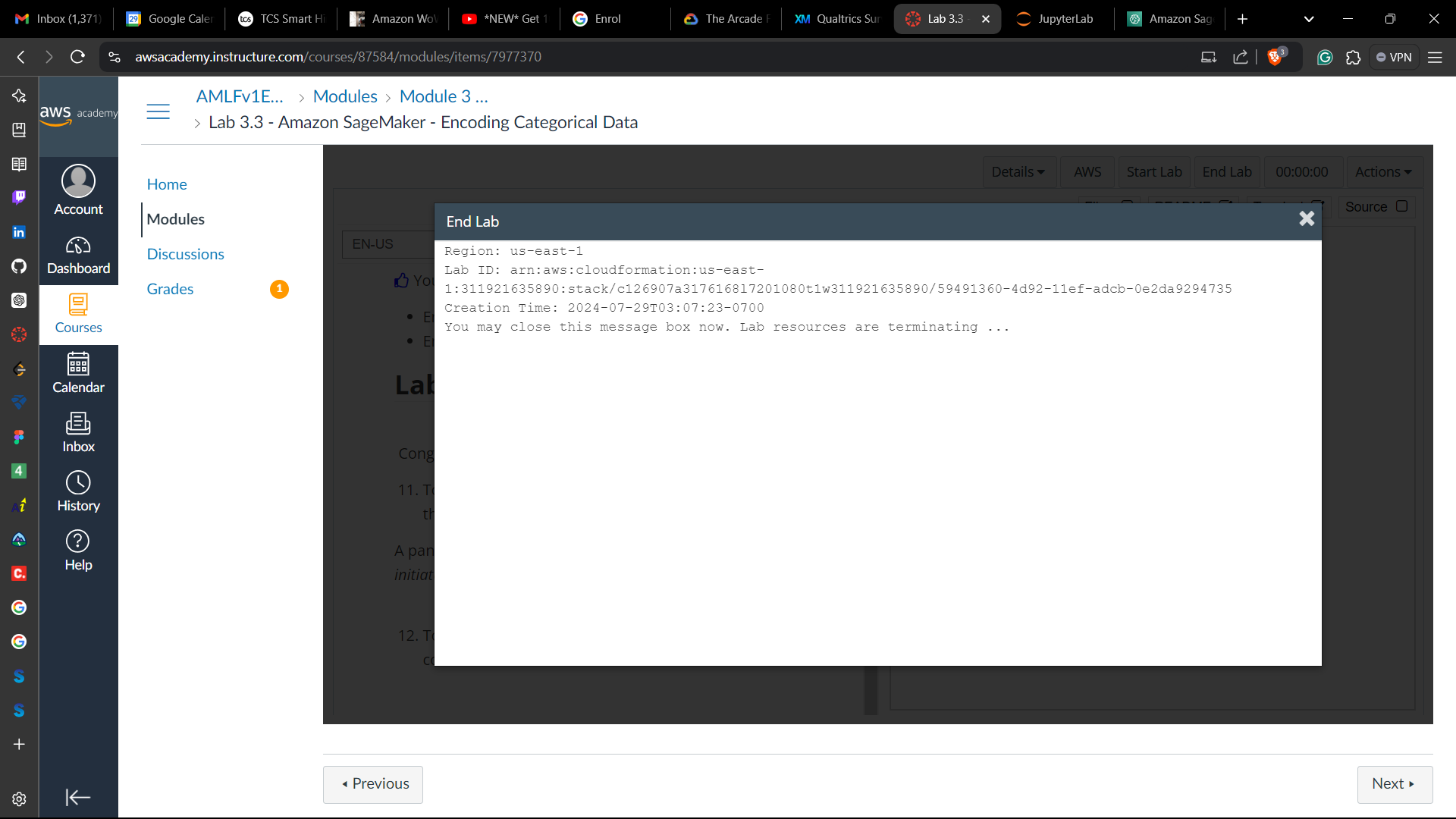


**Conclusion**

* Created a custom collection for Amazon Rekognition
* Added an image to a custom collection
* Detected known faces in an image

