AWS Rekognition Lab

By: amaranth-grain (Christy)

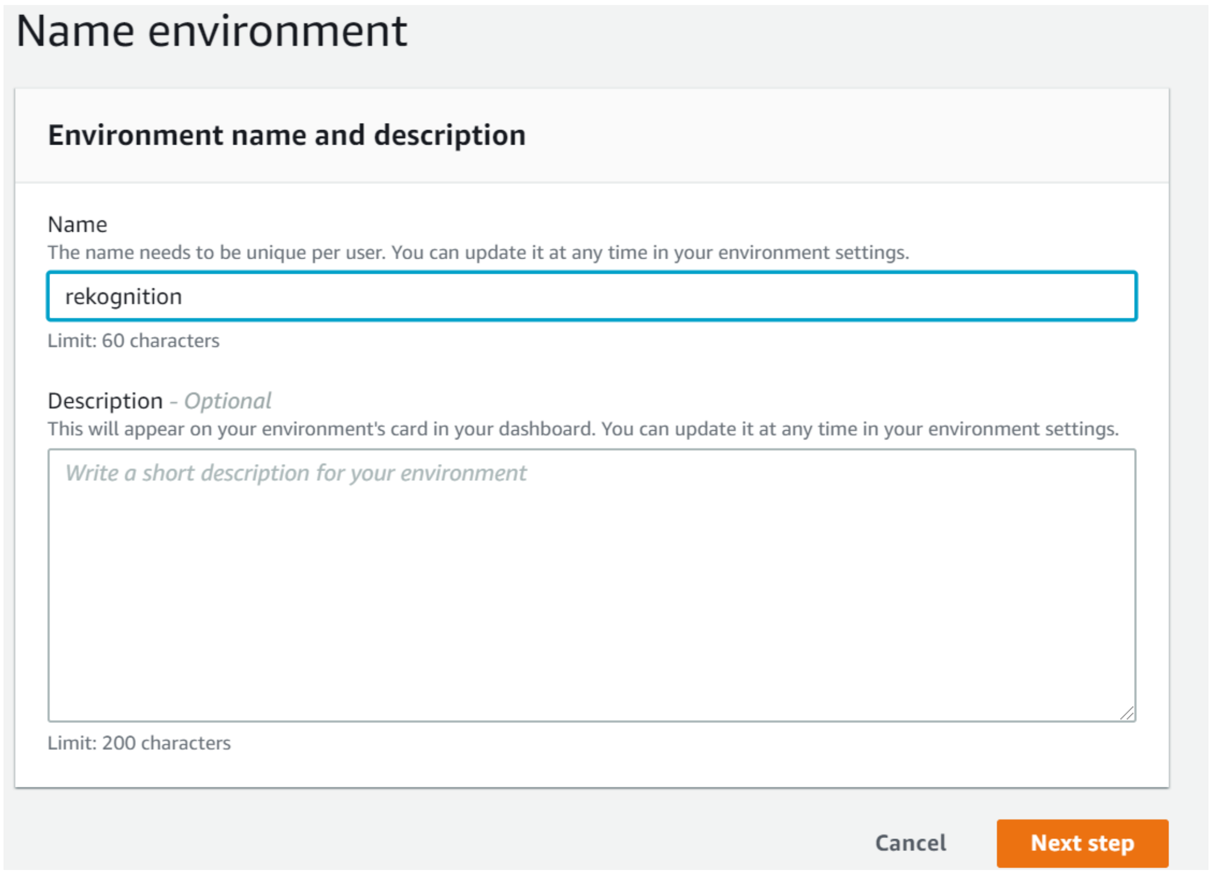
Amazon Rekognition is a microservice that makes it easy to add image, video, and facial analysis to applications. It requires no machine learning expertise to begin using it.

Common uses include user verification, people counting, content moderation, text detection, celebrity recognition, pathing, and tracking.

To start, we will try out Amazon Rekognition's ability to analyse stock photos.

PART A: GETTING STARTED WITH REKOGNITION

1. Create a new Cloud 9 environment.
   Name the environment (**rekognition**) and click "Next Step".



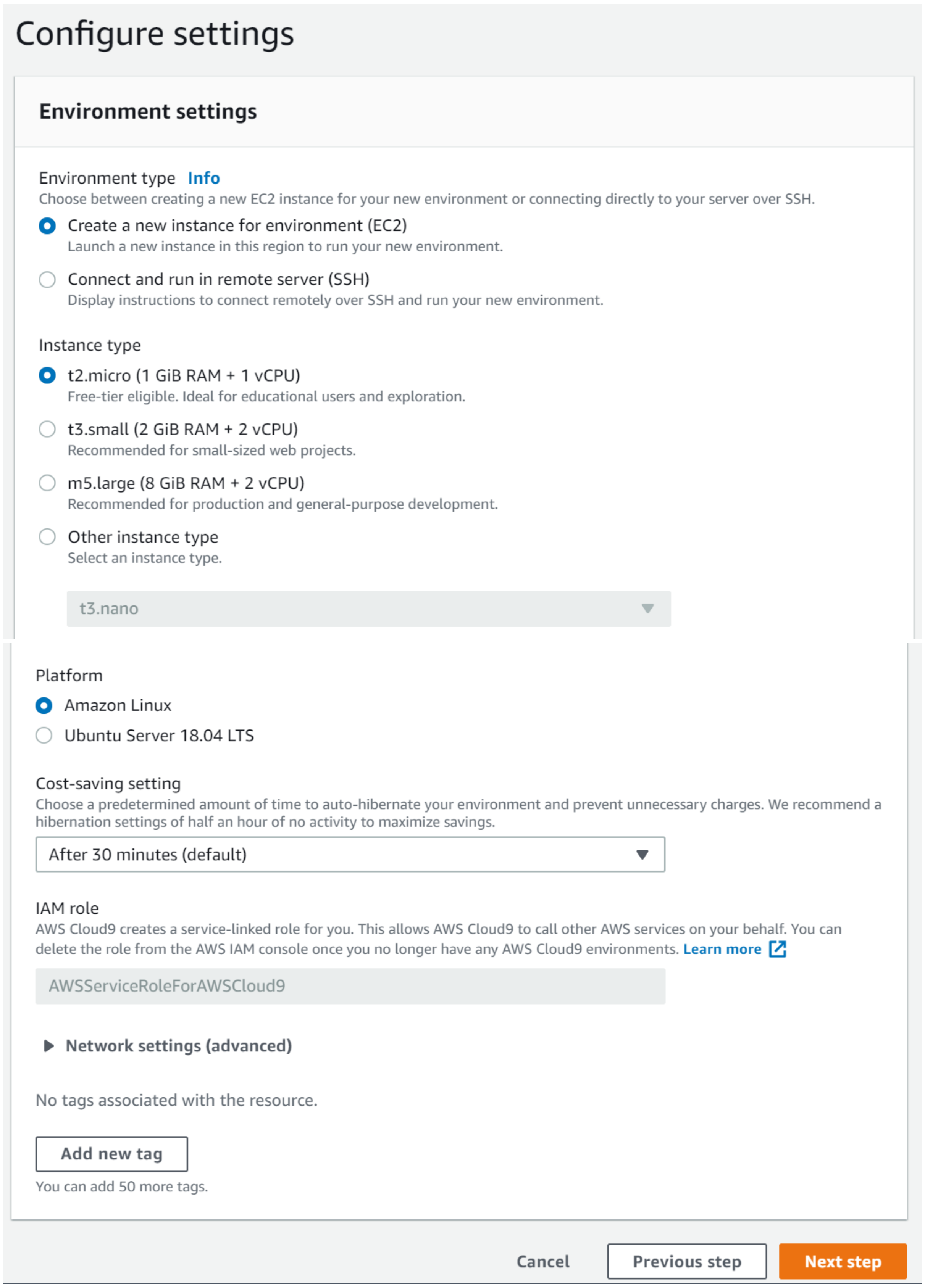
Name environment

Environment name and description

Name
The name needs to be unique per user. You can update it at any time in your environment settings.
rekognition
Limit: 60 characters

Description - *Optional*
This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.
Write a short description for your environment
Limit: 200 characters

2. Accept defaults in "Configure Settings". Click "Next Step" then "Create Environment".



Configure settings

Environment settings

Environment type Info
Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

[✓ Create a new instance for environment (EC2)]
Launch a new instance in this region to run your new environment.

Connect and run in remote server (SSH)
Display instructions to connect remotely over SSH and run your new environment.

Instance type

[✓ t2.micro (1 GiB RAM + 1 vCPU)]
Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU)
Recommended for small-sized web projects.

m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and general-purpose development.

Other instance type
Select an instance type.

t3.nano

Platform

[✓ Amazon Linux]

Ubuntu Server 18.04 LTS

Cost-saving setting
Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation settings of half an hour of no activity to maximize savings.

After 30 minutes (default)

IAM role
AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more

AWSServiceRoleForAWSCloud9

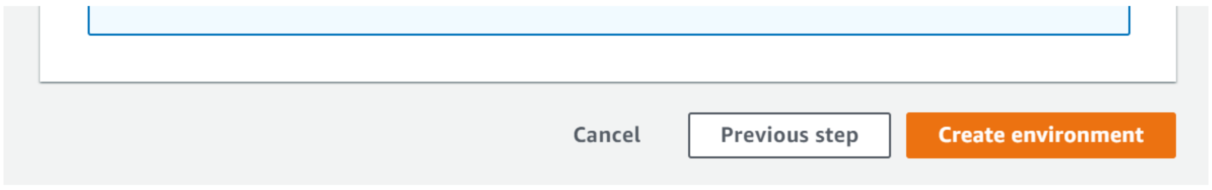
▶ **Network settings (advanced)**

No tags associated with the resource.

