K L UNIVERSITY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

20CS3281PA CSC Project Report

On

BUILDING A PICTIONARY-STYLE GAME WITH AWS DEEPLENS AND AWS ALEXA

SUBMITTED BY:

ROLL NUMBER	NAME
2000031181	PRAVALLIKA KANDI

UNDER THE ESTEEMED GUIDANCE OF

Dr. V. NARESH

KL UNIVERSITY
Green fields, Vaddeswaram – 522 502 Guntur Dt., AP, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to ensure that the project-based laboratory report entitle " BUILDING A PICTIONARY-STYLE GAME WITH AWS DEEPLENS AND AWS ALEXA " submitted by Ms. Pravallika Kandi having Regd.No. 2000031181 to the **Department of Computer Science & Engineering (HONS), KL Deemed to be University** as one of the prerequisites for the successful delivery of a project-based Laboratory in "Cloud & Serverless Computing" course in III B Tech II Semester, is a True record of his/her work done under my guidance during the educational year 2023.

PROJECT SUPERVISOR	**HEAD OF THE DEPT**
Dr. V. Naresh	Dr. A. Senthil

ACKNOWLEDGEMENTS

It gives us immense pleasure to thank our honourable President, Sri. Koneru Satyanarayana, for giving the ability and platform with facilities to complete the project-based laboratory report.

I would like to convey my profound thanks to our Director, Dr. A. Jagdeesh, for his management in fostering our academic growth.

I would like to express my heartfelt gratitude to our Coordinator and HOD-CSE Dr. A. Senthil for his support and continuous motivation in order to ensure successful completion of our academic semester. I take this opportunity to express my heartfelt gratitude for providing us with the effective faculty and facilities to turn our ideas into reality.

I would like to give special thanks to our project supervisor, Dr. V. Naresh, for her novel fusion of ideas, encouraging words, admiration, and intellectual zeal, which motivated us to finish this project successfully.

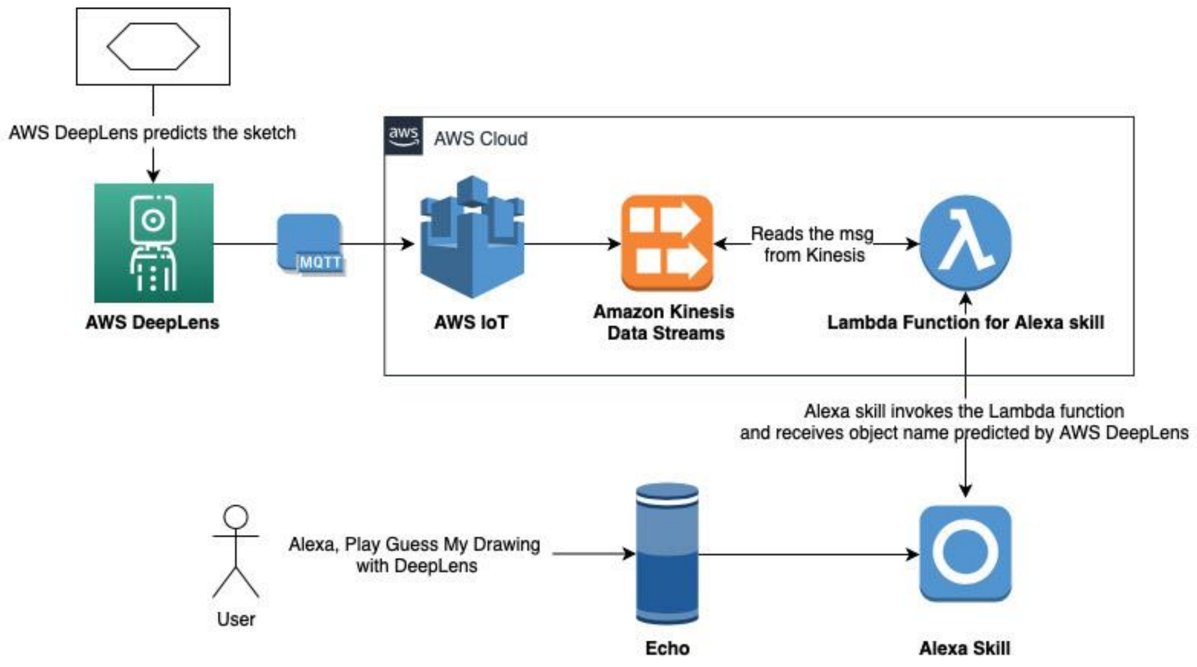
Finally, it is happy to express gratitude to everyone who contributed, directly or indirectly, to the success of this project report.

ROLL NUMBER	NAME
2000031181	PRAVALLIKA KANDI

1.SERVICES USED

- AWS DeepLens
- AWS IoT
- Amazon Kinesis
- AWS Lambda
- Alexa

2. ARCHITECTURE DIAGRAM


AWS DeepLens predicts the sketch
AWS DeepLens
MQTT
AWS Cloud
AWS IoT
Amazon Kinesis Data Streams
Reads the msg from Kinesis
Lambda Function for Alexa skill
Alexa skill invokes the Lambda function and receives object name predicted by AWS DeepLens
User
Alexa, Play Guess My Drawing with DeepLens
Echo
Alexa Skill

3.ABSTRACT

Tired with the same old board games? Tired of repeating yourself with charades week after week? Looking for a new and interesting way to spice up game night? We've got a solution for you!

Guess My Drawing with DeepLens is a do-it-yourself recipe for creating your own Machine Learning (ML)-enabled Pictionary-style game from the creators of AWS DeepLens! In this post, you'll discover how to use AWS DeepLens, the AWS programmable video camera for developers to learn ML, and Amazon Alexa, the Amazon cloud-based speech service, to your advantage.

You begin by learning how to deploy a trained model on AWS DeepLens that can recognise whiteboard sketches and pair it with an Alexa skill that acts as the official scorekeeper.

4.INTRODUCTION

Recognise My Drawing Using AWS To host a multiplayer sketching challenge game, DeepLens leverages Alexa. Gather the game equipment listed in the Prerequisites section before beginning.

Just say, "Alexa, play Guess My Drawing with DeepLens," to begin playing. The game's rules are explained by Alexa, who also questions about the number of players. The order of turns is decided by the players.

Each player receives a common word from Alexa. By way of illustration, Alexa might remark, "Your object to draw is bowtie." It must be drawn on a whiteboard in 12 seconds without the use of any letters or words.

When the timer runs out, the player finishes drawing and requests that Alexa reveal the results. The item that you sketched is predicted by the ML model running on AWS DeepLens. Alexa awards 10 points if the object matches the question. No points are awarded if DeepLens does not properly guess the drawing or if the player takes more than 12 seconds to draw.

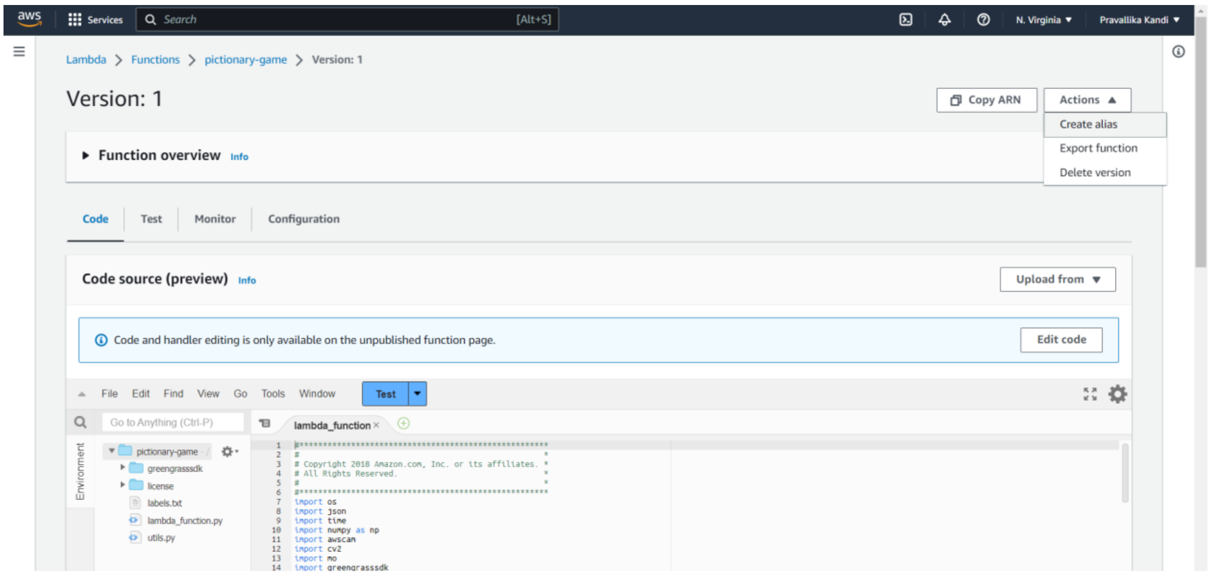
Alexa calls out the next participant's word, and so on until all participants have had their chance. Alexa updates the score after each round. The game is over after five rounds, and the one with the highest score wins!

5.PROPOSED WORK

1. Creating an AWS DeepLens inference Lambda function

To create your function, complete the following steps:

1. Download aws-deeplens-pictionary-lambda.zip.
2. On the Lambda console, choose **Create function**.
3. Choose **Author from scratch** and choose the following options:
    2. For **Runtime**, choose **Python 2.7**.
    3. For **Choose or create an execution role**, choose **Use an existing role**.
    4. For **Existing role**, enter service-role/AWSDeepLensLambdaRole.
4. After you create the function, go to the **Function code.**
5. From the **Actions** drop-down menu in **Function code**, choose **Upload a .zip file**.
6. Upload the aws-deeplens-pictionary-lambda.zip file you downloaded earlier.
7. Choose **Save**.
8. From the **Actions** drop-down menu, choose **Publish new version**.
9. Enter a version number and choose **Publish**.