**Lab complete**

1. In order to end the lab, choose End Lab, and then choose **Yes**.

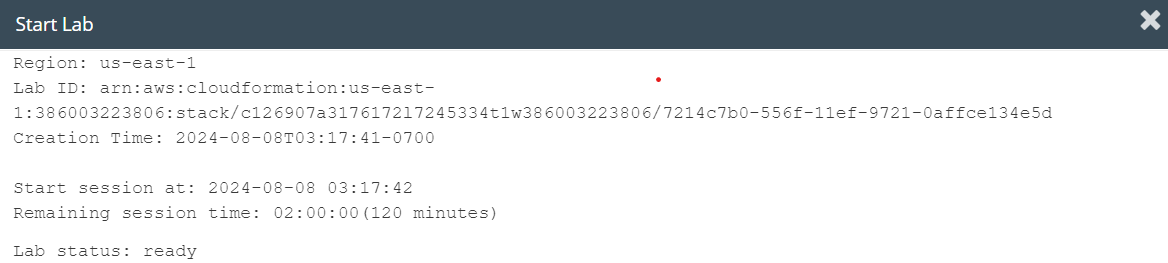


**LAB – 10**

Objective: Amazon Lex - Create a chatbot

Procedure:

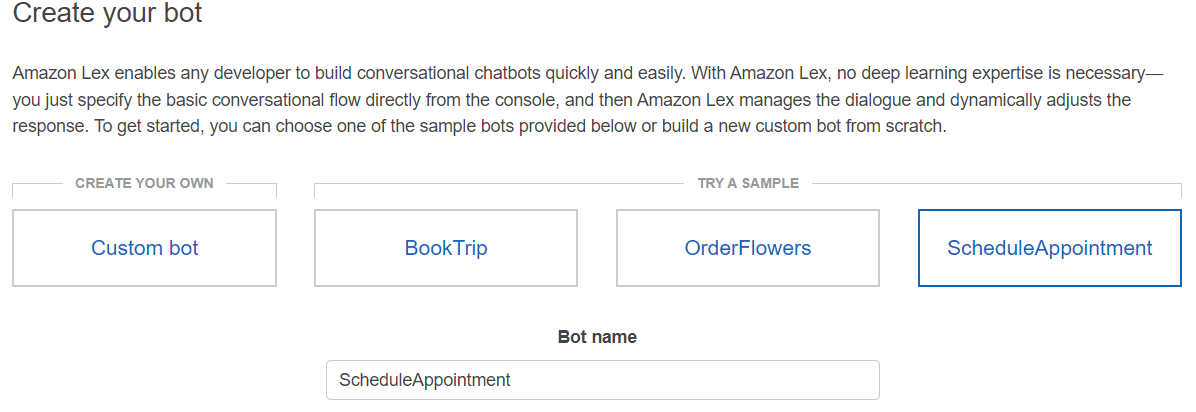
1. Accessing the AWS Management Console

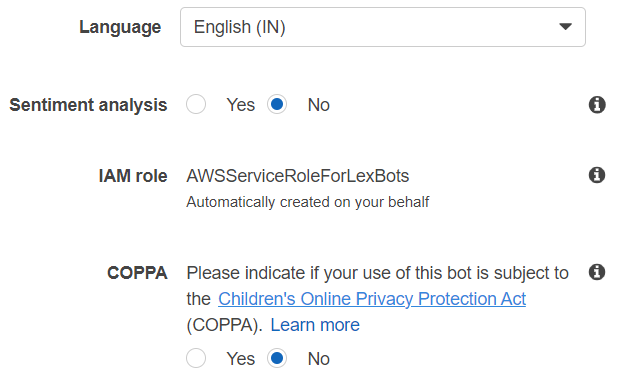


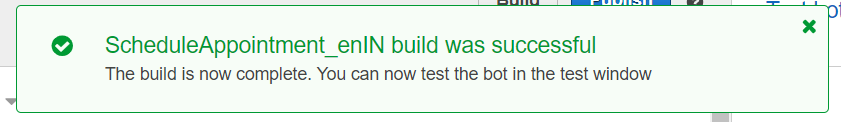
1. Following the instructions of the guided lab

**Task 1: Creating a bot with Amazon Lex**

* 1. On the AWS Management Console, under **Services** menu, choosing **Amazon Lex** & creating our own bot using the ***ScheduleAppointment*** blueprint.