Add new tag

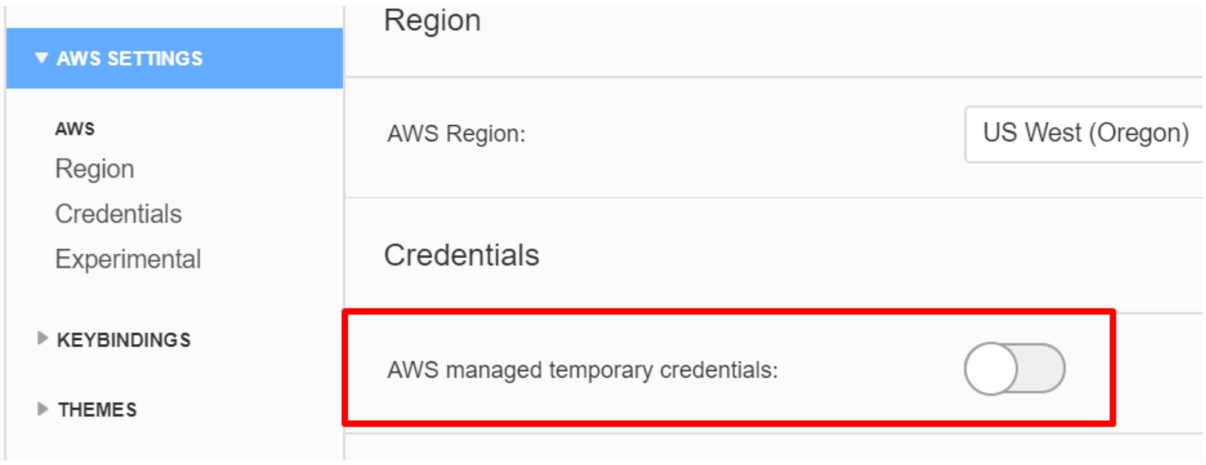
You can add 50 more tags.

Cancel | Previous step | Next step



3. Cloud9 by default has AWS CLI installed, so we only need to configure the credentials.

   Click on the gear on the top right corner of the Cloud9 IDE, and browse the sidebar for “AWS Settings”. Turn “AWS managed temporary credentials” off.



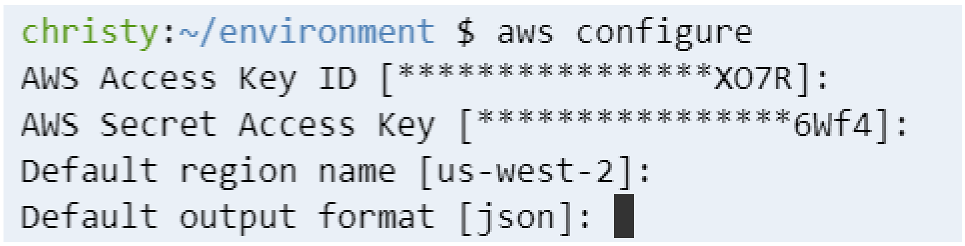
AWS Region: US West (Oregon)

AWS managed temporary credentials: off

   Go back to CLI and type in aws configure.
   Enter your IAM role credentials from the **.csv** file you previously downloaded.

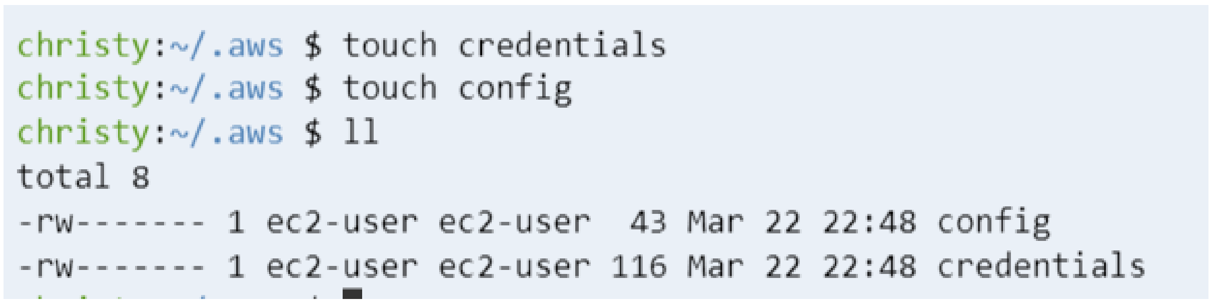
   We will be using region **us-west-2**.

   Confirm the details have been added correctly by typing aws configure again. It should look something like this:



```
christy:~/environment $ aws configure
AWS Access Key ID [****************XO7R]:
AWS Secret Access Key [****************6Wf4]:
Default region name [us-west-2]:
Default output format [json]:
```

4. Navigate to the .aws directory and create two new files, **credentials** and **config**. (If .aws is not present, create the directory as a subdirectory of root. mkdir .aws from root directory.)



```
christy:~/.aws $ touch credentials
christy:~/.aws $ touch config
christy:~/.aws $ ll
total 8
-rw------- 1 ec2-user ec2-user  43 Mar 22 22:48 config
-rw------- 1 ec2-user ec2-user 116 Mar 22 22:48 credentials
```

Open the file **credentials** in an editor and add the following, replacing the values with your access key and secret key from the **.csv** file.

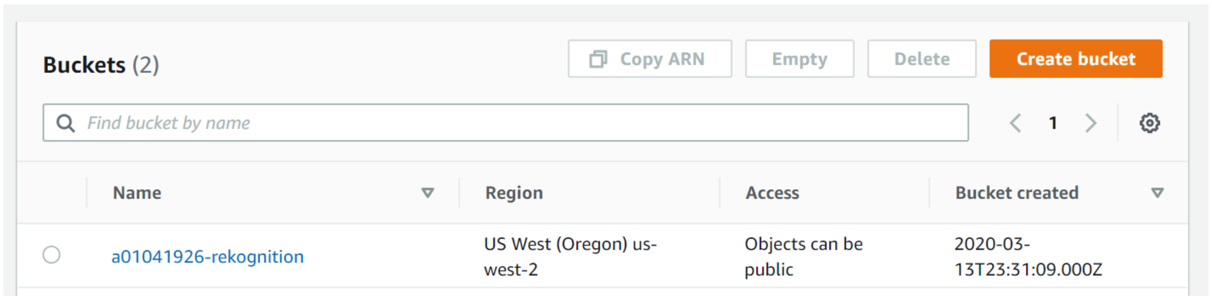
```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Open the file **config** and enter the following and save:

```
[default]
output = json
region = us-west-2
```

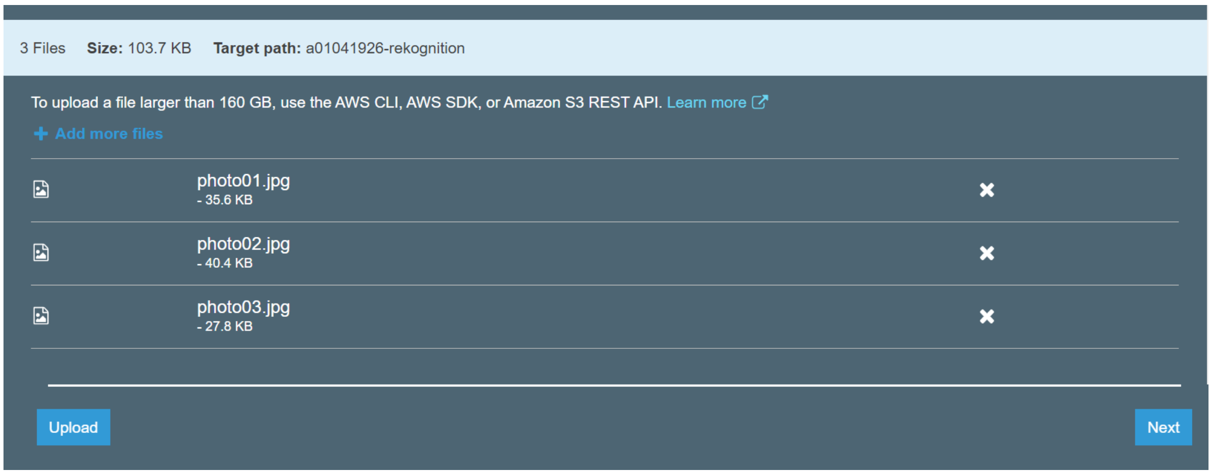
5. With AWS CLI basic set-up finished, we will create an S3 bucket to host our images for Rekognition to analyse.

   Create a new bucket and ensure it's in **us-west-2**. AWS Rekognition won't be able to access images hosted in S3 buckets in a region different from where it's run.



Name	Region	Access	Bucket created
a01041926-rekognition	US West (Oregon) us-west-2	Objects can be public	2020-03-13T23:31:09.000Z

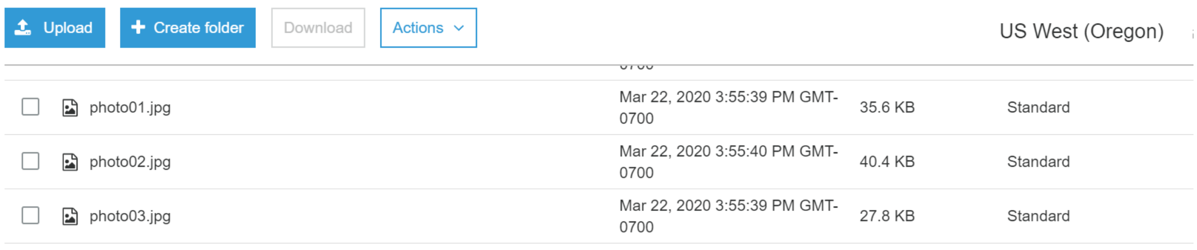
Upload the three stock photos provided in **images** folder.


3 Files **Size:** 103.7 KB **Target path:** a01041926-rekognition

To upload a file larger than 160 GB, use the AWS CLI, AWS SDK, or Amazon S3 REST API. Learn more

+ Add more files

	photo01.jpg - 35.6 KB	✖
	photo02.jpg - 40.4 KB	✖
	photo03.jpg - 27.8 KB	✖

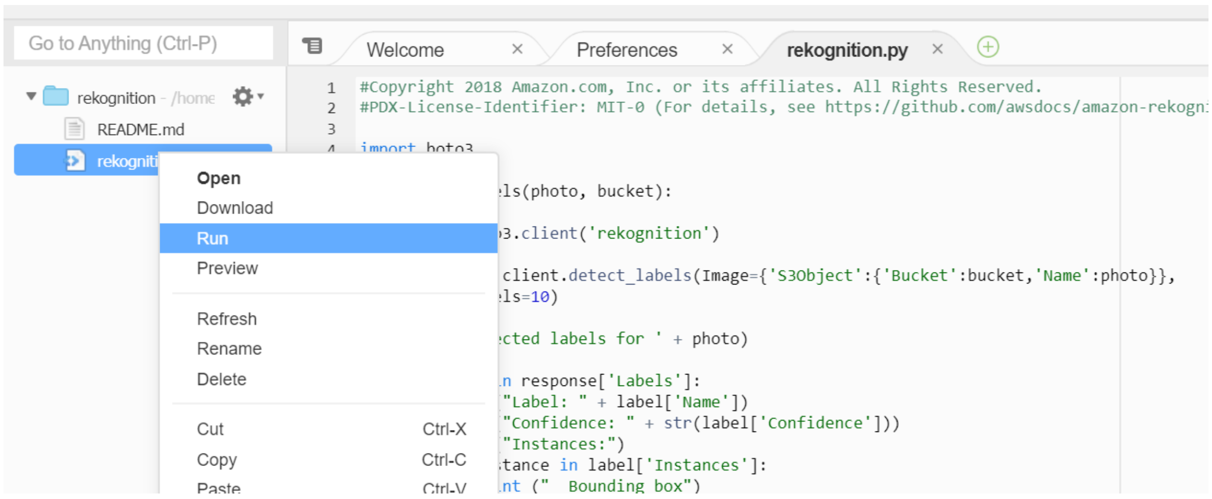


☐	photo01.jpg	Mar 22, 2020 3:55:39 PM GMT-0700	35.6 KB	Standard
☐	photo02.jpg	Mar 22, 2020 3:55:40 PM GMT-0700	40.4 KB	Standard
☐	photo03.jpg	Mar 22, 2020 3:55:39 PM GMT-0700	27.8 KB	Standard

6. Create a new file called **rekognition.py** in Cloud9 and copy and paste the following code:

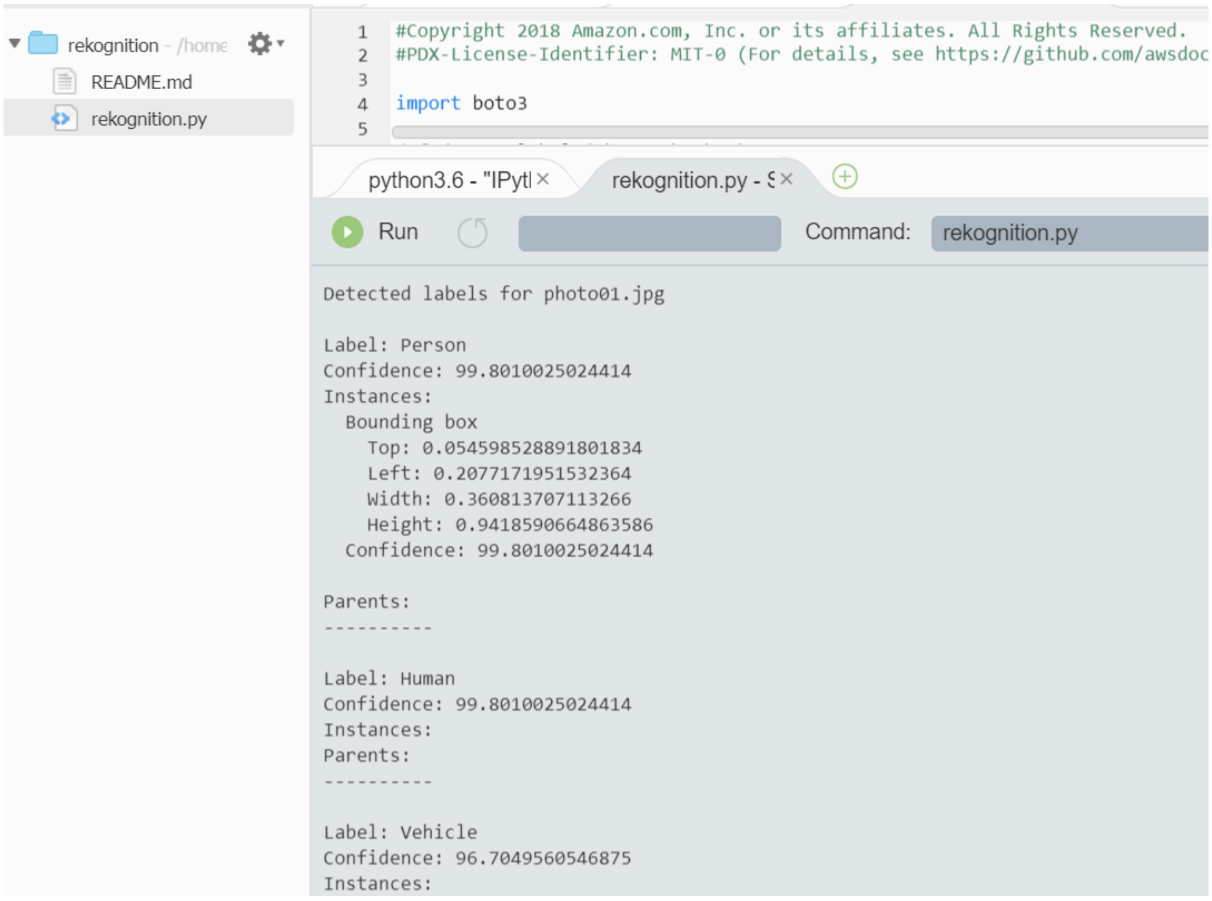
```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All
Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see
https://github.com/awsdocs/amazon-rekognition-developer-
guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_labels(photo, bucket):

    client=boto3.client('rekognition')

    response =
client.detect_labels(Image={'S3Object':{'Bucket':bucket,
'Name':photo}},
        MaxLabels=10)

    print('Detected labels for ' + photo)
    print()
    for label in response['Labels']:
        print ("Label: " + label['Name'])
        print ("Confidence: " +
str(label['Confidence']))
        print ("Instances:")
        for instance in label['Instances']:
            print ("  Bounding box")
            print ("    Top: " +
str(instance['BoundingBox']['Top']))
            print ("    Left: " +
str(instance['BoundingBox']['Left']))
            print ("    Width: " +
str(instance['BoundingBox']['Width']))
            print ("    Height: " +
str(instance['BoundingBox']['Height']))
            print ("  Confidence: " +
str(instance['Confidence']))
            print()

        print ("Parents:")
        for parent in label['Parents']:
            print ("   " + parent['Name'])
        print ("----------")
        print ()
    return len(response['Labels'])


def main():
    photo=''
    bucket=''
    label_count=detect_labels(photo, bucket)
    print("Labels detected: " + str(label_count))


if __name__ == "__main__":
    main()
```

Look through the code. What kind of changes do you have to make before you can run it? (HINT: You need to change two parameters. Use the unique S3 bucket name, not its ARN.)



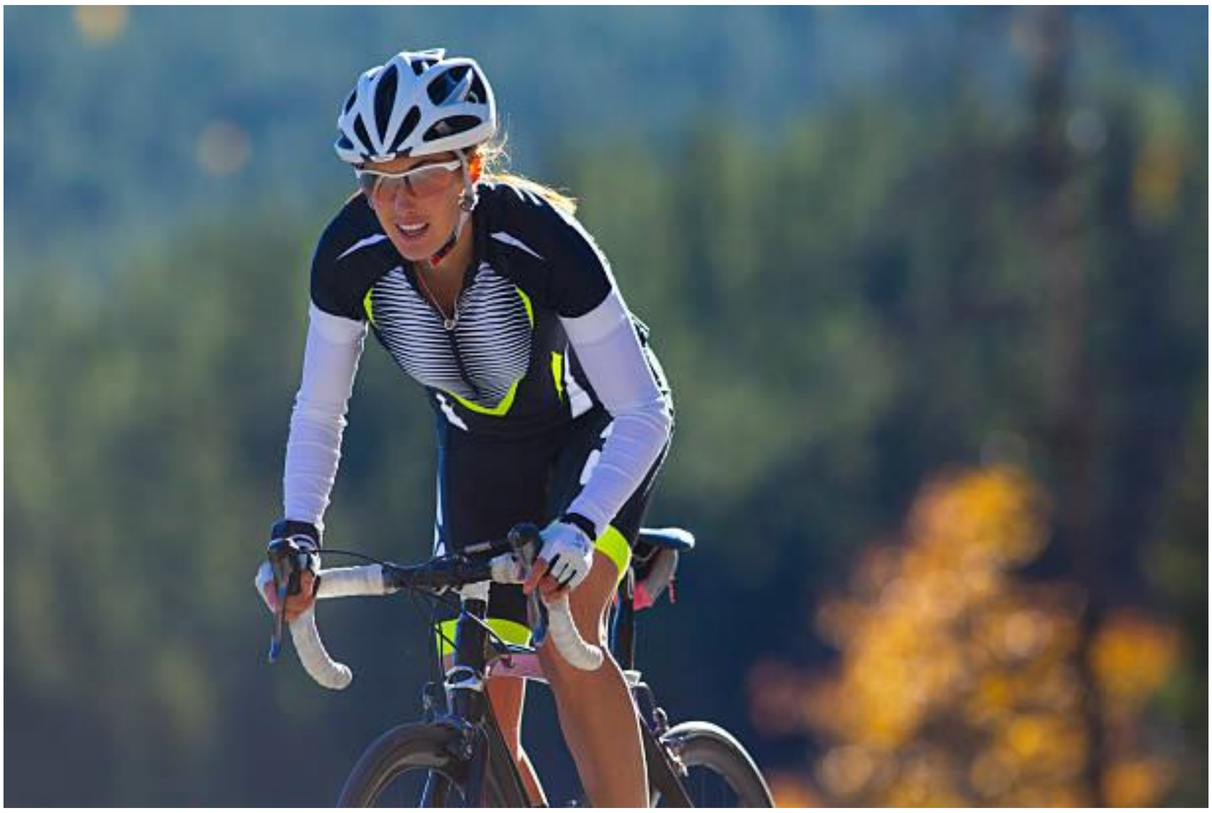
```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekogni

import boto3
ls(photo, bucket):

3.client('rekognition')

client.detect_labels(Image={'S3Object':{'Bucket':bucket,'Name':photo}},
ls=10)

cted labels for ' + photo)

n response['Labels']:
"Label: " + label['Name'])
"Confidence: " + str(label['Confidence']))
"Instances:")
tance in label['Instances']:
nt ("  Bounding box")
```

7. See the results for photo01.jpg.