Code and handler editing is only available on the unpublished function page.

```
#*****************************************************
#                                                    *
# Copyright 2018 Amazon.com, Inc. or its affiliates. *
# All Rights Reserved.                               *
#                                                    *
#*****************************************************
import os
import json
import time
import numpy as np
import awscam
import cv2
import mo
import greengrasssdk
```

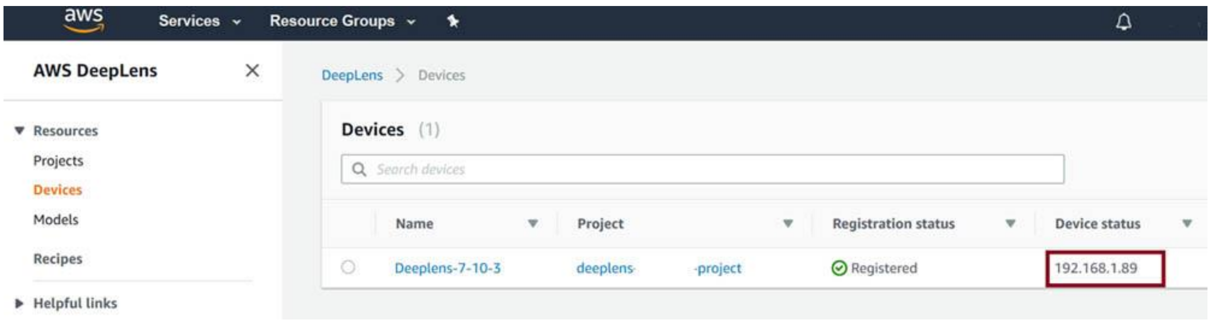
2. <u>Deploying the model to AWS DeepLens</u>

In this section, you set up your AWS DeepLens device, import a pre-trained model, and deploy the model to AWS DeepLens.

Setting up your AWS DeepLens device

You first need to <u>register your AWS DeepLens device</u>, if you haven't already.

After you register your device, you need to install the latest OpenCV (version 4.x) packages and Pillow libraries to enable the pre-processing algorithm in the DeepLens inference Lambda function.



	Name	Project	Registration status	Device status
○	Deeplens-7-10-3	deeplens- -project	Registered	192.168.1.89

Open a terminal application on your computer. SSH into your DeepLens by entering the following code into your terminal application:

```
ssh aws_cam@<YOUR_DEEPLENS_IP>
```

Then enter the following commands in the SSH terminal:

```
sudo su
pip install --upgrade pip
pip install opencv-python
pip install pillow
```

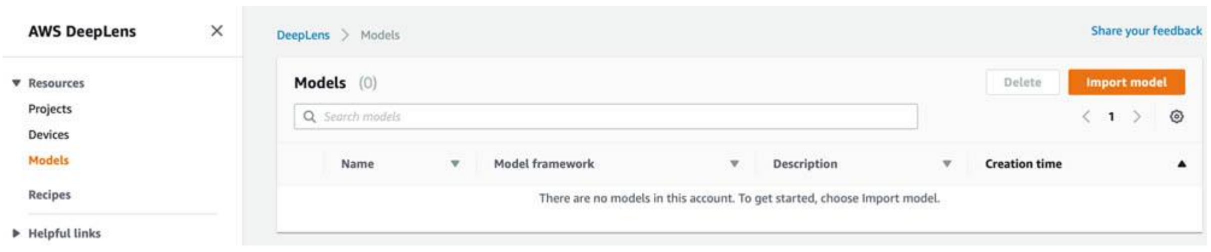
Importing the model to AWS DeepLens

To import your model, complete the following steps:

1. Download the model [aws-deeplens-pictionary-game.tar.gz]().
2. Create an Amazon Simple Storage Service (Amazon S3) bucket to store this model.

The S3 bucket name must contain the term deeplens. The AWS DeepLens default role has permission only to access the bucket with the name containing deeplens.

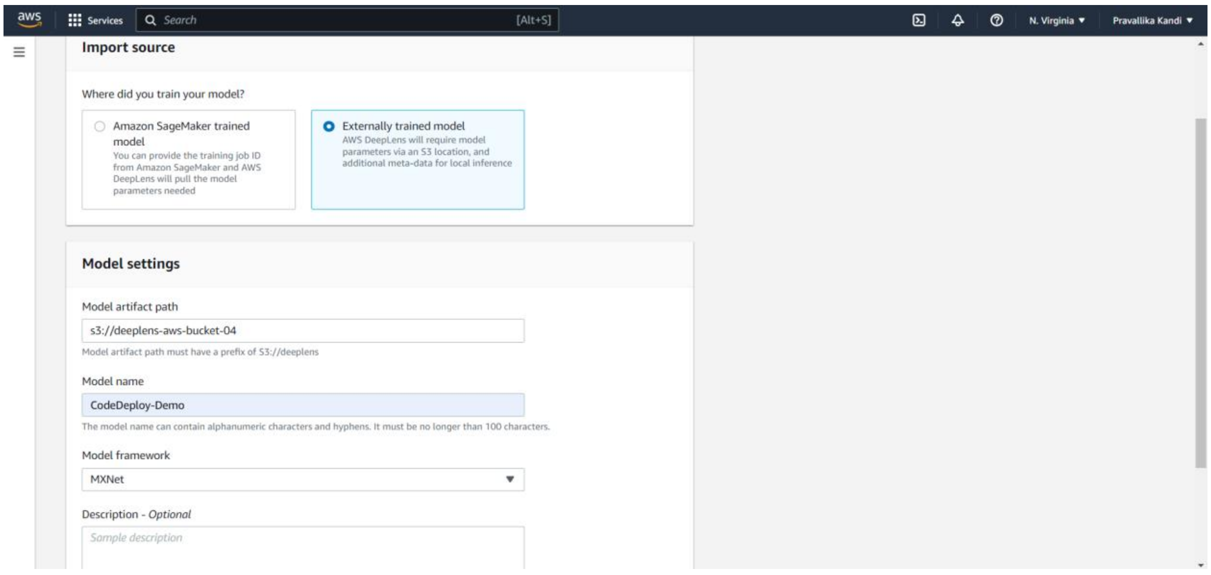
3. After the bucket is created, upload aws-deeplens-pictionary-game.tar.gz to the bucket and copy the model artifact path.
4. On the AWS DeepLens console, under **Resources**, choose **Models**.
5. Choose **Import model**.



There are no models in this account. To get started, choose Import model.

6. On the **Import model to AWS DeepLens** page, choose **Externally trained model**.

7. For **Model artifact path**, enter the Amazon S3 location for the model you uploaded earlier.
8. For **Model name**, enter a name.
9. For **Model framework**, choose **MXNet**.
10. Choose **Import model**.



Import source

Where did you train your model?

Amazon SageMaker trained model
You can provide the training job ID from Amazon SageMaker and AWS DeepLens will pull the model parameters needed

[✓ Externally trained model]
AWS DeepLens will require model parameters via an S3 location, and additional meta-data for local inference

Model settings

Model artifact path
s3://deeplens-aws-bucket-04
Model artifact path must have a prefix of S3://deeplens

Model name
CodeDeploy-Demo
The model name can contain alphanumeric characters and hyphens. It must be no longer than 100 characters.

Model framework
MXNet

Description - *Optional*
Sample description

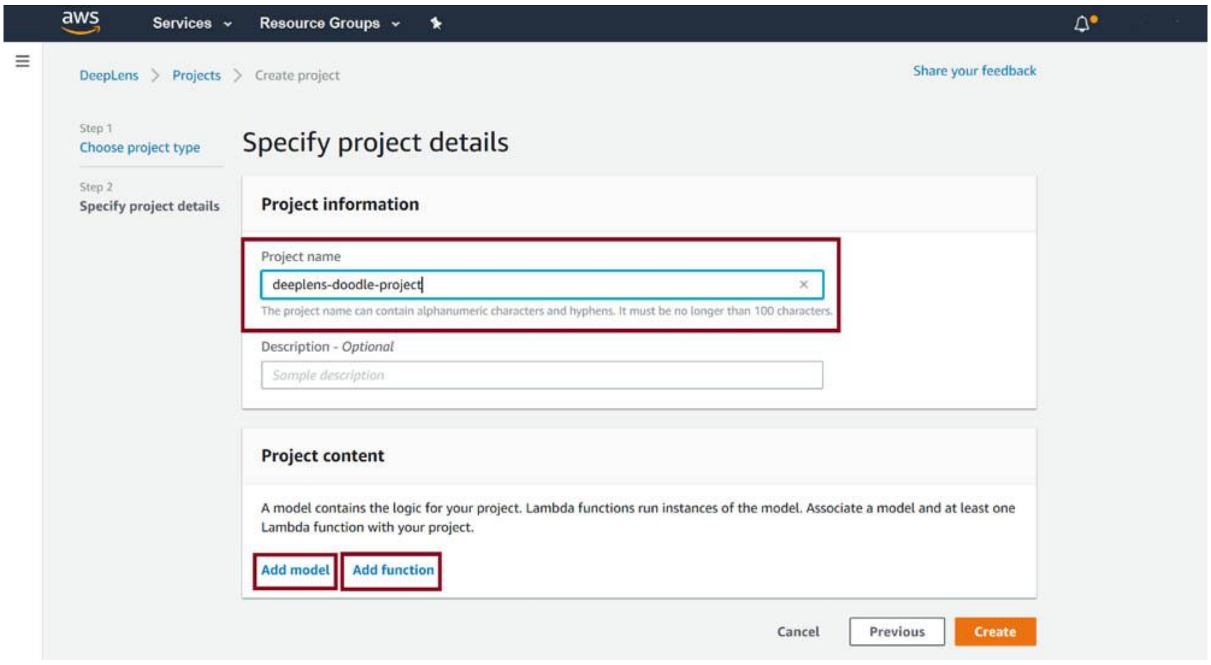
Deploying the model to your AWS DeepLens device

To deploy your model, complete the following steps:

1. On the AWS DeepLens console, under **Resources**, choose **Projects**.
2. Choose **Create new project**.
3. Choose **Create a new blank project**.
4. For **Project name**, enter a name.
5. Choose **Add model** and choose the model you imported earlier.

6. Choose **Add function** and choose the Lambda function you created earlier.

7. Choose **Create**.



Project name: deeplens-doodle-project

The project name can contain alphanumeric characters and hyphens. It must be no longer than 100 characters.