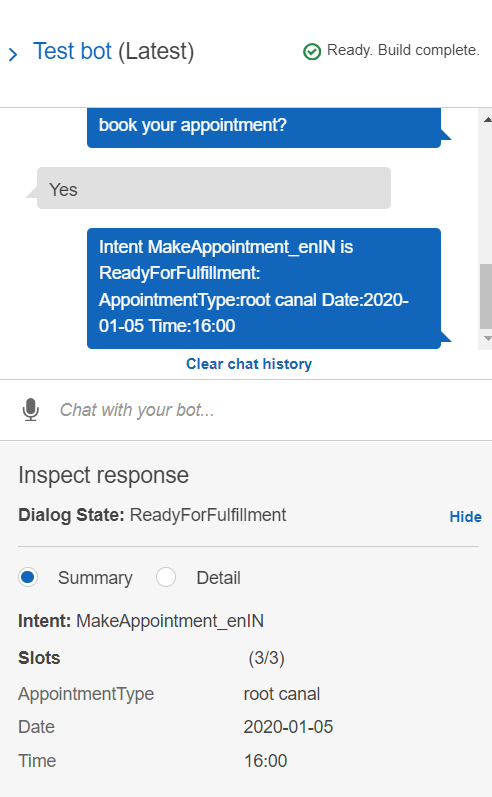






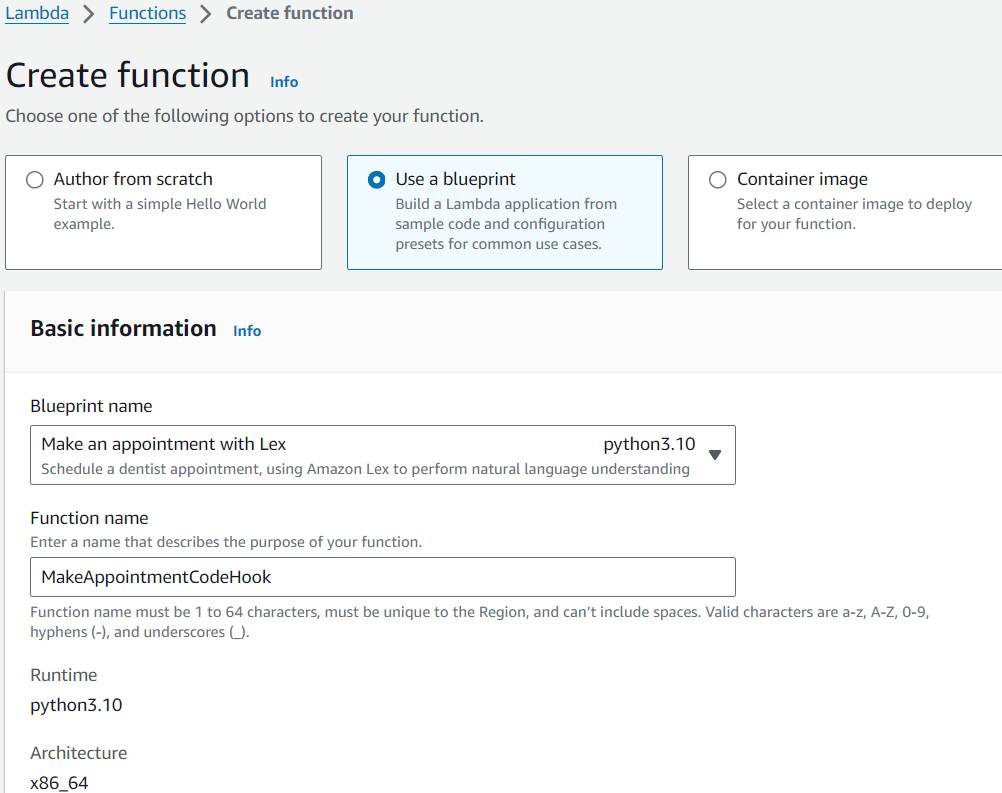
**Task 2: Testing your bot**

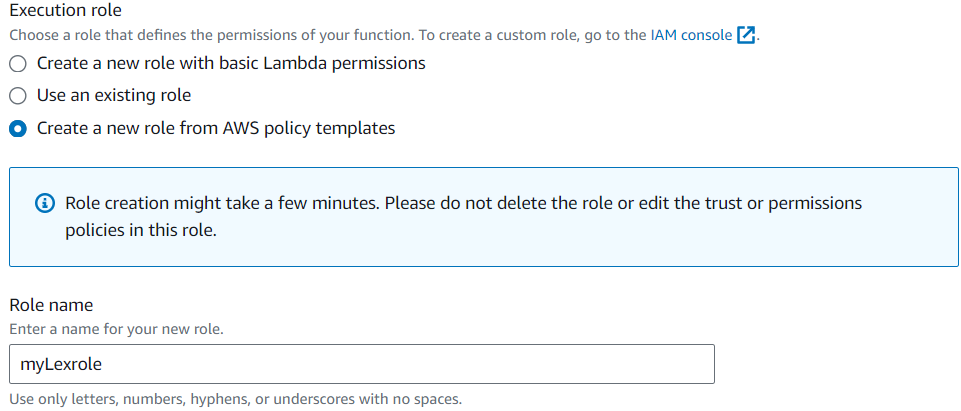
* 1. Testing the bot using custom values
  2. Once confirmed, the test bot would look like:



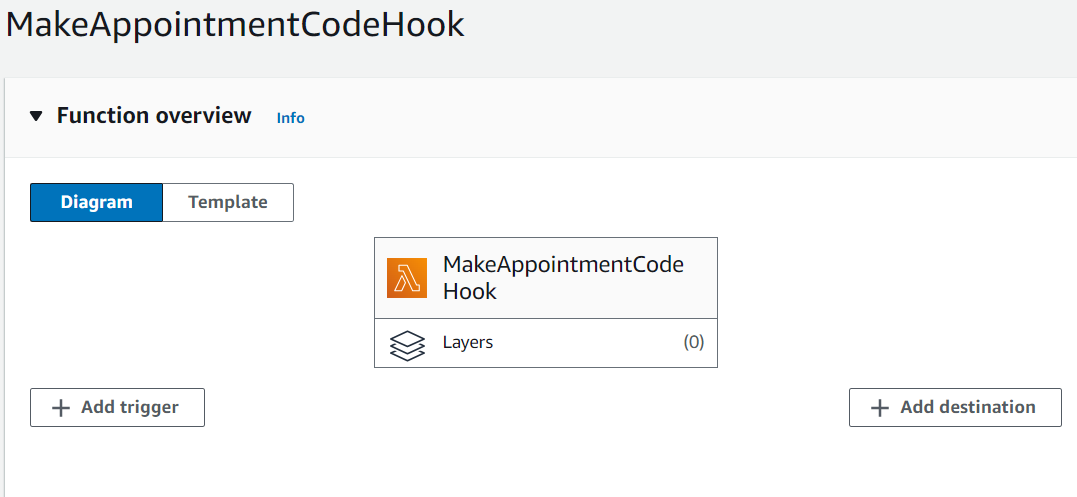
**Task 3: Creating an AWS Lambda function**

* 1. On the AWS Console, under **Services** menu choosing **Lambda &** Creating a function using the blueprint tab.



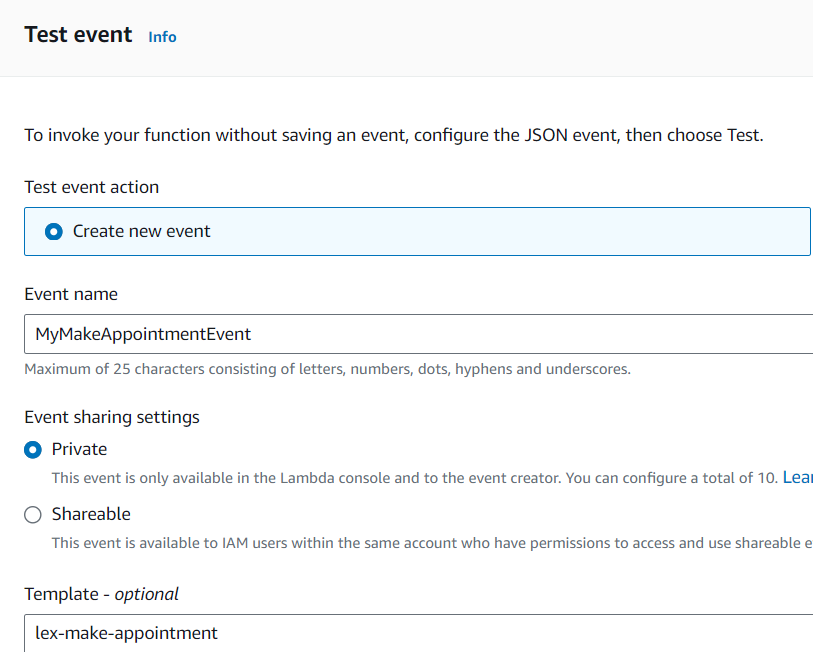


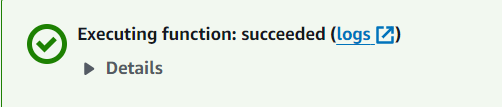
* 1. When the function is ready, it should be in the **Lambda Designer** window.



**Task 4: Testing your Lambda function**

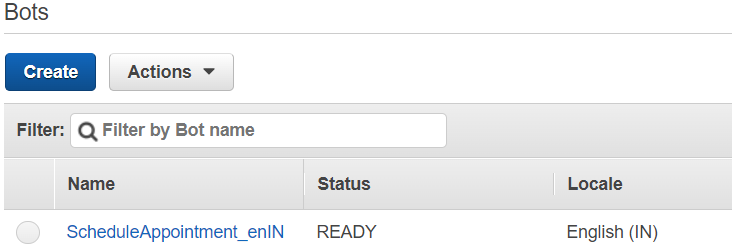
On the **Designer** page, for the name of the event, entering *MyMakeAppointmentEvent* & choosing to test