```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdoc

import boto3
```

```
Detected labels for photo01.jpg

Label: Person
Confidence: 99.8010025024414
Instances:
  Bounding box
    Top: 0.054598528891801834
    Left: 0.2077171951532364
    Width: 0.360813707113266
    Height: 0.9418590664863586
  Confidence: 99.8010025024414

Parents:
----------

Label: Human
Confidence: 99.8010025024414
Instances:
Parents:
----------

Label: Vehicle
Confidence: 96.7049560546875
Instances:
```

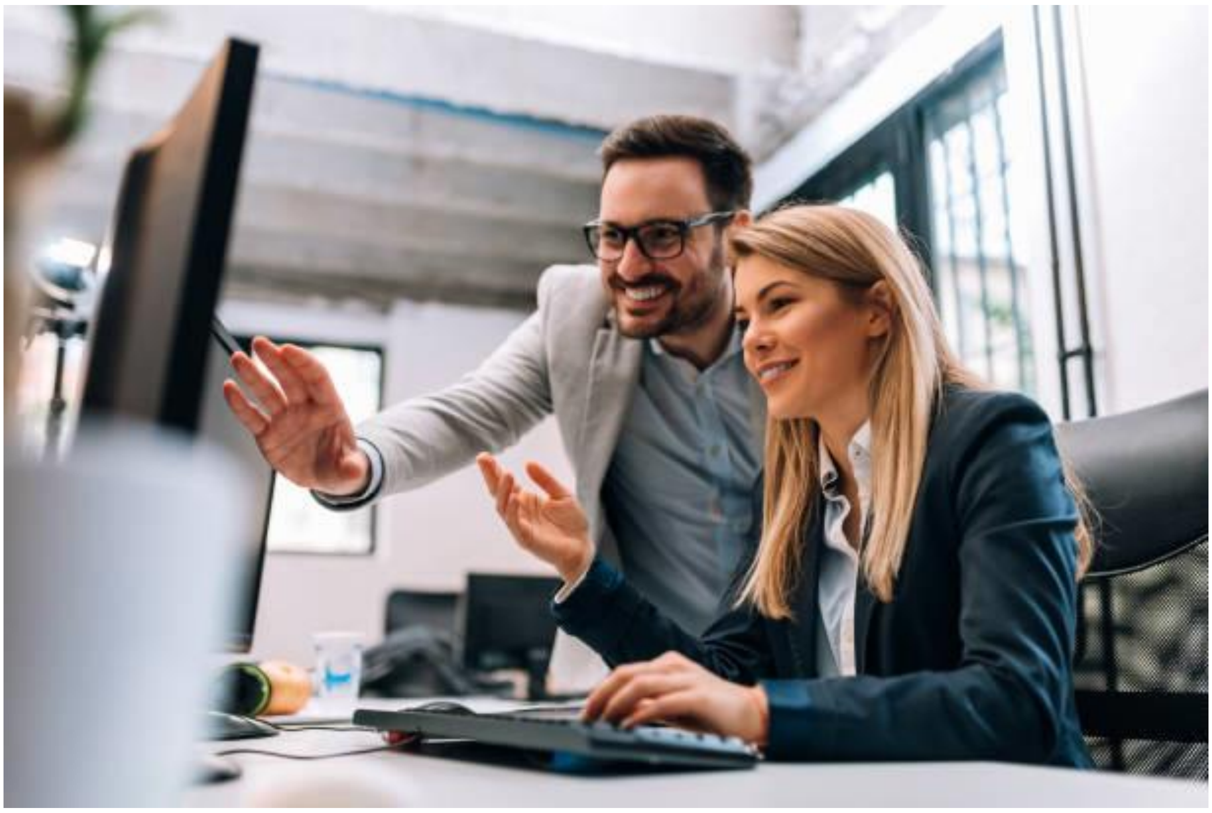


The code limited Rekognition to pick out a maximum of 10 labels it’s most confident about. They are summarised below, in decreasing confidence:

1. Person
2. Human
3. Vehicle
4. Bicycle
5.Transportation
6. Mountain Bike
7. Cyclist
8. Sport
9. Clothing
10. Helmet

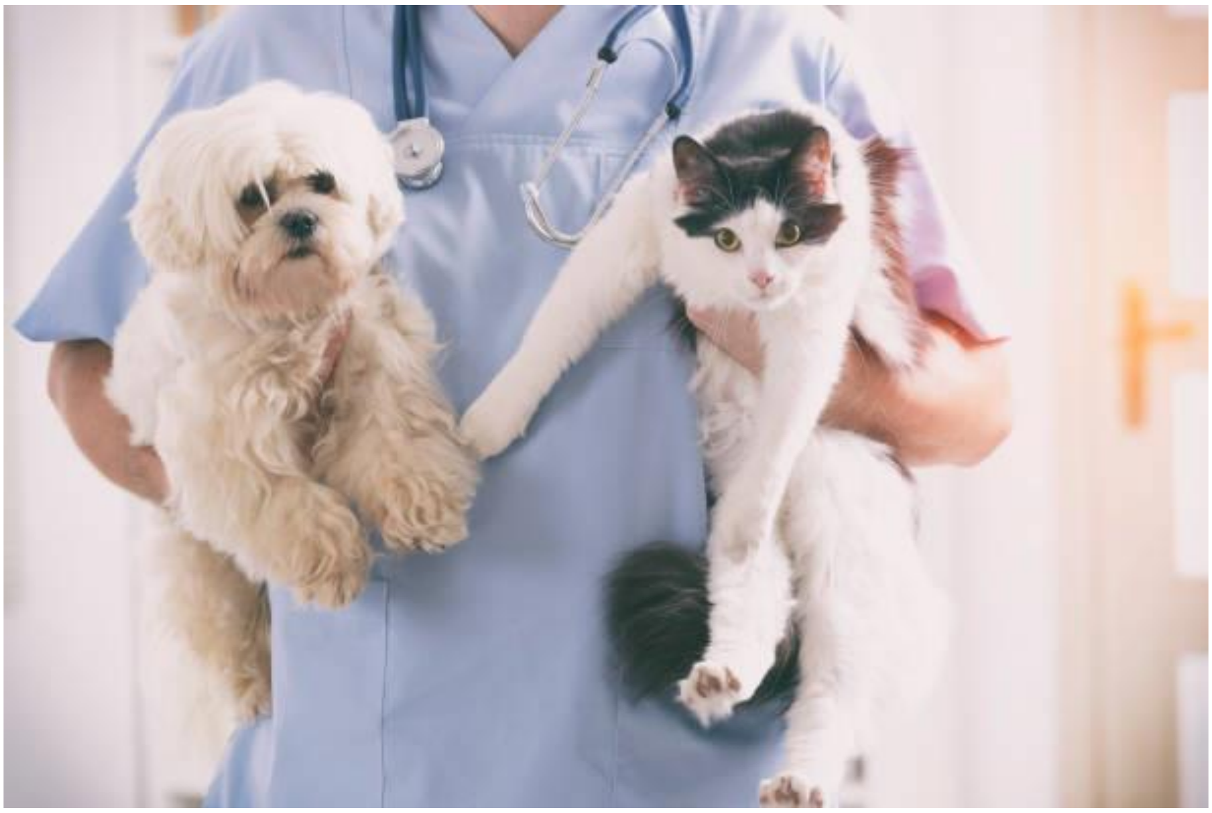
Pretty neat, right? Try it out with photo02.jpg and photo03.jpg and see if you get the same thing:

Photo02.jpg



1. Person
2. Sitting
3. Indoors
4. Executive
5. Crowd
6. Interview
7. Office
8. Room
9. Clothing
10. Head

Photo03.jpg



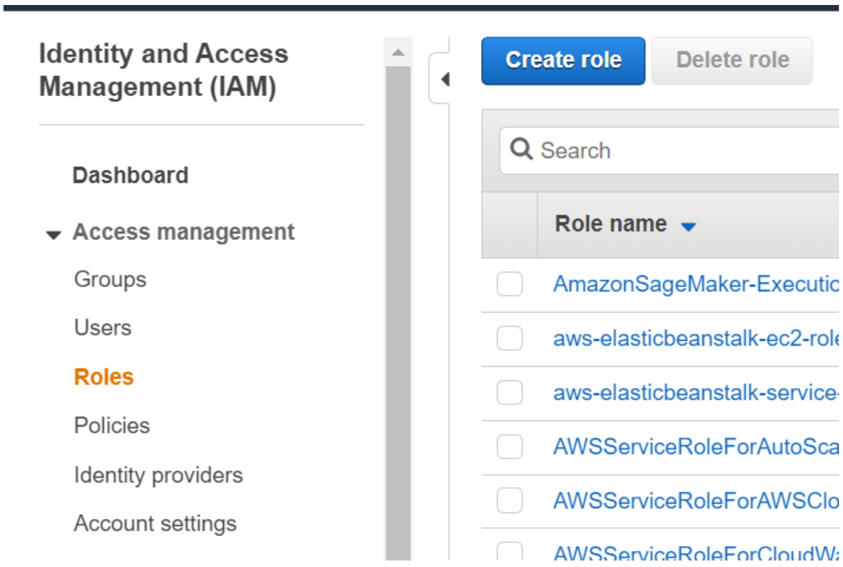
1. Human
2. Person
3. Doctor
4. Veterinarian
5. Animal
6. Pet
7. Mammal
8. Cat

PART B: REKOGNITION + NOTIFICATIONS

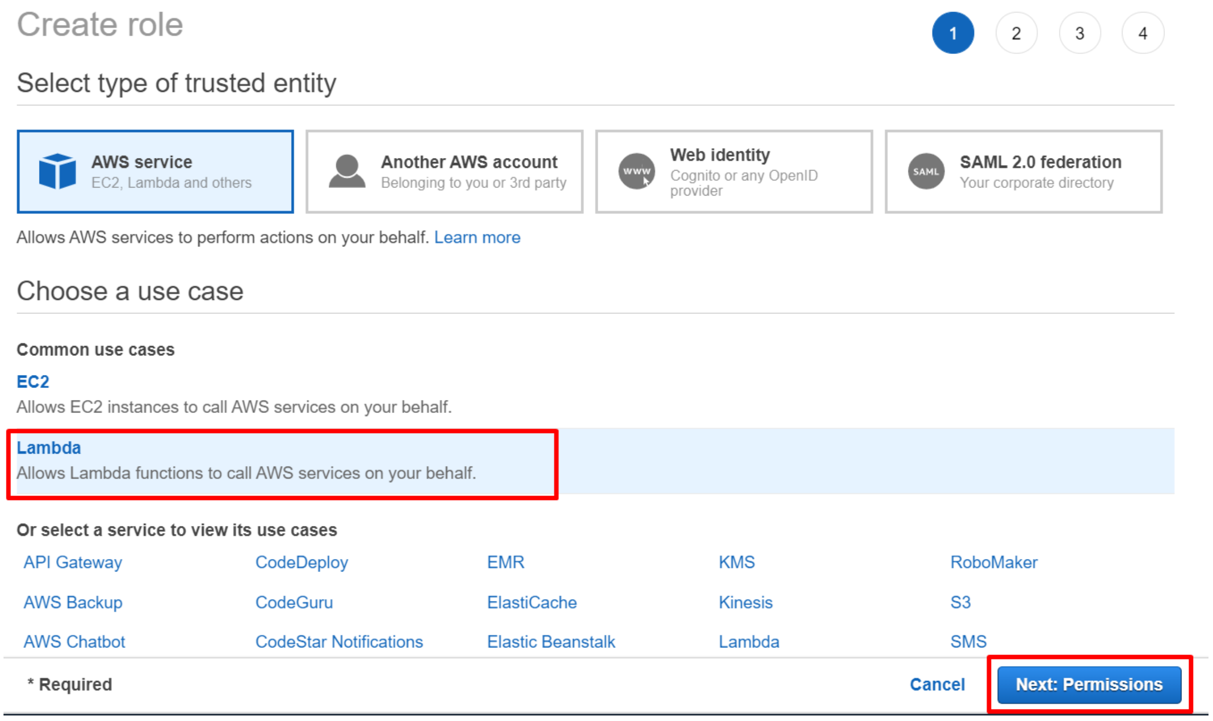
1. Now that we have a basic understanding of Rekognition, let’s try pairing it with other services. We’ll be using AWS Lambda and SNS in conjunction with Rekognition and S3 buckets.