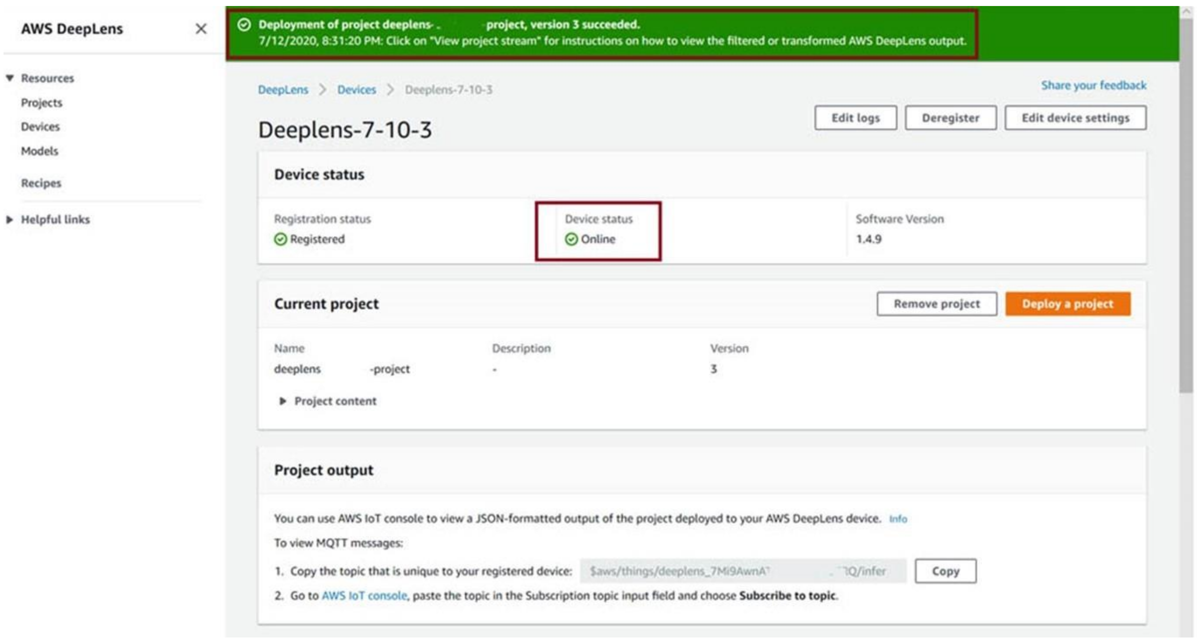
A model contains the logic for your project. Lambda functions run instances of the model. Associate a model and at least one Lambda function with your project.

8. Select your newly created project and choose **Deploy to device**.

9. On the **Target device** page, select your device from the list.

10. On the **Review and deploy** page, choose **Deploy**.

The deployment can take up to 5 minutes to complete, depending on the speed of the network your AWS DeepLens is connected to. When the deployment is complete, you should see a green banner message that the deployment succeeded.



Deployment of project deeplens- project, version 3 succeeded.
7/12/2020, 8:31:20 PM: Click on "View project stream" for instructions on how to view the filtered or transformed AWS DeepLens output.

Deeplens-7-10-3

Device status

Registration status	Device status	Software Version
Registered	Online	1.4.9

Current project

Name	Description	Version
deeplens -project	-	3

Project content

Project output

You can use AWS IoT console to view a JSON-formatted output of the project deployed to your AWS DeepLens device. Info

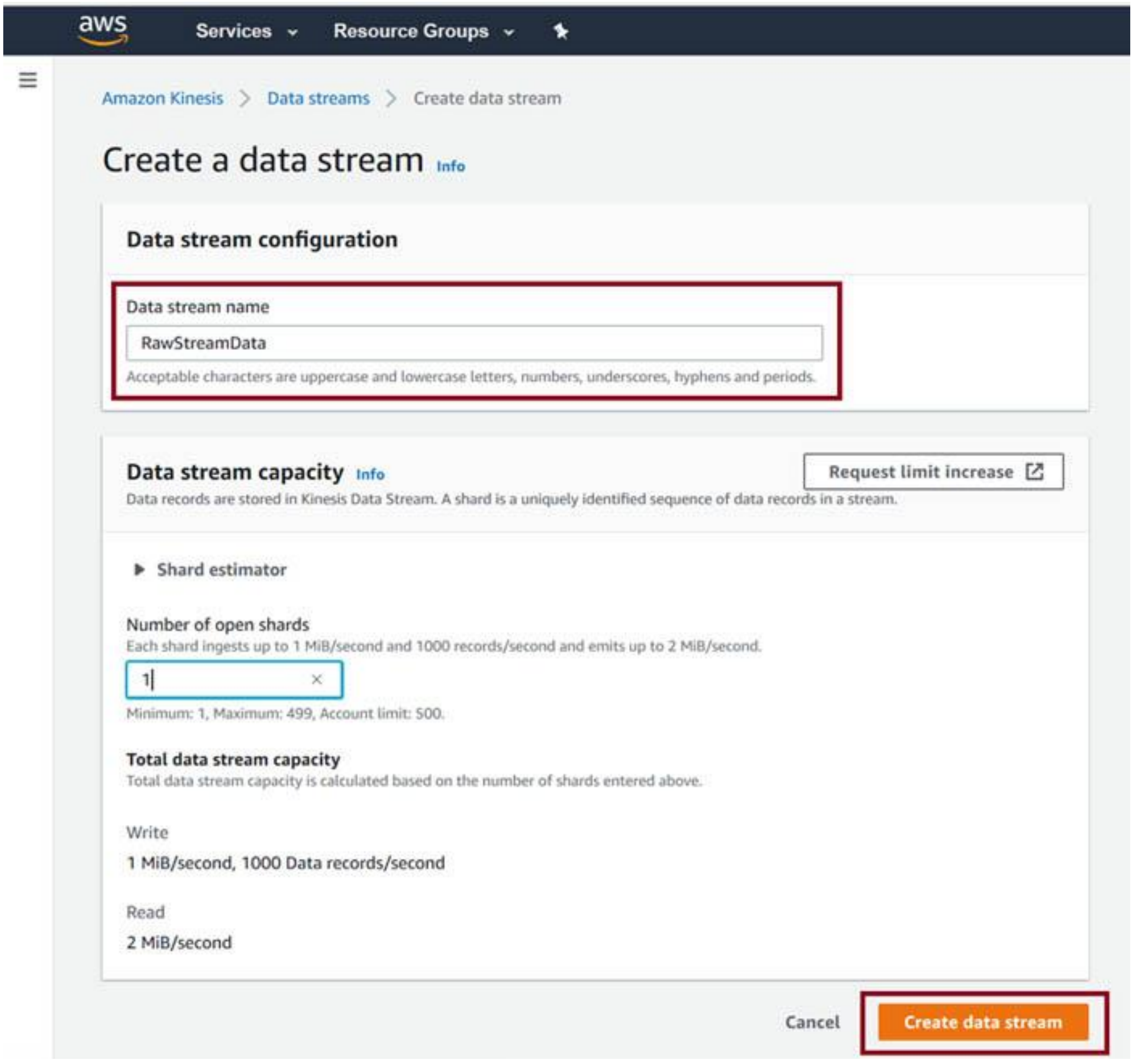
To view MQTT messages:

1. Copy the topic that is unique to your registered device: $aws/things/deeplens_7Mi9AwnA Q/infer
2. Go to AWS IoT console, paste the topic in the Subscription topic input field and choose **Subscribe to topic**.

3. <u>Sending results from AWS DeepLens to a data stream</u>

In this section, you learn how to send messages from AWS DeepLens to a Kinesis data stream by configuring an AWS IoT rule.

1. On the Kinesis console, create a new data stream.
2. For **Data stream name**, enter a name.
3. For **Number of shards**, choose 1.
4. Choose **Create data stream**.



Amazon Kinesis > Data streams > Create data stream

Create a data stream Info

Data stream configuration

Data stream name

RawStreamData

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods.

Data stream capacity Info

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

Shard estimator

Number of open shards

Each shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second.

1

Minimum: 1, Maximum: 499, Account limit: 500.

Total data stream capacity

Total data stream capacity is calculated based on the number of shards entered above.