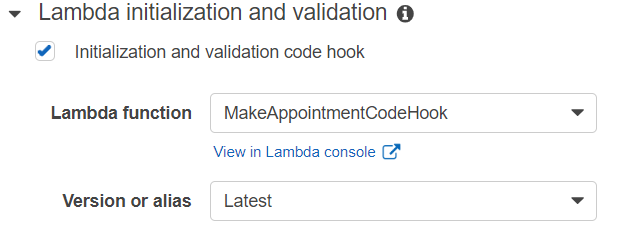




**Task 5: Updating the intent of your bot**

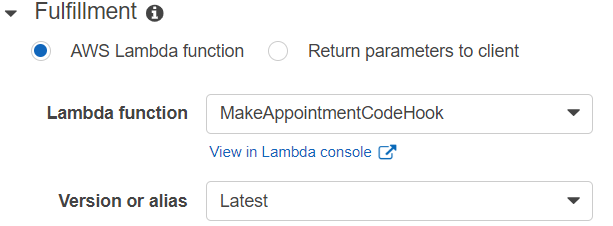
* 1. From the Bots list, initially select the *ScheduleAppointment* bot & selecting the **Initialization and validation code hook** option.



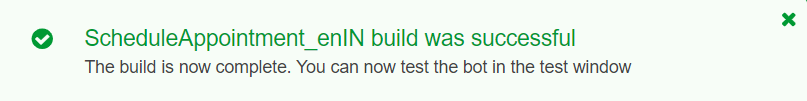


Hence, we have given Amazon Lex permission to invoke our Lambda function.

* 1. Now, adding code to fulfill the request by choosing the **AWS Lambda function** radio button & selecting *MakeAppointmentCodeHook* and Saving Intent.



**Task 6: Building and testing your bot**



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

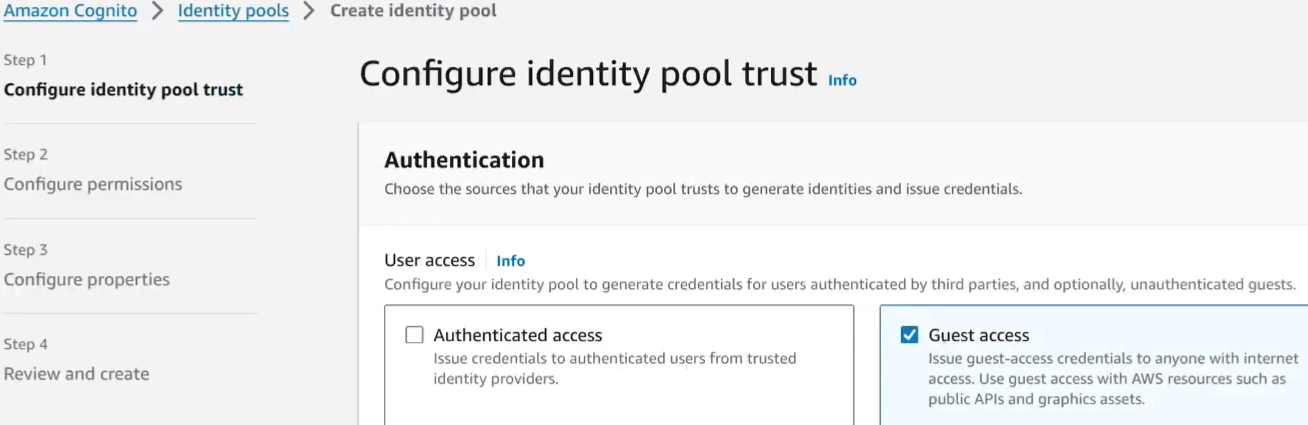
Now that we have a working version of our bot, one must ***publish*** it so that one can test it by calling it from a webpage.

**INCORPORATING OUR BOT INTO A WEBPAGE**

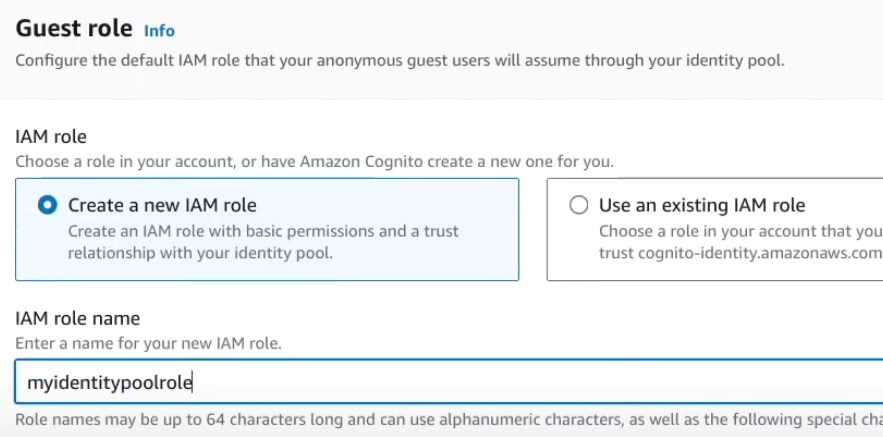
**Task 7: Setting up an Amazon Cognito identity pool**

Setting up a webpage for testing your appointment bot. This webpage will be hosted in Amazon S3 as a static webpage.