   Let’s create an IAM Role with CloudWatch, Rekognition, and S3 access.

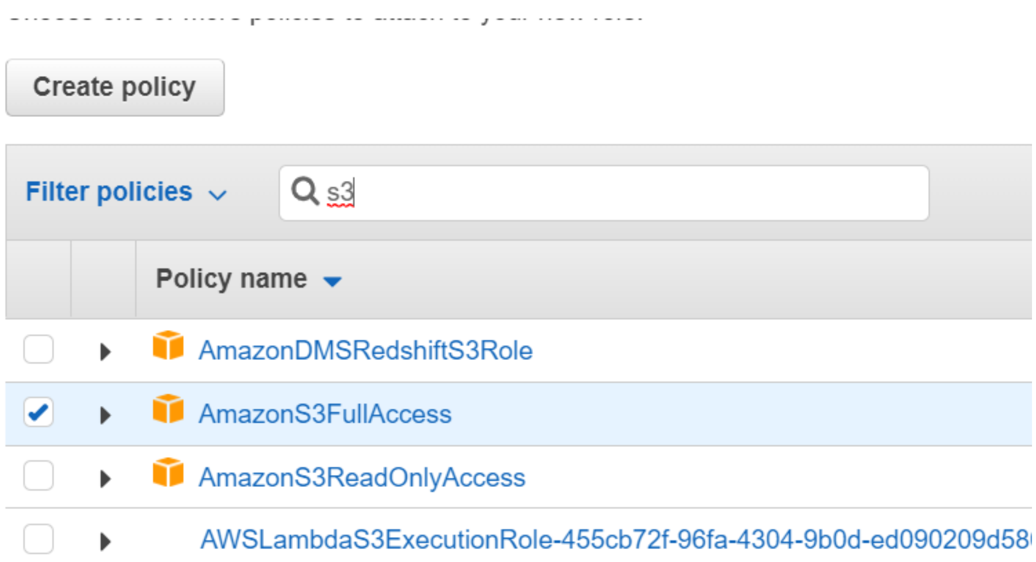
   Go to “IAM”, “Role”, “Create Role”.



2. Pick “Lambda” and click “Next: Permissions”

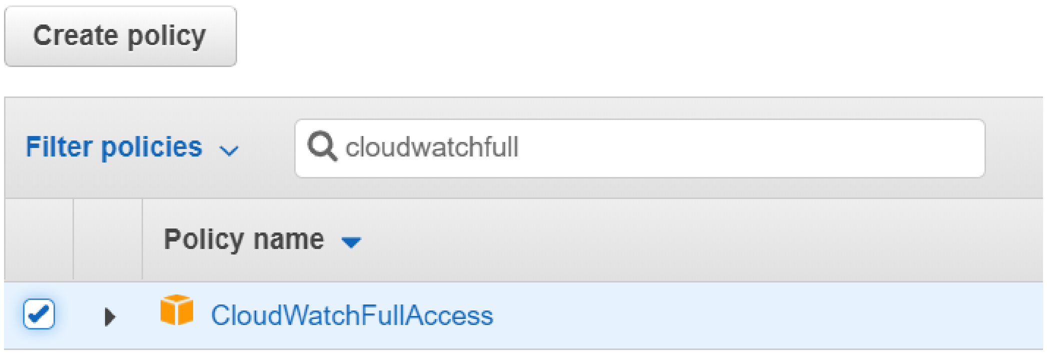


3. Select the following three permissions:



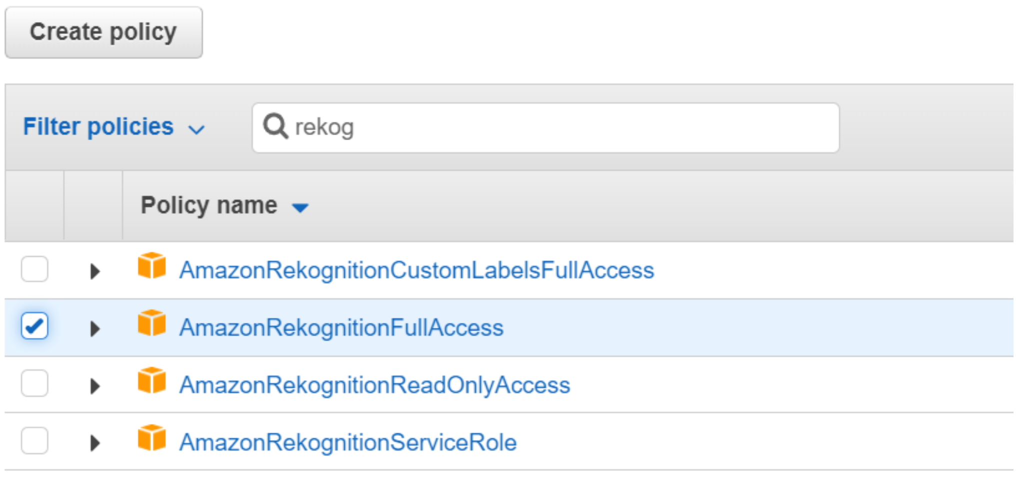
Search: s3

	Policy name
	AmazonDMSRedshiftS3Role
✓	AmazonS3FullAccess
	AmazonS3ReadOnlyAccess
	AWSLambdaS3ExecutionRole-455cb72f-96fa-4304-9b0d-ed090209d58



Search: cloudwatchfull

	Policy name
✓	CloudWatchFullAccess

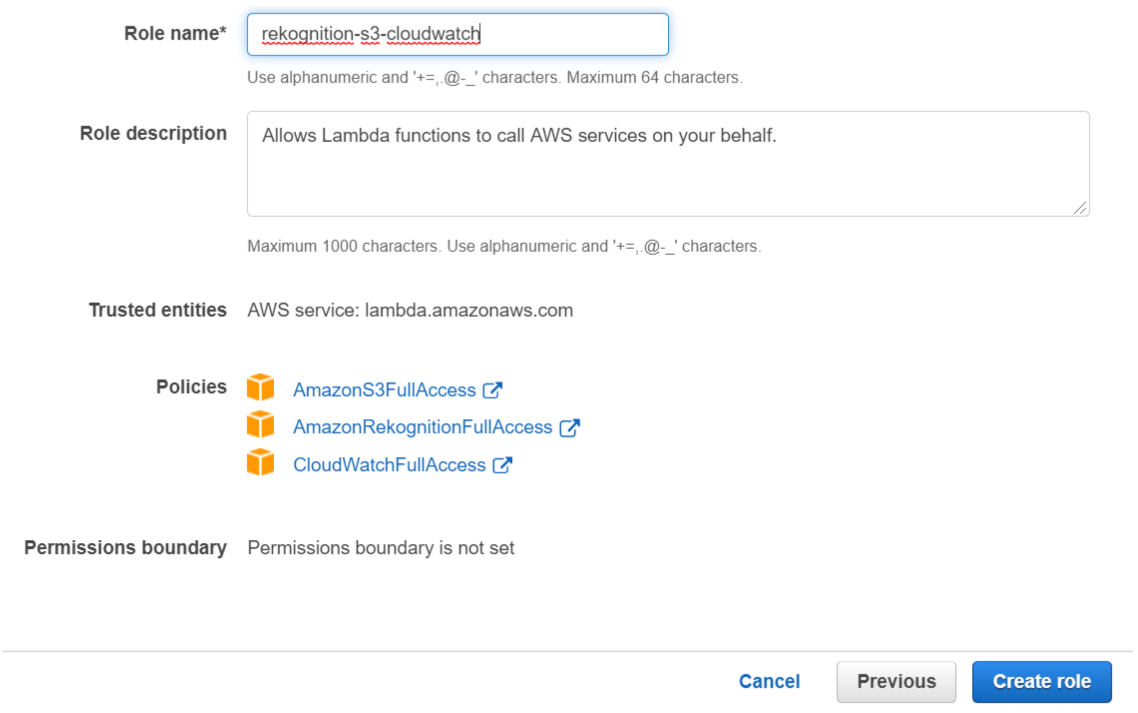


Search: rekog

	Policy name
	AmazonRekognitionCustomLabelsFullAccess
✓	AmazonRekognitionFullAccess
	AmazonRekognitionReadOnlyAccess
	AmazonRekognitionServiceRole

4. Click “Next: Tags” then “Next: Review”.

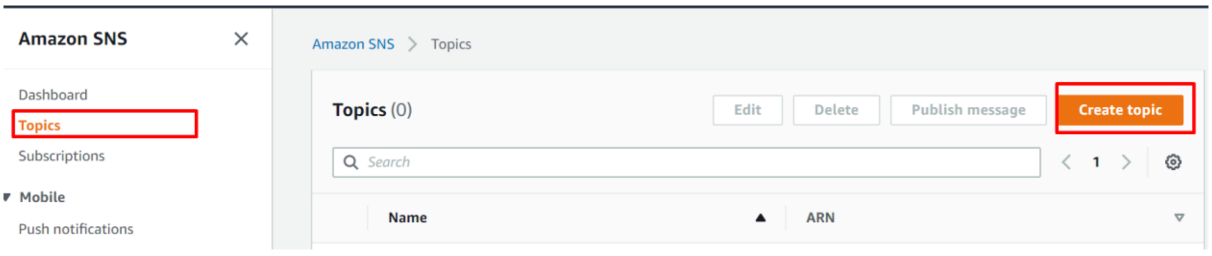
   Name the role something descriptive, like **rekognition-s3-cloudwatch** and click “Create Role”.



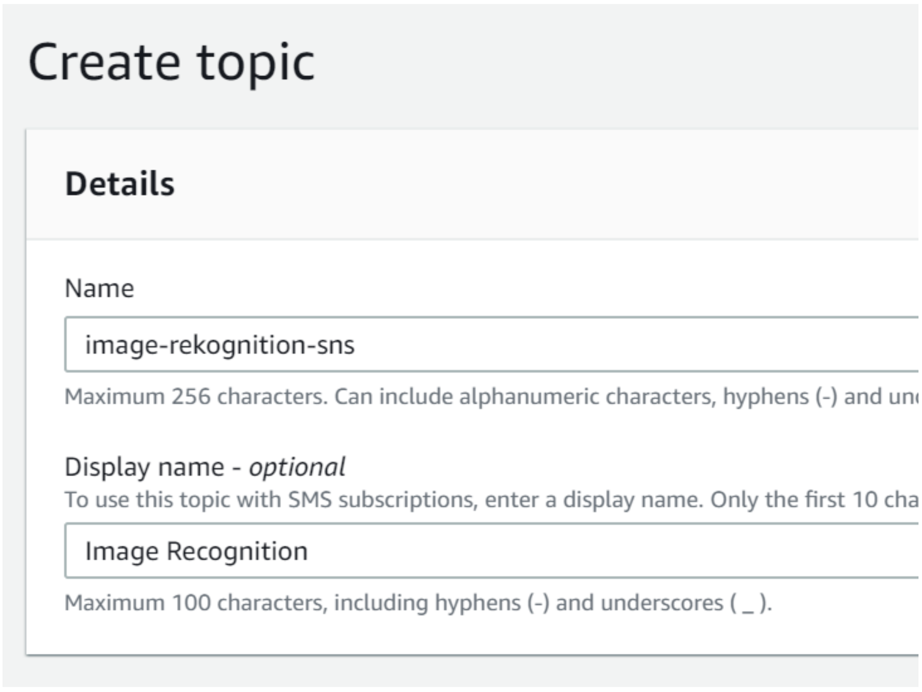
Role name*	rekognition-s3-cloudwatch
	Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.
Role description	Allows Lambda functions to call AWS services on your behalf.
	Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Trusted entities	AWS service: lambda.amazonaws.com
Policies	AmazonS3FullAccess AmazonRekognitionFullAccess CloudWatchFullAccess
Permissions boundary	Permissions boundary is not set

5. Now that we've created an IAM Role that we'll be using for the Lambda function shortly, we need to create an SNS topic for the notification.