Write

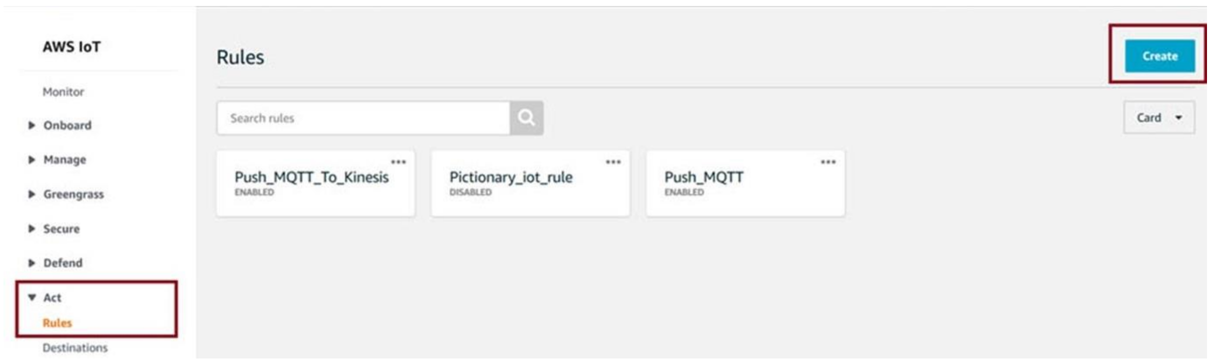
1 MiB/second, 1000 Data records/second

Read

2 MiB/second

4. On the AWS IoT console, under **Act**, choose **Rules**.

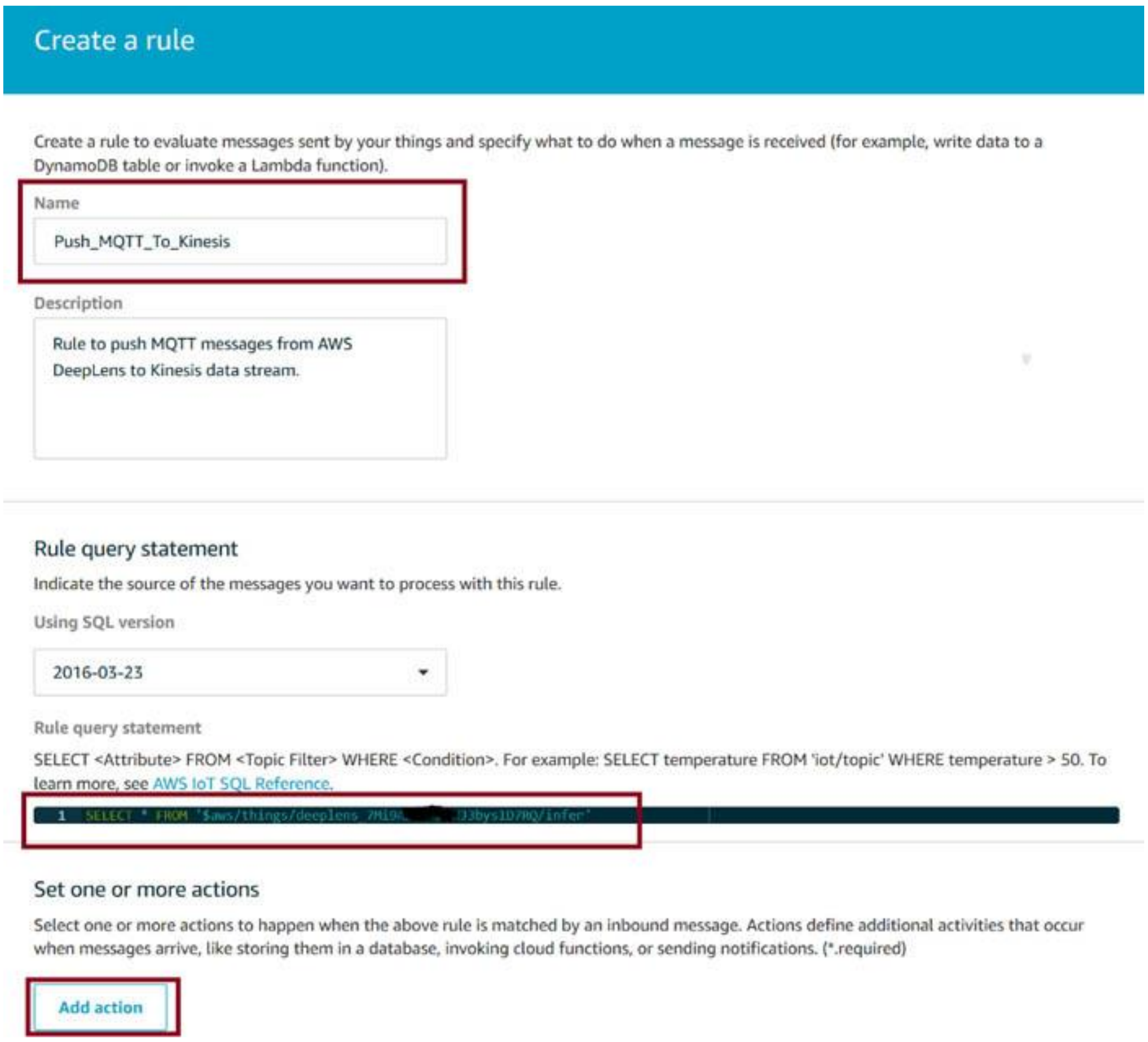
5. Choose **Create** to set up a rule to push MQTT messages from AWS DeepLens to the newly created data stream.



1. On the **Create a rule** page, enter a name for your rule.

2. For **Rule query statement**, enter the DeepLens device MQTT topic.

3. Choose **Add action**.



Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name: Push_MQTT_To_Kinesis

Description: Rule to push MQTT messages from AWS DeepLens to Kinesis data stream.

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version: 2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM '$aws/things/deeplens_7Mi9A[redacted]J3bys1D7RQ/infer'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

4. Choose **Send a message to an Amazon Kinesis Stream**.

5. Choose **Configuration**.

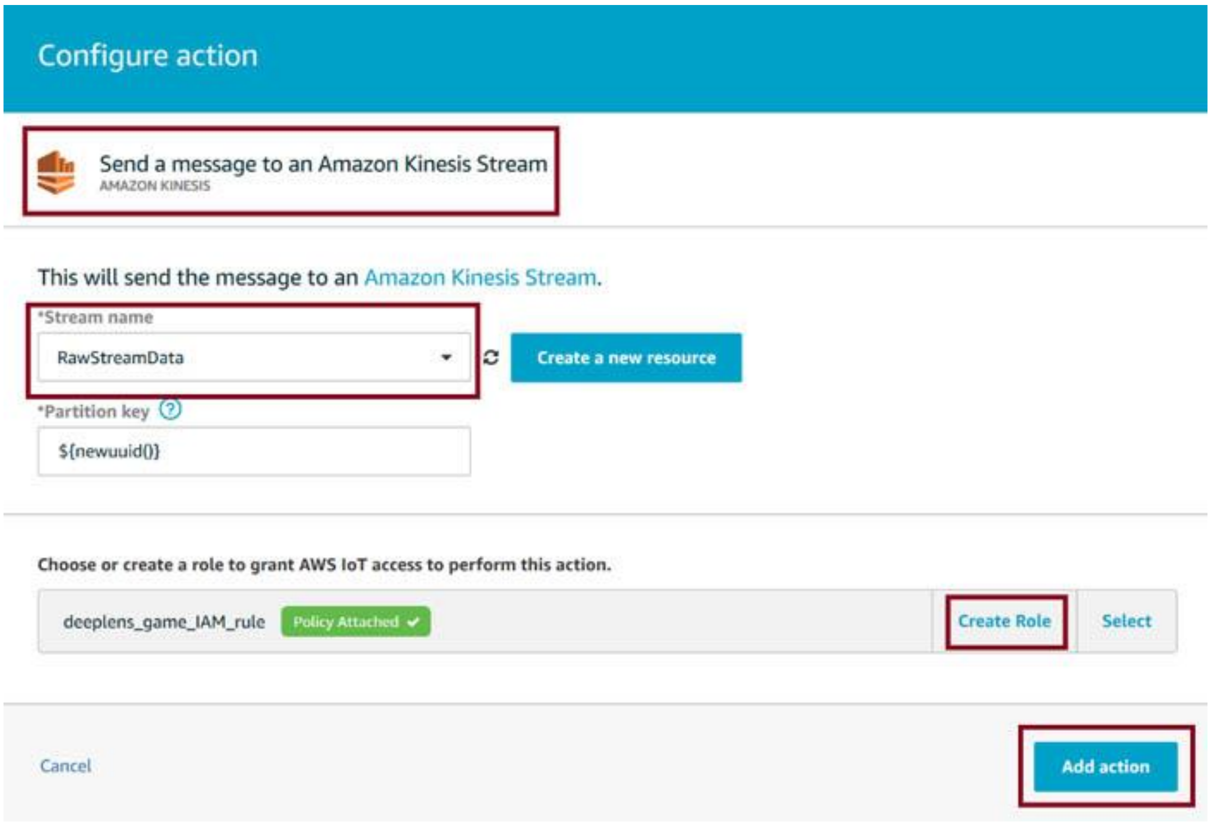
6. Choose the data stream you created earlier.

7. For **Partition key**, enter ${newuuid()}.

8. Choose **Create a new role** or **Update role**.

9. Choose **Add action**.

10. Choose **Create rule** to finish the setup.


Configure action

Send a message to an Amazon Kinesis Stream
AMAZON KINESIS

This will send the message to an Amazon Kinesis Stream.

*Stream name: RawStreamData

*Partition key: ${newuuid()}

Choose or create a role to grant AWS IoT access to perform this action.

deeplens_game_IAM_rule — Policy Attached ✓

4.Creating an Alexa skill

In this section, you first create a Lambda function that queries a data stream and returns the sketches detected by AWS DeepLens to Alexa. Then, you create a custom Alexa skill to start playing the game.

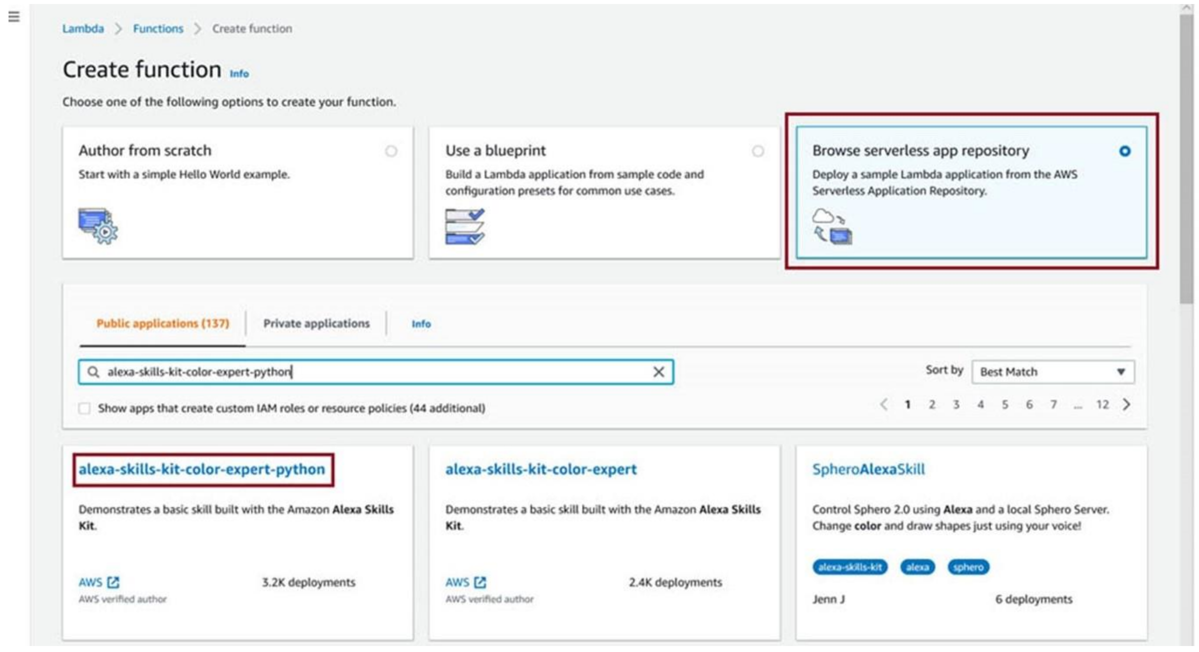
Creating a custom skill with Lambda

To create your custom skill in Lambda, complete the following steps:

1. On the Lambda console, create a new function.

The easiest way to create an Alexa skill is to create the function from the existing blueprints or serverless app repository provided by Lambda and overwrite the code with your own.

2. For **Create function**, choose **Browse serverless app repository**.
3. For **Public repositories**, search for and choose **alexa-skills-kit-color-expert-python**.