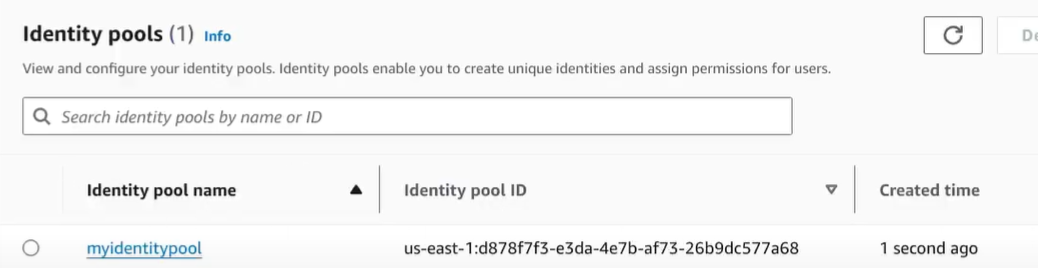
* 1. Setting up the identity pool, by navigating to **Amazon Cognito** pane, choosing *Identity pools.*



* 1. For **IAM role name** using *myidentitypoolrole*. Choose Next. For Name use *myidentitypool*.

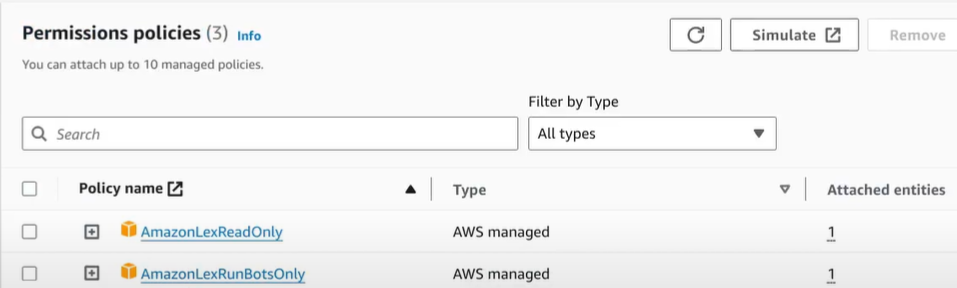


* 1. Choose Next. Choose Create identity pool.



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

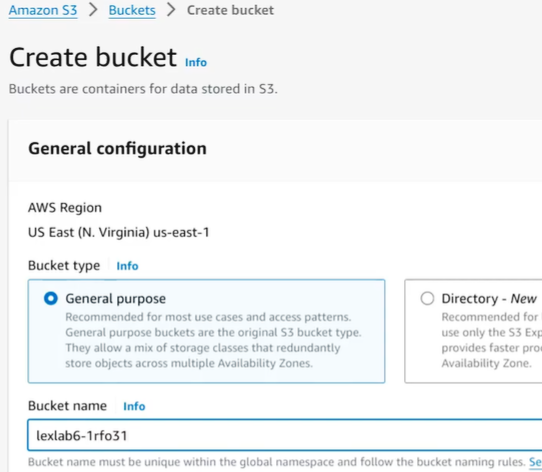
Configuring AWS Identity and Access Management (IAM) to grant permissions to the identity pool roles.