   Click "Create Topic".



6. Give your topic an appropriate name (e.g. **image-rekognition-sns**) and display name and click "Create Topic" again.



Create topic

Details

Name

image-rekognition-sns

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and un

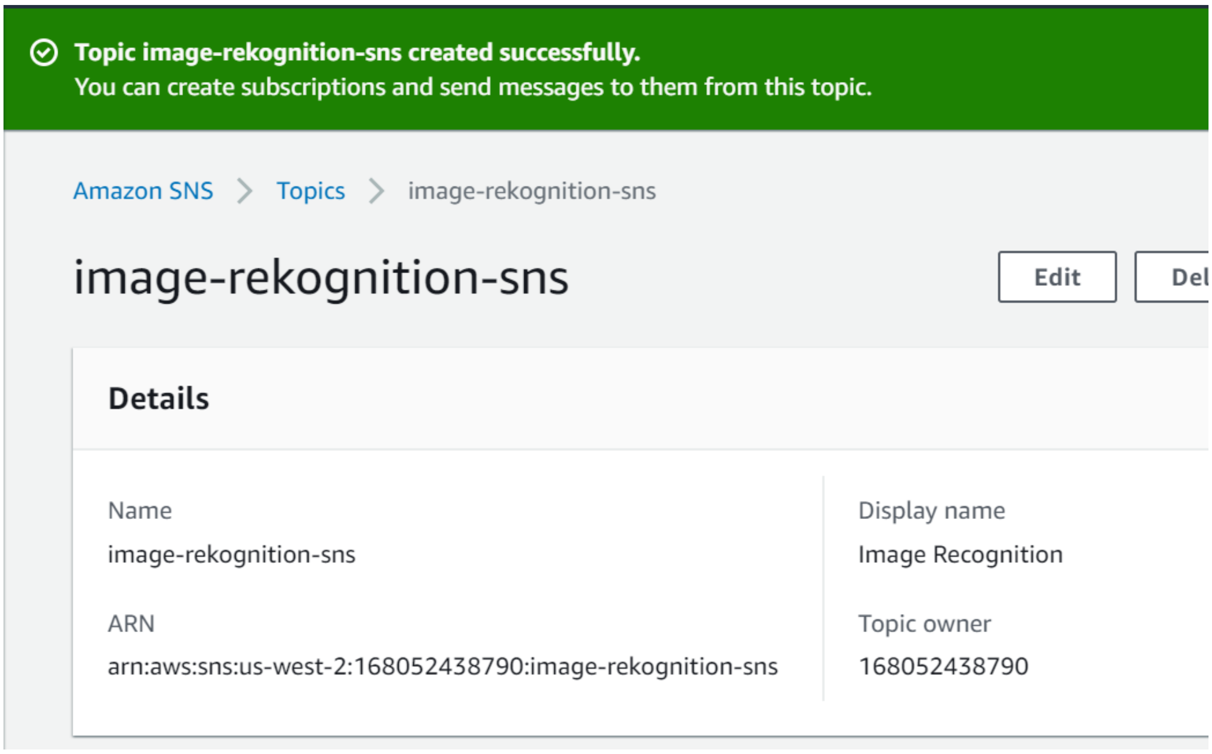
Display name - *optional*

To use this topic with SMS subscriptions, enter a display name. Only the first 10 cha

Image Recognition

Maximum 100 characters, including hyphens (-) and underscores (_).

7. It should redirect you to the SNS topic page where you will see that it's been created successfully.



Topic image-rekognition-sns created successfully.
You can create subscriptions and send messages to them from this topic.

Amazon SNS > Topics > image-rekognition-sns

image-rekognition-sns

Details

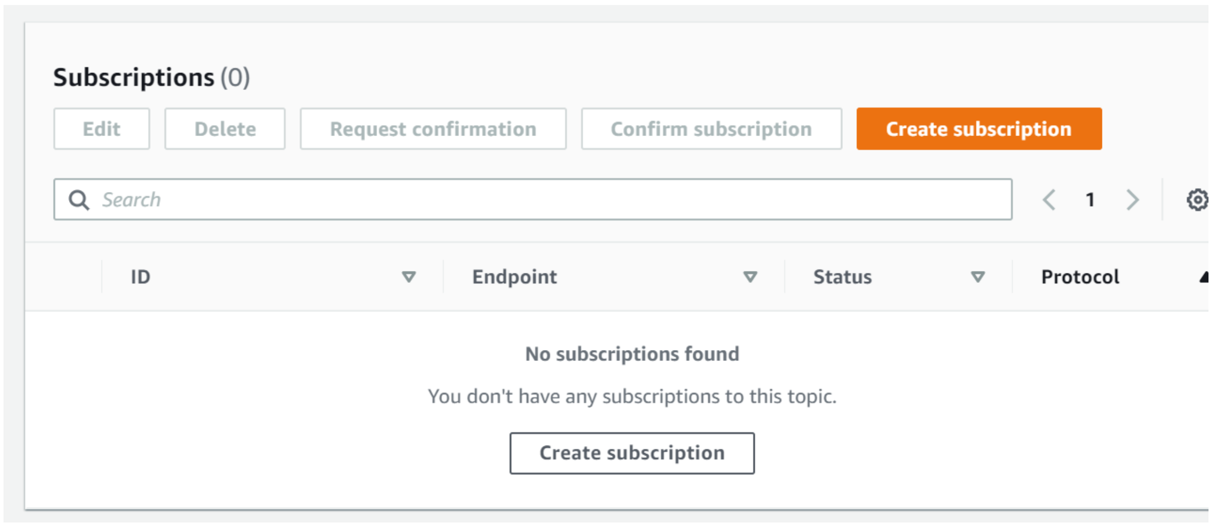
Name
image-rekognition-sns

ARN
arn:aws:sns:us-west-2:168052438790:image-rekognition-sns

Display name
Image Recognition

Topic owner
168052438790

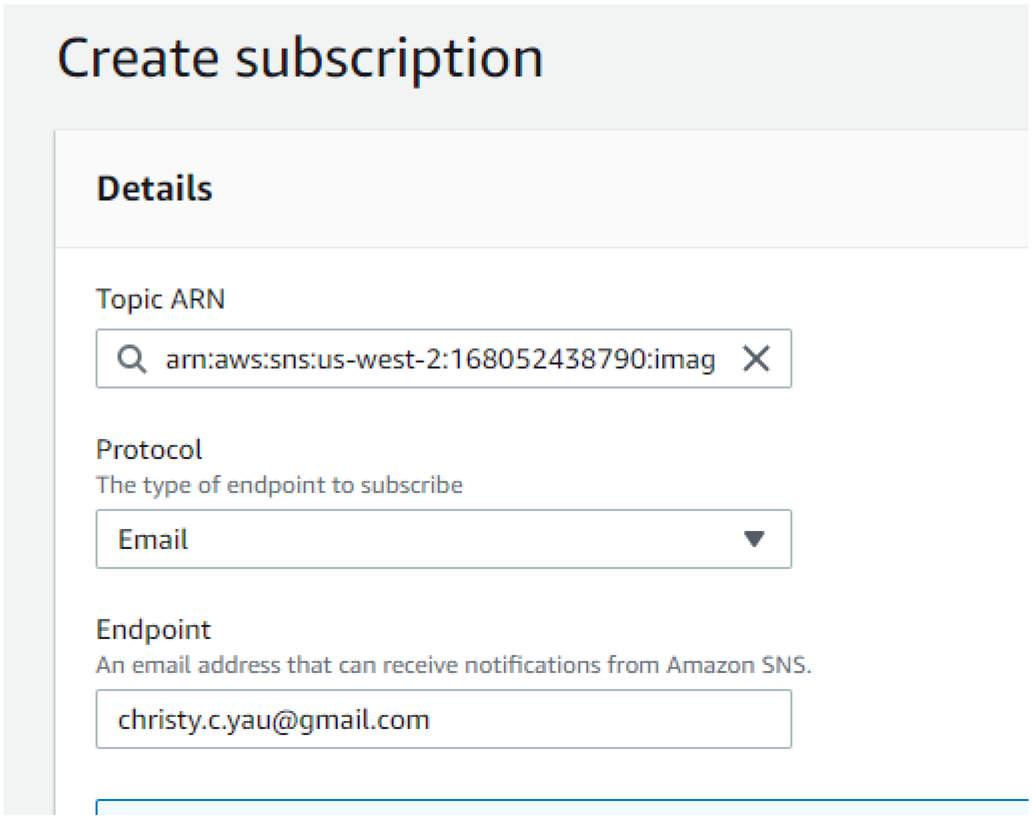
8. Scroll down unti you see "Create Subscription" and click it.



No subscriptions found

You don't have any subscriptions to this topic.

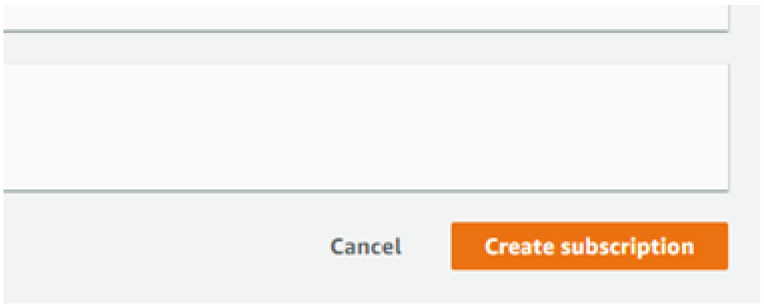
9. Enter “Email” for Protocol and type in your e-mail address for Endpoint. Click Create Subscription.



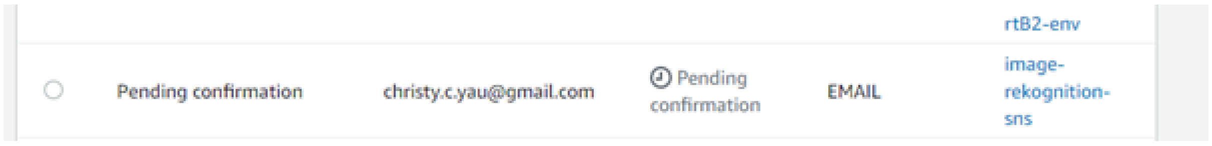
Topic ARN: arn:aws:sns:us-west-2:168052438790:imag