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Public applications (137) | Private applications | Info

alexa-skills-kit-color-expert-python

Sort by Best Match

Show apps that create custom IAM roles or resource policies (44 additional)

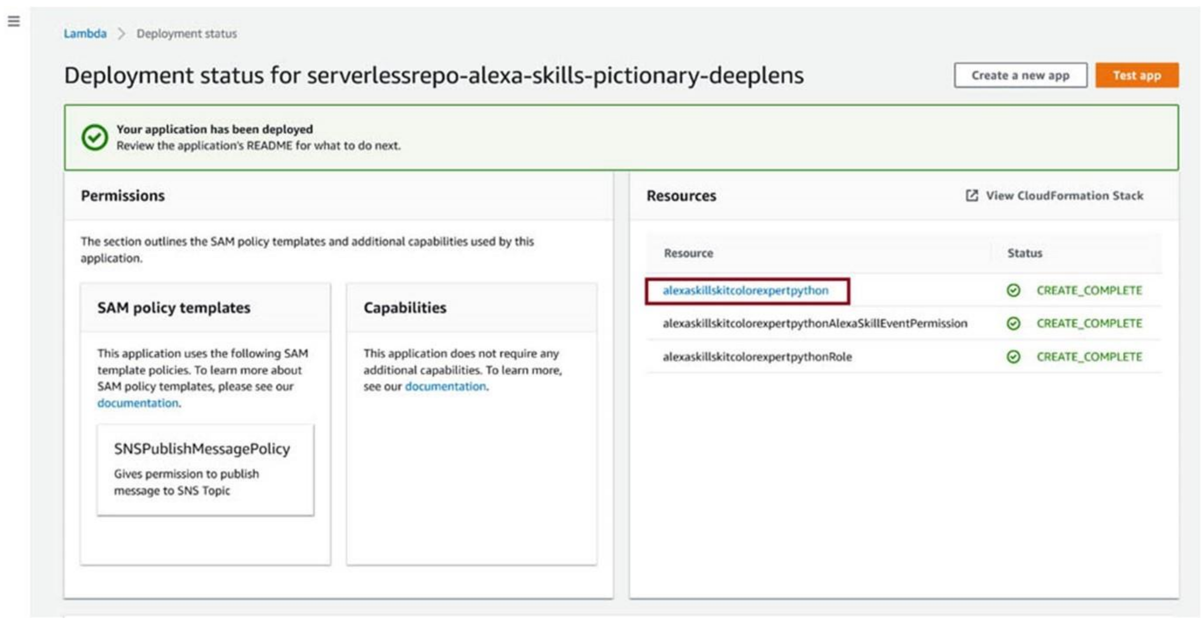
< 1 2 3 4 5 6 7 ... 12 >

alexa-skills-kit-color-expert-python
Demonstrates a basic skill built with the Amazon Alexa Skills Kit.
AWS
AWS verified author
3.2K deployments

alexa-skills-kit-color-expert
Demonstrates a basic skill built with the Amazon Alexa Skills Kit.
AWS
AWS verified author
2.4K deployments

SpheroAlexaSkill
Control Sphero 2.0 using Alexa and a local Sphero Server. Change color and draw shapes just using your voice!
alexa-skills-kit alexa sphero
Jenn J
6 deployments

4. Under **Application settings**, enter an application name and TopicNameParameter.
5. Choose **Deploy**.
6. When the application has been deployed, open the Python file.


Lambda > Deployment status

Deployment status for serverlessrepo-alexa-skills-pictionary-deeplens

Your application has been deployed
Review the application's README for what to do next.

Permissions

The section outlines the SAM policy templates and additional capabilities used by this application.

SAM policy templates

This application uses the following SAM template policies. To learn more about SAM policy templates, please see our documentation.

SNSPublishMessagePolicy
Gives permission to publish message to SNS Topic

Capabilities

This application does not require any additional capabilities. To learn more, see our documentation.

Resources

Resource	Status
alexaskillskitcolorexpertpython	CREATE_COMPLETE
alexaskillskitcolorexpertpythonAlexaSkillEventPermission	CREATE_COMPLETE
alexaskillskitcolorexpertpythonRole	CREATE_COMPLETE

7. Download the alexa-lambda-function.py file onto your computer.

8. Copy the Python code from the file and replace the sample code in the lambda_function.py file in the **Function code** section.

9. Set the **Timeout** value to 20 seconds.

You now need to give your Lambda function IAM permissions to read data from the data stream.

10. In your Lambda function editor, choose **Permissions**.

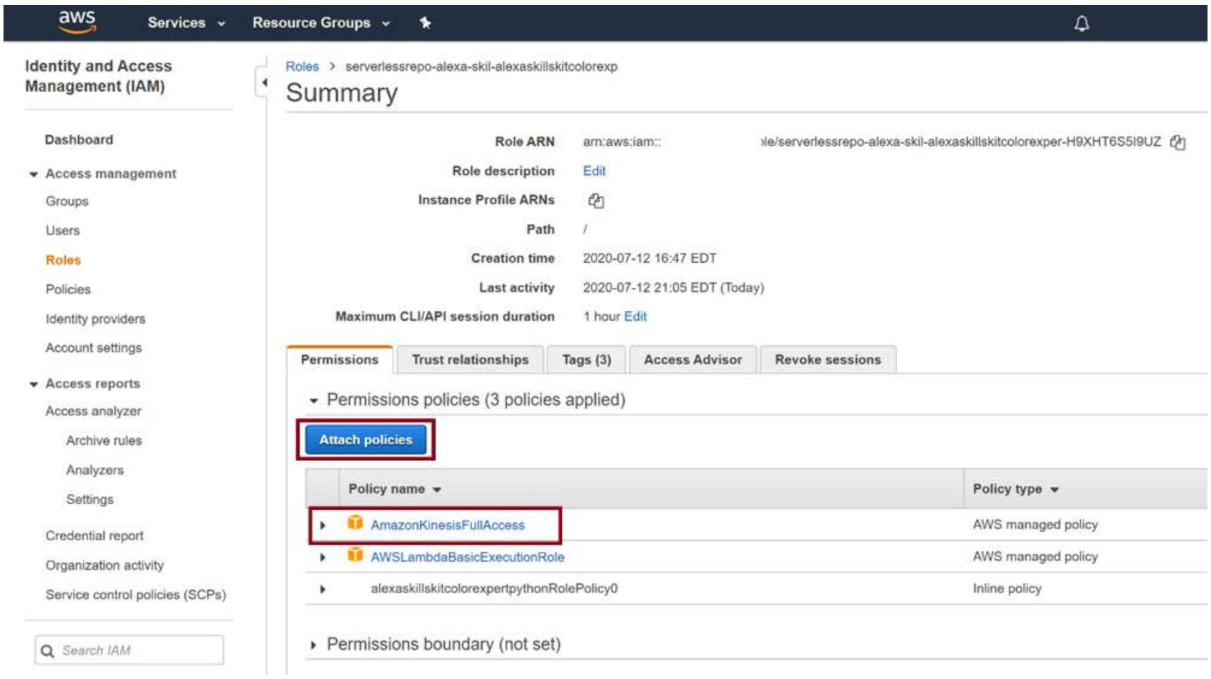
11. Choose the **Role name** under the **Execution role.**

You're directed to the IAM role editor.

12. In the editor, choose **Attach policies**.

13. Enter `Kinesis` and choose **AmazonKinesisFullAccess**.

14. Choose **Attach policy**.



Roles > serverlessrepo-alexa-skil-alexaskillskitcolorexp

Summary

Role ARN	arn:aws:iam:: ›le/serverlessrepo-alexa-skil-alexaskillskitcolorexper-H9XHT6S5I9UZ
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2020-07-12 16:47 EDT
Last activity	2020-07-12 21:05 EDT (Today)
Maximum CLI/API session duration	1 hour Edit

Permissions policies (3 policies applied)

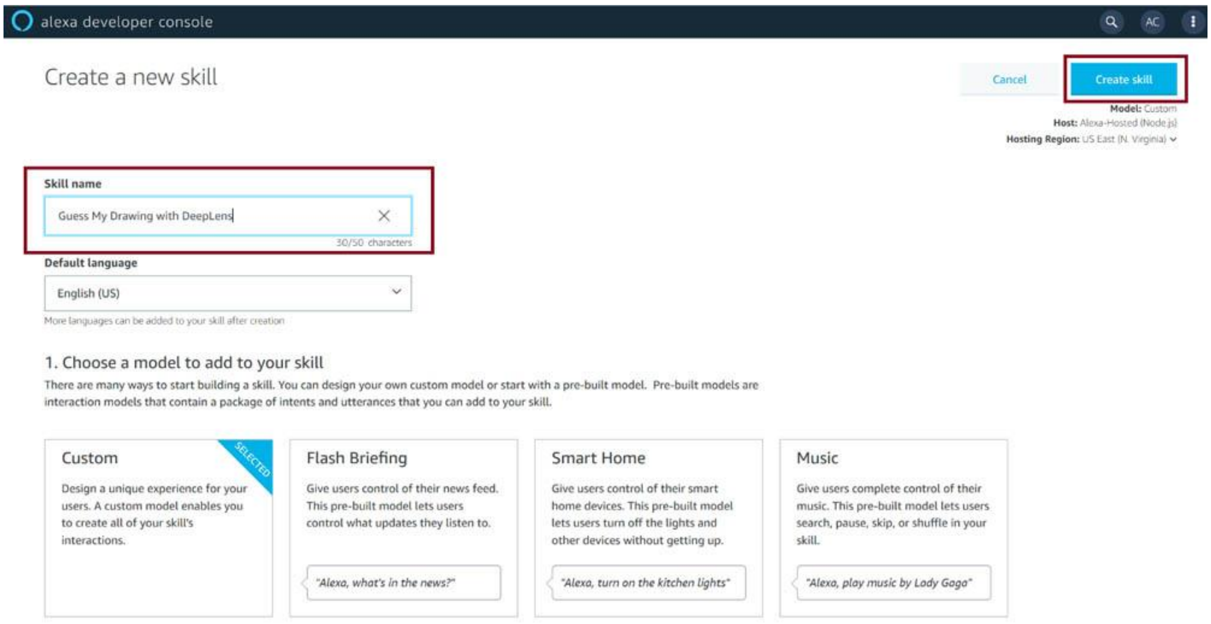
Policy name	Policy type
AmazonKinesisFullAccess	AWS managed policy
AWSLambdaBasicExecutionRole	AWS managed policy
alexaskillskitcolorexpertpythonRolePolicy0	Inline policy

Permissions boundary (not set)

5.Creating a custom skill to play the game

To create your second custom skill to start playing the game, complete the following steps:

1. Log in to the Alexa Developer Console.
2. Create a new custom Alexa skill.
3. On the Create a new skill page, for **Skill name**, enter a skill name.
4. For **Choose a model to add to your skill**, choose **Custom**.
5. For **Choose a method to host your skill's backend resources,** choose **Provision your own.**
6. Choose **Create skill**.