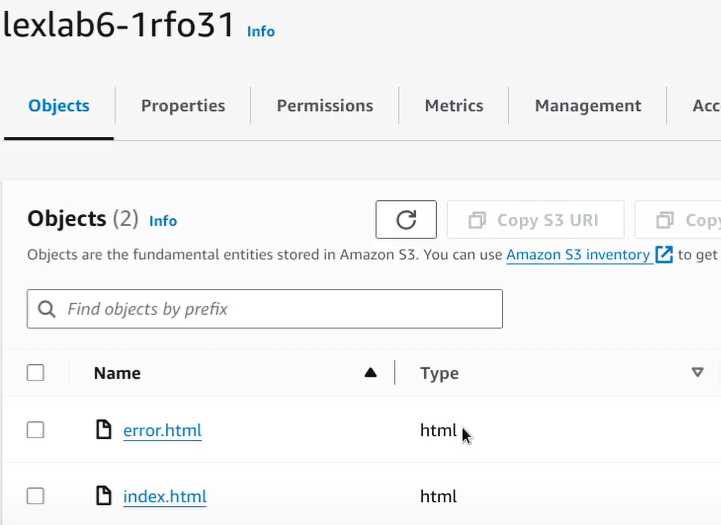
**Task 9: Creating an S3 bucket**

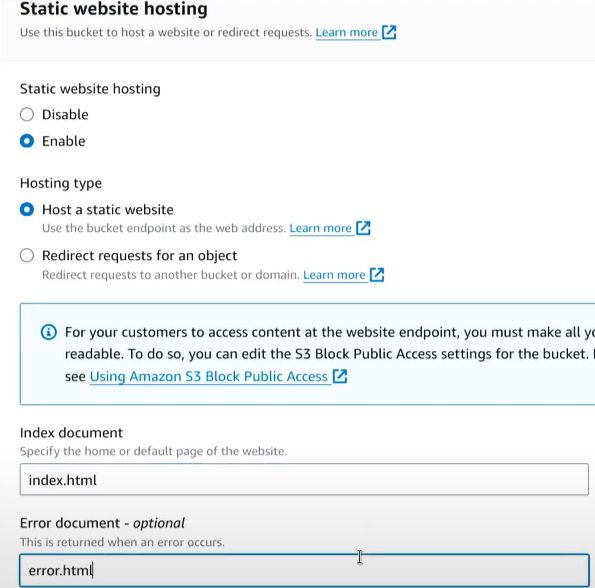
Now that the security permissions are set up, creating an S3 bucket to host our webpage.

* 1. First, downloading the 2 webpage files and extract them to a local file directory [index.html, error.html]
  2. On the **Amazon S3** page, choosing to **Create bucket**.



* 1. And, uploading the previously downloaded files
  2. After the upload is complete, on the **Properties** tab, scroll down to the **Static website hosting** section and select Edit > Enable > Save changes

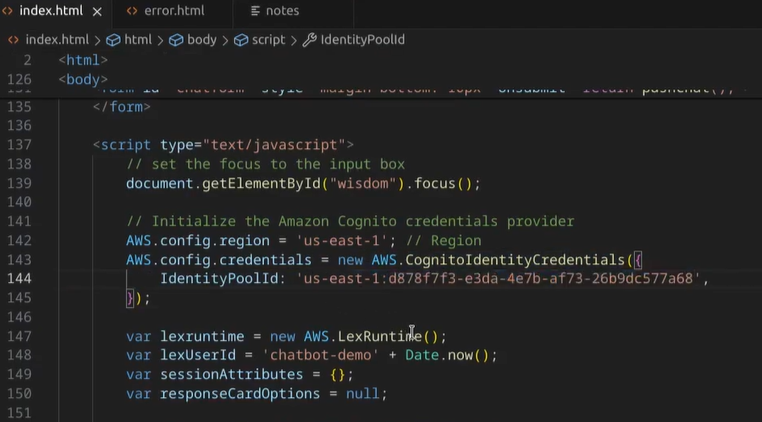




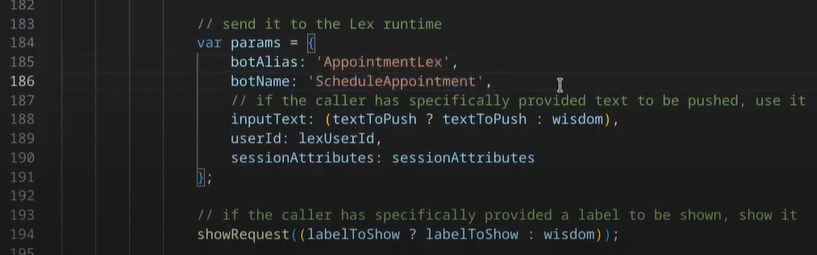
**Task 10: Updating and testing the demonstration file**