Protocol
The type of endpoint to subscribe
Email

Endpoint
An email address that can receive notifications from Amazon SNS.
christy.c.yau@gmail.com



10. If you click Subscriptions on the left, you will see your subscription is Pending Confirmation:



					rtB2-env
○	Pending confirmation	christy.c.yau@gmail.com	Pending confirmation	EMAIL	image-rekognition-sns

Check your inbox for the subscription confirmation and confirm your subscription.

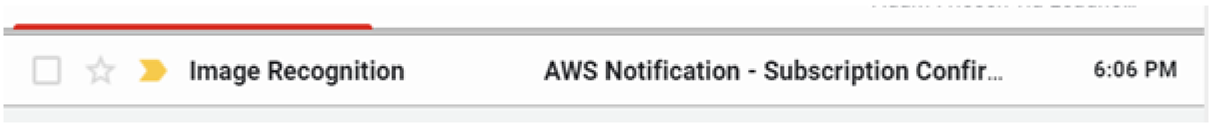
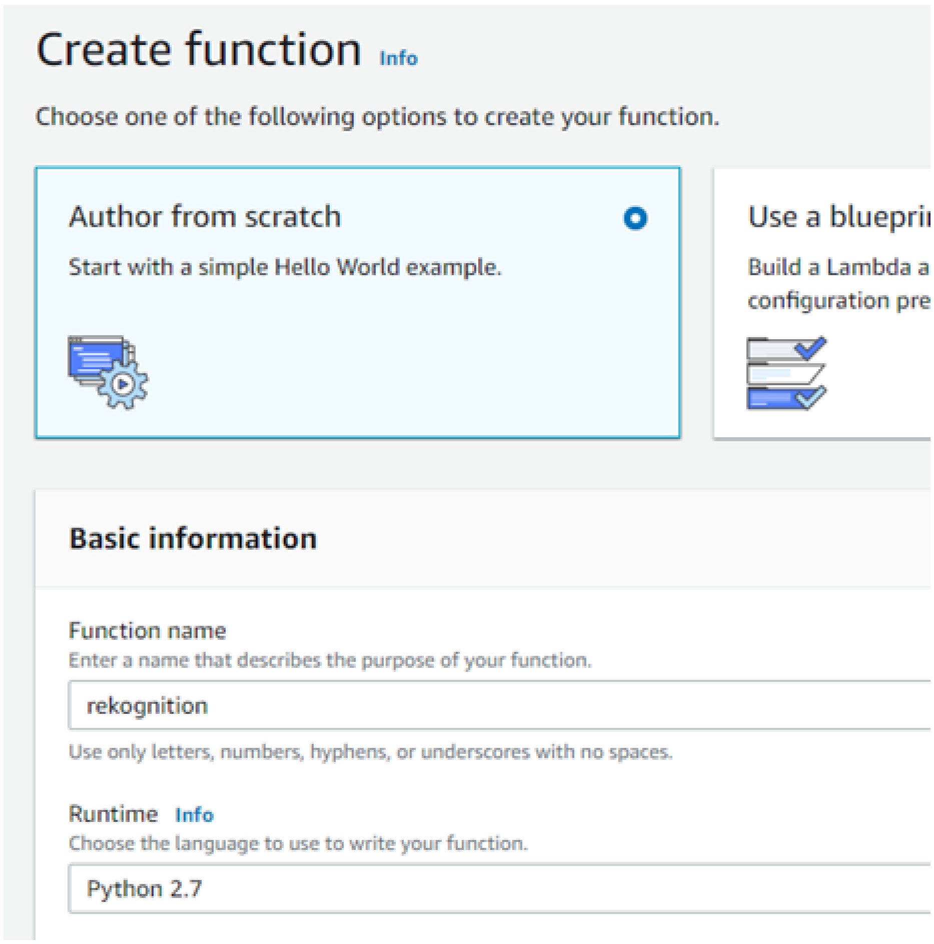


Image Recognition AWS Notification - Subscription Confir... 6:06 PM

11. Because we already set up the S3 bucket in the previous step, we can continue using the same one.

12. Let's now create the Lambda function.

    Pick author from scratch and Python 2.7. Use the IAM role that we just created (rekognition-s3-cloudwatch).


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 bluepri

Build a Lambda a
configuration pre

Basic information

Function name
Enter a name that describes the purpose of your function.

rekognition

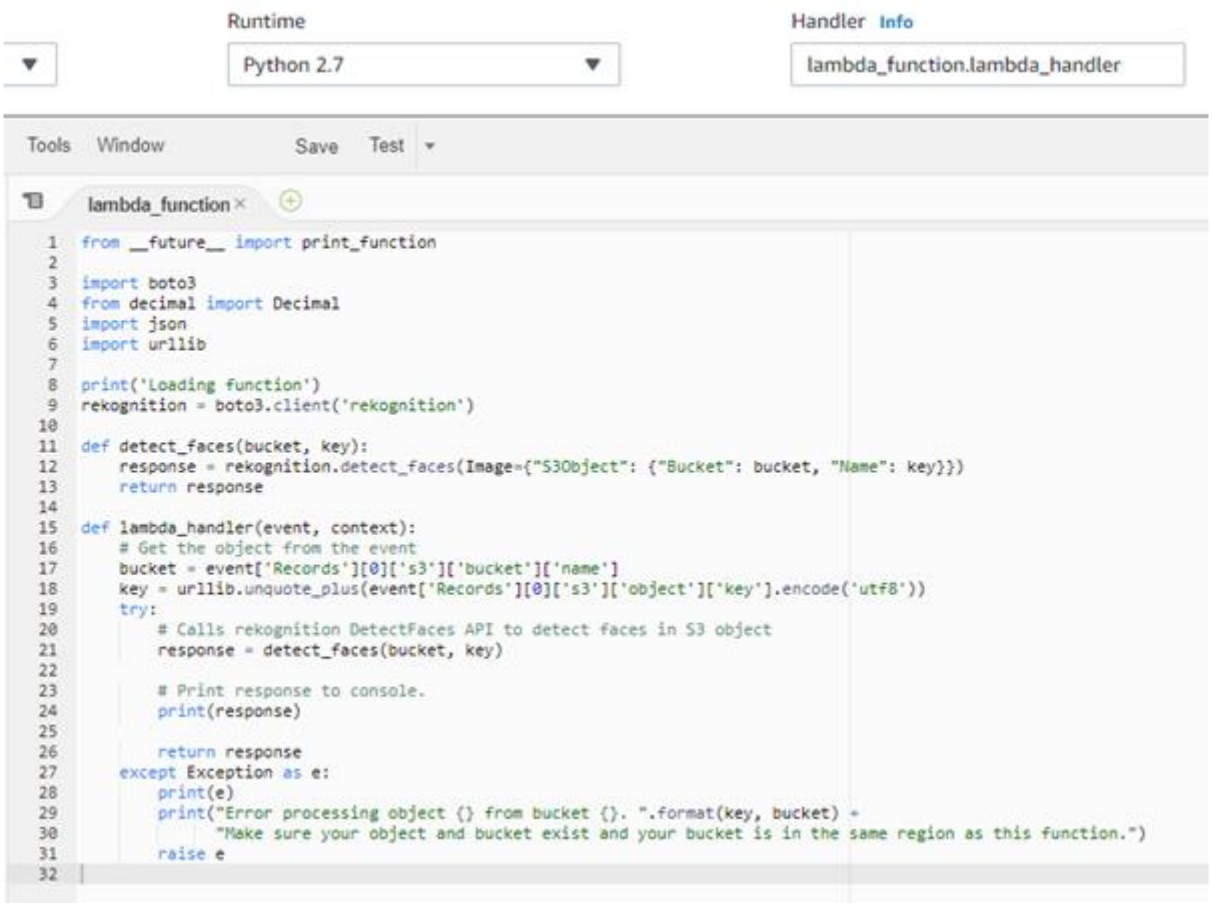
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.

Python 2.7

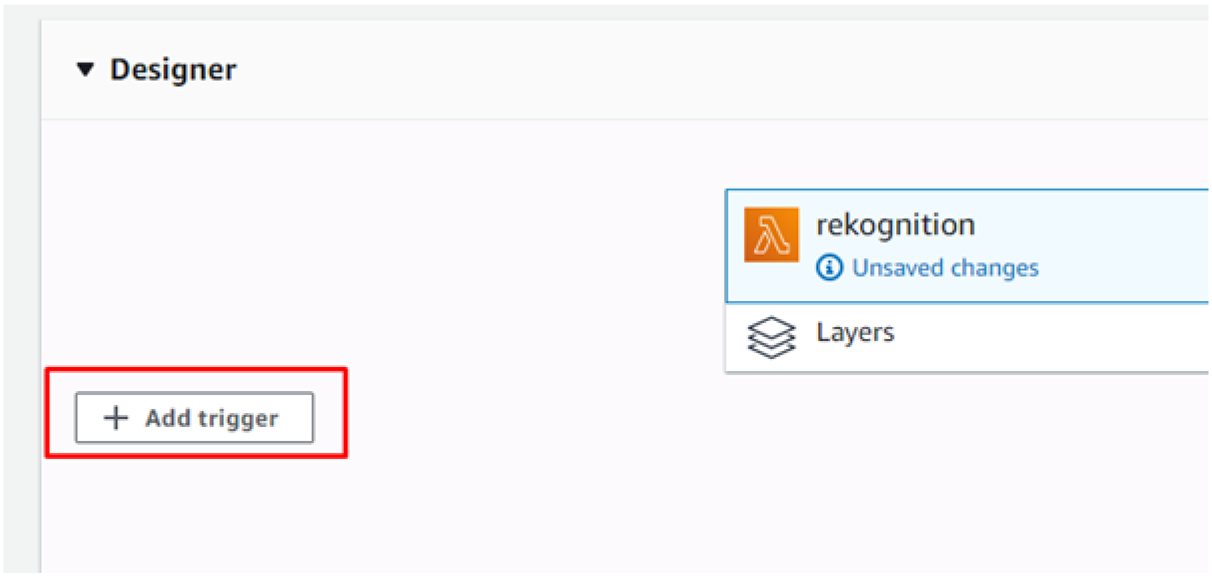
13. Copy and paste the code from part02.py. Look at the code. What does it do? It uses the **DetectFaces** feature to track and analyse the faces in photos.

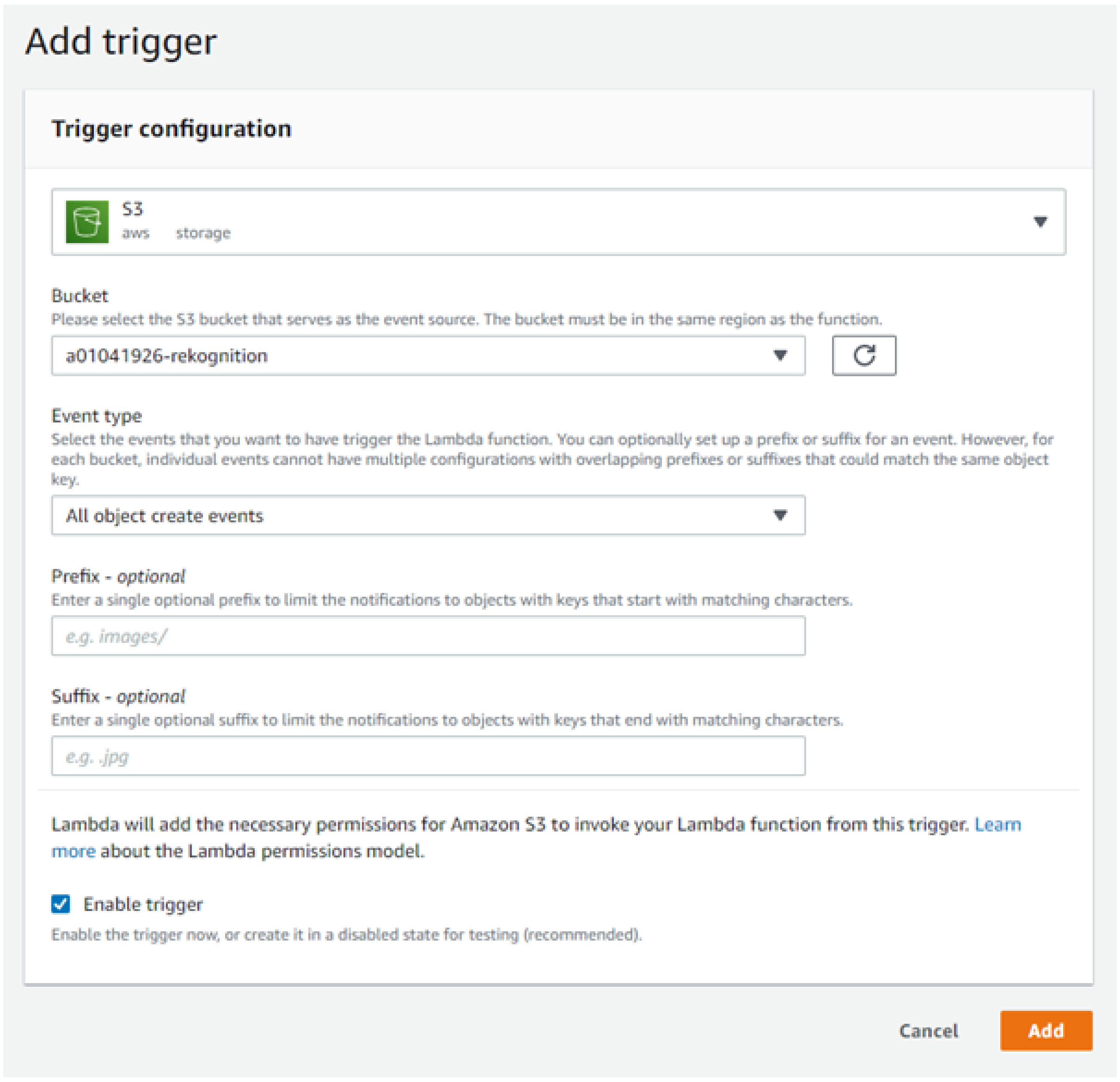
    Make sure you save the lambda function.