Create a new skill

Model: Custom
Host: Alexa-Hosted (Node.js)
Hosting Region: US East (N. Virginia)

Skill name: Guess My Drawing with DeepLens
30/50 characters

Default language: English (US)
More languages can be added to your skill after creation

1. Choose a model to add to your skill
There are many ways to start building a skill. You can design your own custom model or start with a pre-built model. Pre-built models are interaction models that contain a package of intents and utterances that you can add to your skill.

Custom (SELECTED): Design a unique experience for your users. A custom model enables you to create all of your skill's interactions.

Flash Briefing: Give users control of their news feed. This pre-built model lets users control what updates they listen to. "Alexa, what's in the news?"

Smart Home: Give users control of their smart home devices. This pre-built model lets users turn off the lights and other devices without getting up. "Alexa, turn on the kitchen lights"

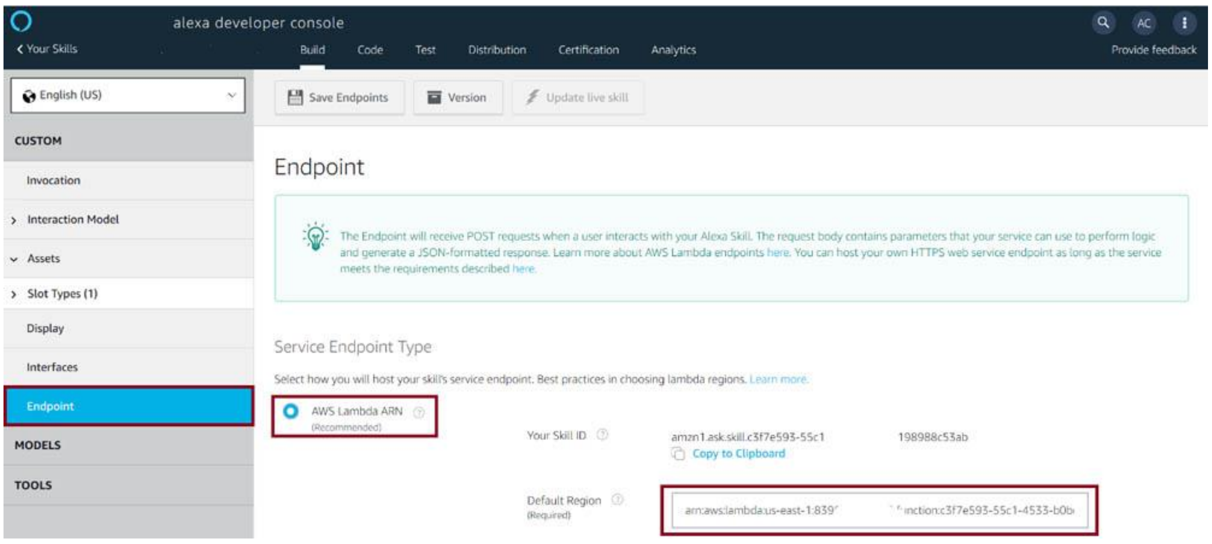
Music: Give users complete control of their music. This pre-built model lets users search, pause, skip, or shuffle in your skill. "Alexa, play music by Lady Gaga"

7. On the next page, choose the default template to add your skill.

8. Choose **Continue with template**.

After about 1–2 minutes, your skill appears on the console.

9. In the **Endpoint** section, enter the Amazon Resource Name (ARN) of the Lambda function created for the Alexa skill in the previous step.


Endpoint

The Endpoint will receive POST requests when a user interacts with your Alexa Skill. The request body contains parameters that your service can use to perform logic and generate a JSON-formatted response. Learn more about AWS Lambda endpoints here. You can host your own HTTPS web service endpoint as long as the service meets the requirements described here.

Service Endpoint Type
Select how you will host your skill's service endpoint. Best practices in choosing lambda regions. Learn more.

AWS Lambda ARN (Recommended)

Your Skill ID: amzn1.ask.skill.c3f7e593-55c1 198988c53ab
Copy to Clipboard

Default Region (Required): arn:aws:lambda:us-east-1:839 nction:c3f7e593-55c1-4533-b0b

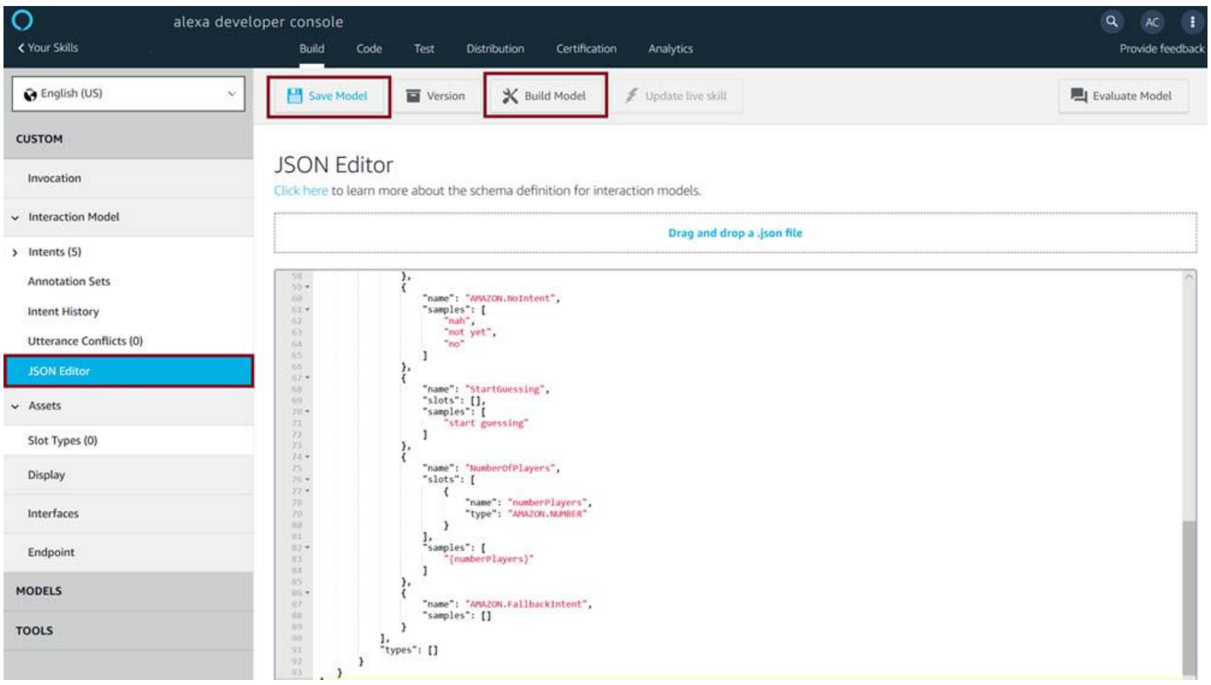
10. Download alexa-skill-json-code.txt onto your computer.

11. Copy the code from the file and paste in the Alexa skill JSON editor to automatically configure intents and sample utterances for the custom skill.

In the Alexa architecture, intents can be thought of as distinct functions that a skill can perform. Intents can take arguments that are known here as *slots*.

12. Choose **Save Model** to apply the changes.

13. Choose **Build Model**.



JSON Editor

Click here to learn more about the schema definition for interaction models.

```
                },
                {
                    "name": "AMAZON.NoIntent",
                    "samples": [
                        "nah",
                        "not yet",
                        "no"
                    ]
                },
                {
                    "name": "StartGuessing",
                    "slots": [],
                    "samples": [
                        "start guessing"
                    ]
                },
                {
                    "name": "NumberOfPlayers",
                    "slots": [
                        {
                            "name": "numberPlayers",
                            "type": "AMAZON.NUMBER"
                        }
                    ],
                    "samples": [
                        "{numberPlayers}"
                    ]
                },
                {
                    "name": "AMAZON.FallbackIntent",
                    "samples": []
                }
            ],
            "types": []
        }
    }
```