10.1 Use a text editor to make the following changes to the HTML page:

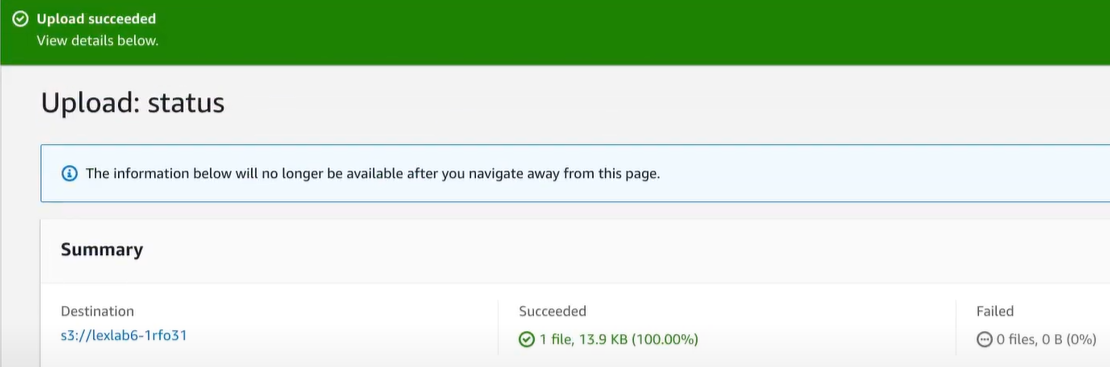
1. On line 144 of the script, add the *IdentitypoolID* for the identity pool that you created in Task 6



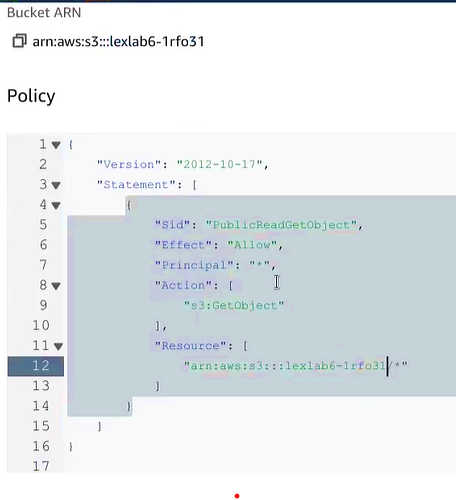
1. On lines 185 and 186 of the script, add the *botAlias* and *botName* for your bot



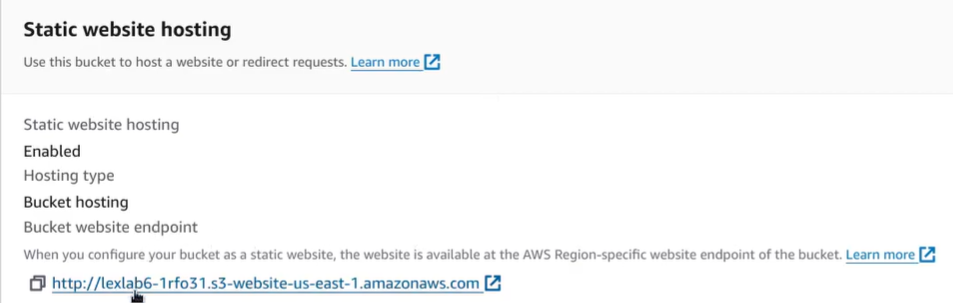
& save your updated HTML page locally.



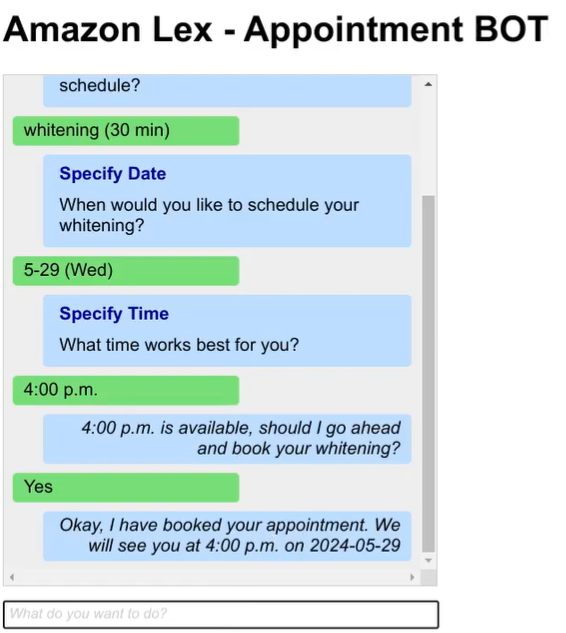
* 1. In the **Bucket policy** section, to grant public read access to the webpage, copy the following bucket policy. In the **Bucket policy editor**, paste the policy.
  2. In the **Resource** part of the policy, edit the value of the Amazon Resource Name (ARN) by replacing *example.com* with the name of our bucket.



* 1. At the bottom of the **Static website hosting** section, choose the URL



Once the webpage opens, and we can interact with our bot like we did in the Console.



**Conclusion –** Created a bot with Amazon Lex and tested it in a webpage.

**Lab complete**

In order to end the lab, choose End Lab, and then choose **Yes**