14. On the Lambda console, open the Lambda function for the Alexa skill you created earlier.

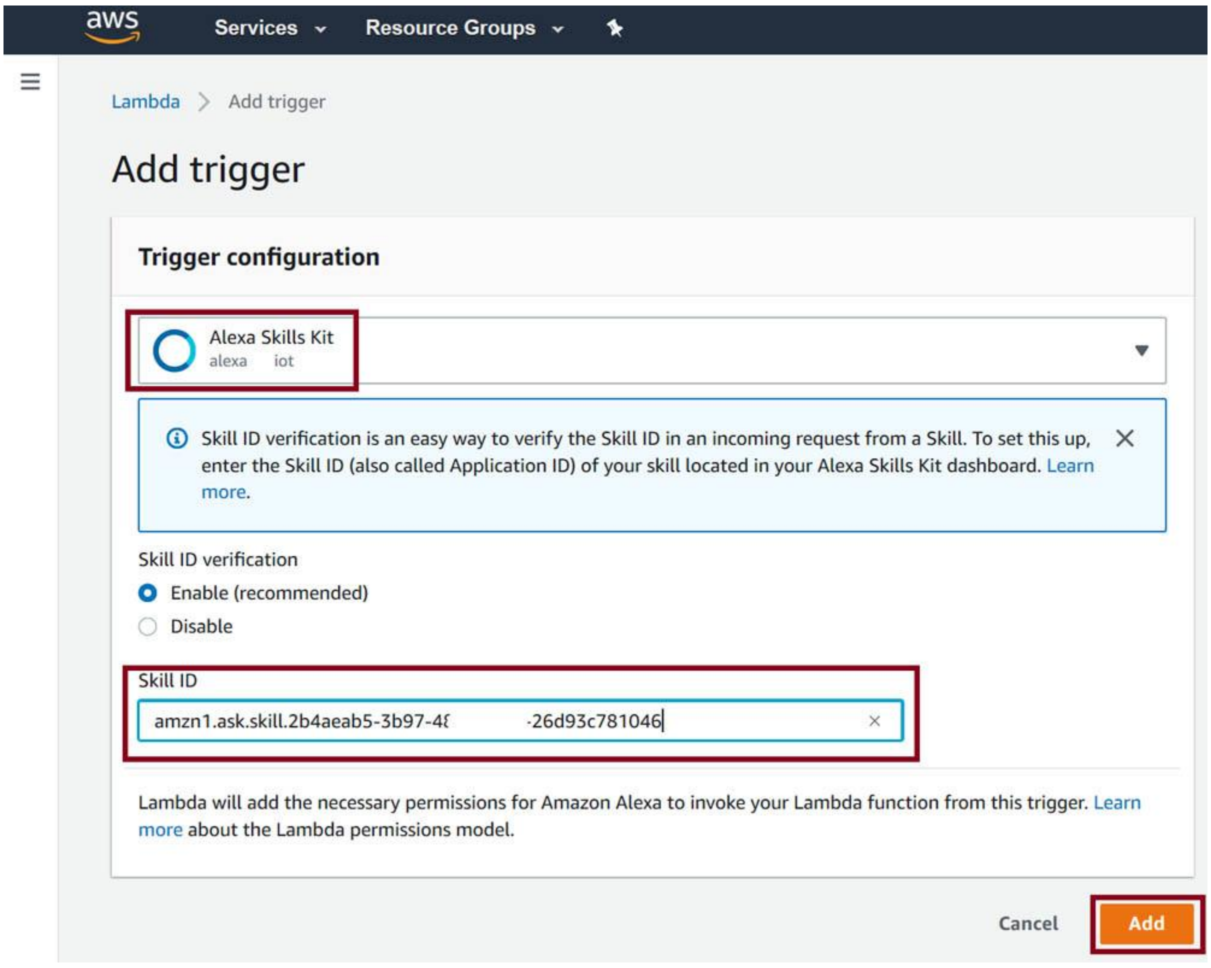
You need to enable the skill by adding a trigger to the Lambda function.

15. Choose **Add trigger**.

16. Choose **Alexa Skills Kit**.

17. For **Skill ID**, enter the ID for the Alexa skill you created.

18. Choose **Add**.



Lambda > Add trigger

Add trigger

Trigger configuration

Alexa Skills Kit
alexa iot

Skill ID verification is an easy way to verify the Skill ID in an incoming request from a Skill. To set this up, enter the Skill ID (also called Application ID) of your skill located in your Alexa Skills Kit dashboard. Learn more.

Skill ID verification
Enable (recommended) [✓]
Disable

Skill ID
amzn1.ask.skill.2b4aeab5-3b97-4ξ -26d93c781046

Lambda will add the necessary permissions for Amazon Alexa to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel Add

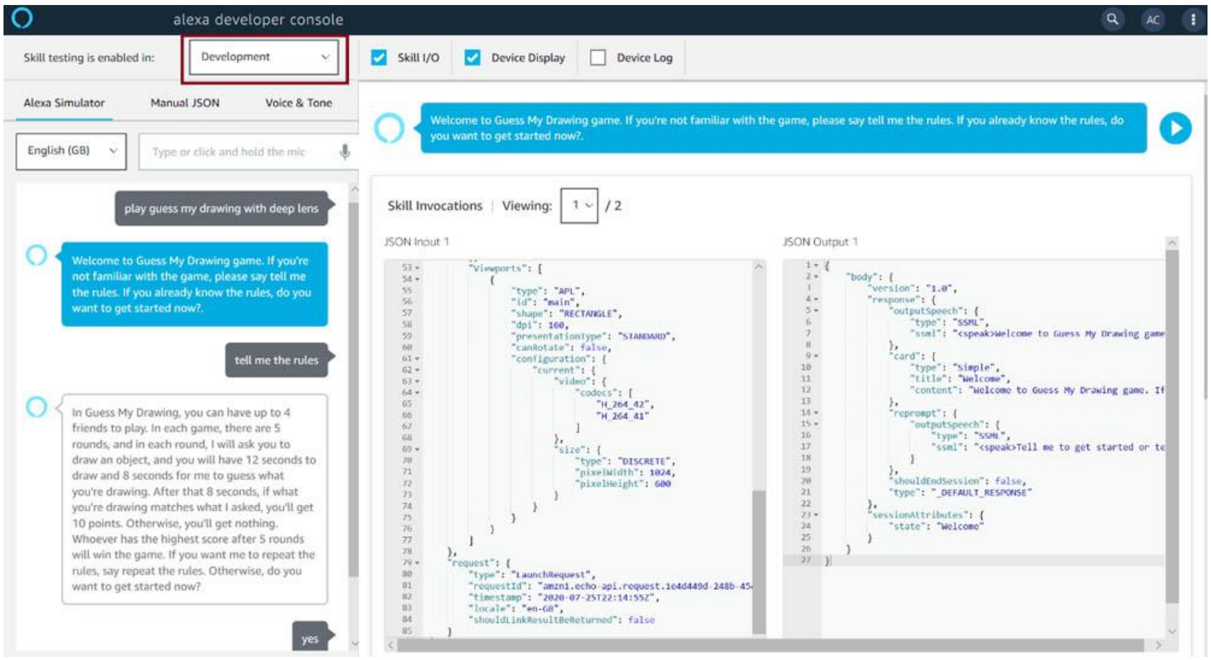
6. TESTING THE SKILL

Your Alexa skill is now ready to tell you the drawings detected by AWS DeepLens. To test with an Alexa-enabled device (such as an Amazon Echo), register the device with the same email address you used to sign up for your developer account on the Amazon Developer Portal. You can invoke your skill with the wake word and your invocation name: “Alexa, Play Guess My Drawing with DeepLens.”

The language in your Alexa companion app should match with the language chosen in your developer account. Alexa considers English US and English UK to be separate languages.

Alternatively, the **Test** page includes a simulator that lets you test your skill without a device. For **Skill testing is enabled in**, choose **Development**. You can test your skill with the phrase, “Alexa, Play Guess My Drawing with DeepLens.”

Windows 10 users can download the free Alexa app from the Microsoft Store and interact with it from their PC.



Skill testing is enabled in: Development

English (GB)

☑ Skill I/O ☑ Device Display ☐ Device Log

play guess my drawing with deep lens

Welcome to Guess My Drawing game. If you're not familiar with the game, please say tell me the rules. If you already know the rules, do you want to get started now?.

tell me the rules

In Guess My Drawing, you can have up to 4 friends to play. In each game, there are 5 rounds, and in each round, I will ask you to draw an object, and you will have 12 seconds to draw and 8 seconds for me to guess what you're drawing. After that 8 seconds, if what you're drawing matches what I asked, you'll get 10 points. Otherwise, you'll get nothing. Whoever has the highest score after 5 rounds will win the game. If you want me to repeat the rules, say repeat the rules. Otherwise, do you want to get started now?

yes

Skill Invocations | Viewing: 1 / 2

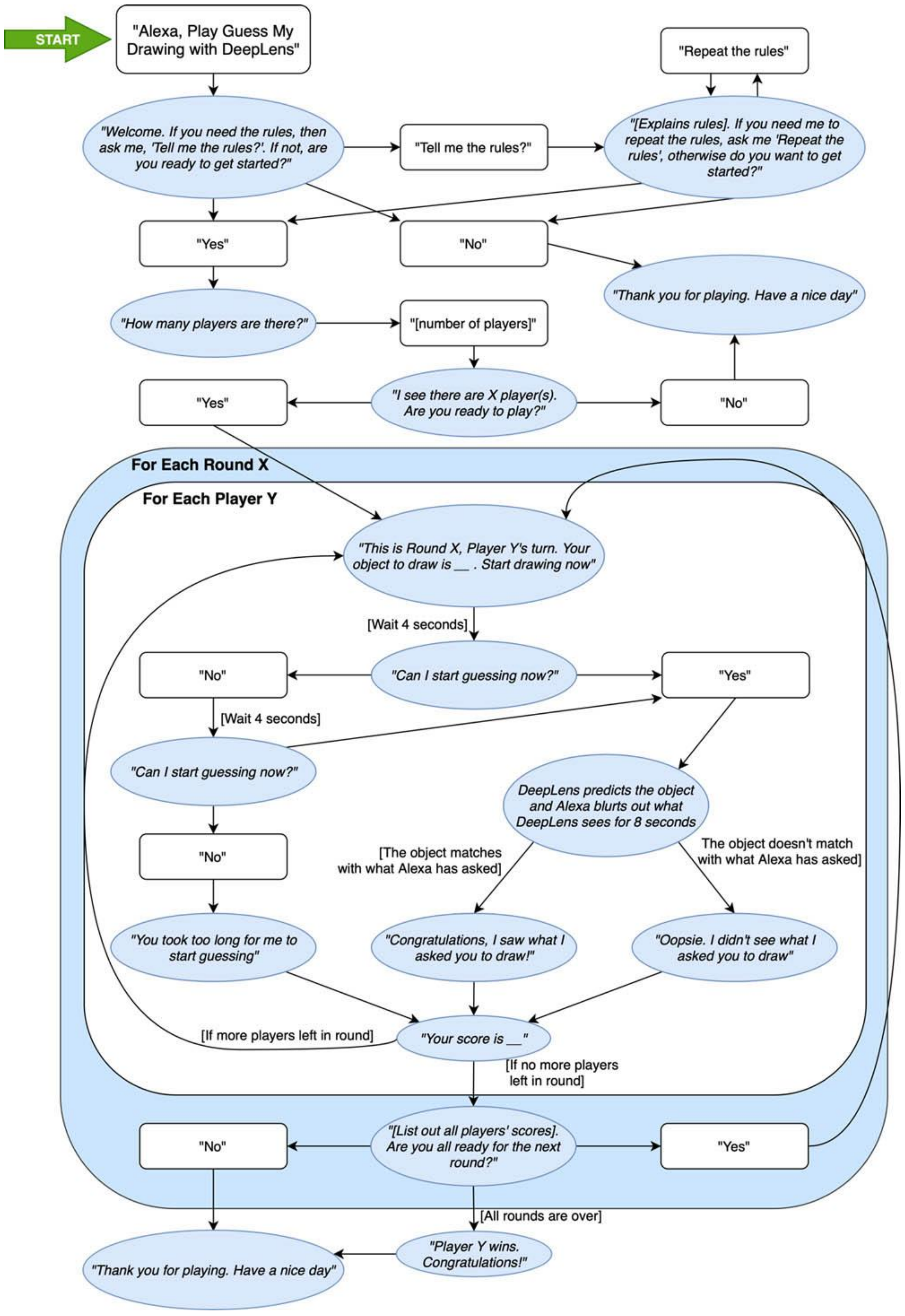
JSON Input 1

```
        "Viewports": [
            {
                "type": "APL",
                "id": "main",
                "shape": "RECTANGLE",
                "dpi": 160,
                "presentationType": "STANDARD",
                "canRotate": false,
                "configuration": {
                    "current": {
                        "video": {
                            "codecs": [
                                "H_264_42",
                                "H_264_41"
                            ]
                        },
                        "size": {
                            "type": "DISCRETE",
                            "pixelWidth": 1024,
                            "pixelHeight": 600
                        }
                    }
                }
            }
        ]
    },
    "request": {
        "type": "LaunchRequest",
        "requestId": "amzn1.echo-api.request.1e4d449d-248b-45
        "timestamp": "2020-07-25T22:14:55Z",
        "locale": "en-GB",
        "shouldLinkResultBeReturned": false
    }
```

JSON Output 1

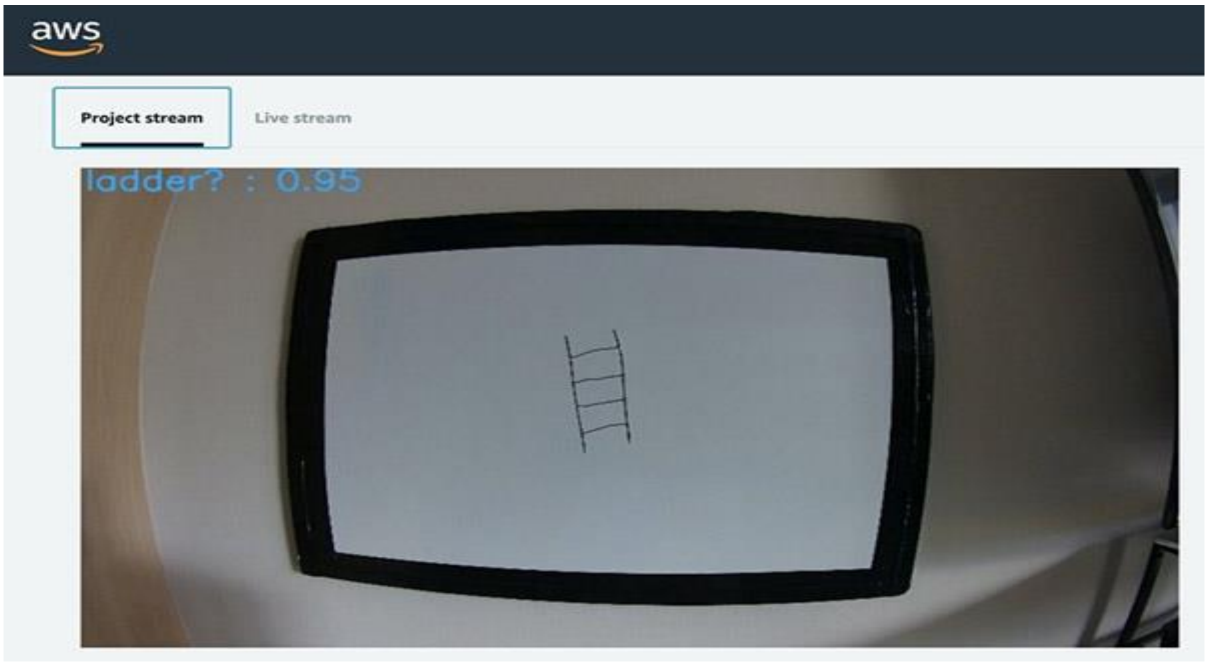
```
{
    "body": {
        "version": "1.0",
        "response": {
            "outputSpeech": {
                "type": "SSML",
                "ssml": "<speak>Welcome to Guess My Drawing game
            },
            "card": {
                "type": "Simple",
                "title": "Welcome",
                "content": "Welcome to Guess My Drawing game. If
            },
            "reprompt": {
                "outputSpeech": {
                    "type": "SSML",
                    "ssml": "<speak>Tell me to get started or te
                }
            },
            "shouldEndSession": false,
            "type": "_DEFAULT_RESPONSE"
        },
        "sessionAttributes": {
            "state": "Welcome"
        }
    }
}
```

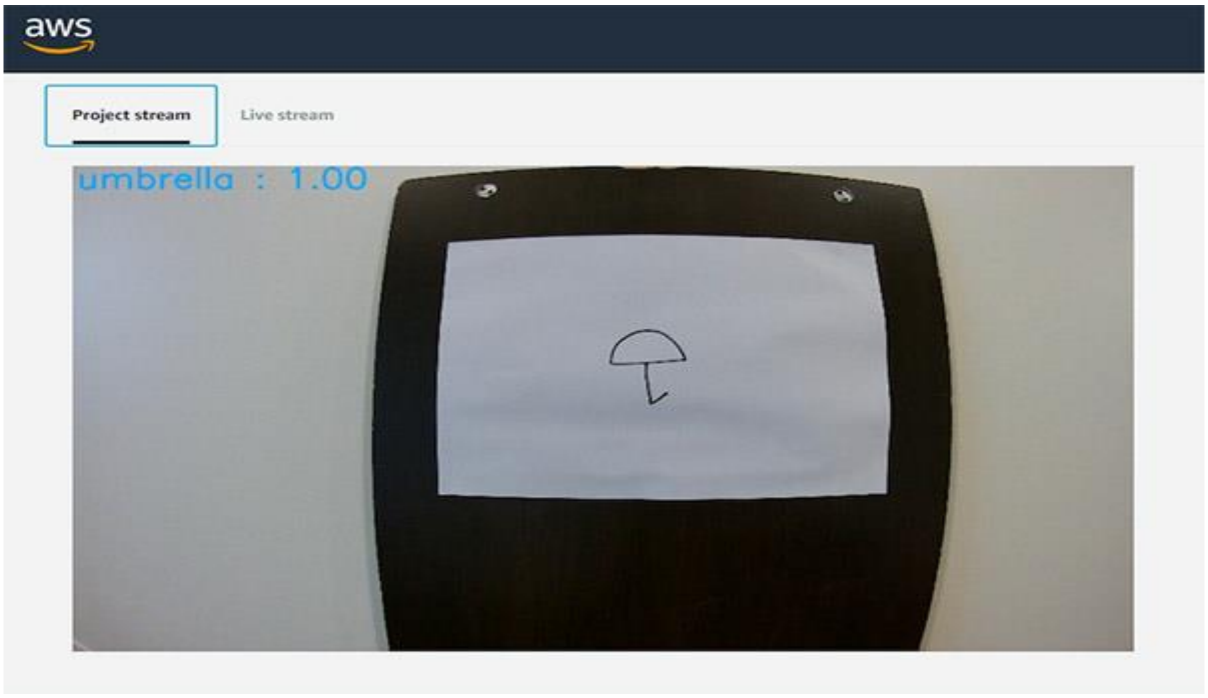
The following diagram shows the user interaction flow of our game.


START
"Alexa, Play Guess My Drawing with DeepLens"
"Welcome. If you need the rules, then ask me, 'Tell me the rules?'. If not, are you ready to get started?"
"Tell me the rules?"
"[Explains rules]. If you need me to repeat the rules, ask me 'Repeat the rules', otherwise do you want to get started?"
"Repeat the rules"
"Yes"
"No"
"Thank you for playing. Have a nice day"
"How many players are there?"
"[number of players]"
"I see there are X player(s). Are you ready to play?"
"Yes"
"No"
For Each Round X
For Each Player Y
"This is Round X, Player Y's turn. Your object to draw is __ . Start drawing now"
[Wait 4 seconds]
"Can I start guessing now?"
"No"
"Yes"
[Wait 4 seconds]
"Can I start guessing now?"
DeepLens predicts the object and Alexa blurts out what DeepLens sees for 8 seconds
"No"
[The object matches with what Alexa has asked]
The object doesn't match with what Alexa has asked]
"You took too long for me to start guessing"
"Congratulations, I saw what I asked you to draw!"
"Oopsie. I didn't see what I asked you to draw"
[If more players left in round]
"Your score is __"
[If no more players left in round]
"[List out all players' scores]. Are you all ready for the next round?"
"No"
"Yes"
[All rounds are over]
"Player Y wins. Congratulations!"
"Thank you for playing. Have a nice day"

7. EXPERIMENTAL RESULTS

The following images show the prediction outputs of our model with the name of an object and its probability. You need to have your AWS DeepLens located in front of a rectangular-shaped whiteboard or a piece of white paper to ensure that the edges are visible in the frame.





8. CONCLUSION

Building a Pictionary style game with AWS DeepLens and Amazon Alexa can be a fun and engaging way to incorporate both computer vision and voice-enabled technology into a game. The DeepLens device can be used to capture images of the player's drawings, while the Alexa device can be used to provide clues or prompts for the player to draw.

To build this game, you would need to train a custom object detection model on the DeepLens device to recognize the objects that the player is drawing. You could also use pre-trained models to recognize common objects or categories, such as animals, vehicles, or household items.

Once the model is trained, the game can be played by having the player draw an object or concept while the DeepLens captures an image of the drawing. The image can then be sent to the Alexa device, which can provide a clue or prompt for the player to guess the object. The player can then respond with their guess, and the game can continue with another round.

Overall, building a Pictionary style game with AWS DeepLens and Amazon Alexa can be a fun and innovative way to combine computer vision and voice-enabled technology into a game that can be enjoyed by players of all ages.

9. LINKS TO OTHER SOURCES OF PROJECT

GitHub Link: **https://github.com/pravallikakandi/CSC-PROJECT-AWS**

PPT Link: **https://kluniversityin-my.sharepoint.com/:f:/g/personal/2000031181_kluniversity_in/EqA-CIeN84ZGvbM179AeDyMBtAjaXXkLiYhXX1Op5d5TWg?e=WeUXU7**

YouTube Link: **https://youtu.be/k4PgIBkPcoo**

LinkedIn Link: **https://www.linkedin.com/posts/pravallika-kandi_aws-awscloud-machinelearning-activity-7052814358950678528-RG22?utm_source=share&utm_medium=member_desktop**