```python
from __future__ import print_function

import boto3
from decimal import Decimal
import json
import urllib

print('Loading function')
rekognition = boto3.client('rekognition')

def detect_faces(bucket, key):
    response = rekognition.detect_faces(Image={"S3Object": {"Bucket": bucket, "Name": key}})
    return response

def lambda_handler(event, context):
    # Get the object from the event
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.unquote_plus(event['Records'][0]['s3']['object']['key'].encode('utf8'))
    try:
        # Calls rekognition DetectFaces API to detect faces in S3 object
        response = detect_faces(bucket, key)

        # Print response to console.
        print(response)

        return response
    except Exception as e:
        print(e)
        print("Error processing object {} from bucket {}. ".format(key, bucket) +
              "Make sure your object and bucket exist and your bucket is in the same region as this function.")
        raise e

```

14. We're almost ready to test it out. Let's add an S3 trigger to our lambda function. This will cause the lambda function to run whenever a new object is created in the S3 bucket.





Add trigger

Trigger configuration

S3
aws storage

Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
a01041926-rekognition

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events

Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/

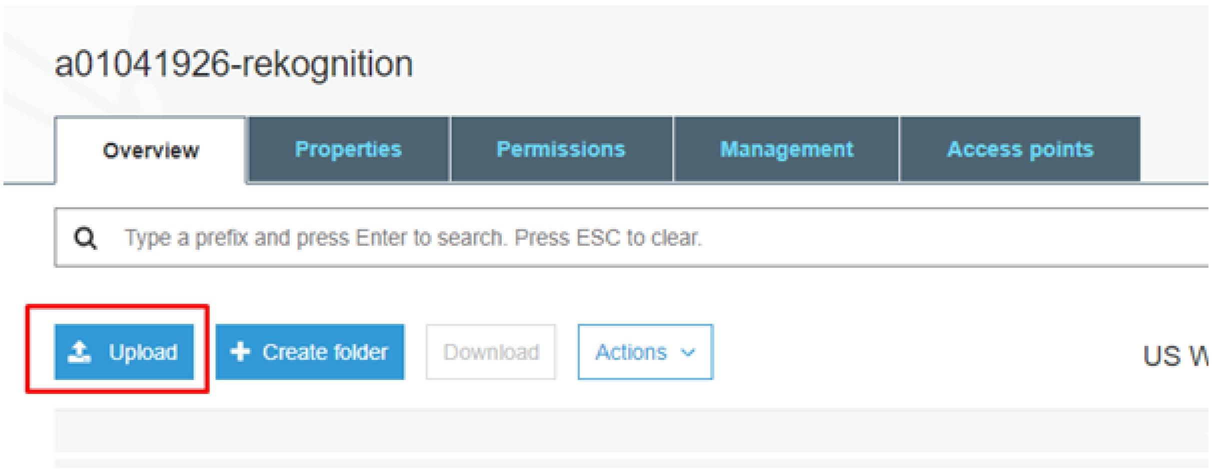
Suffix - *optional*
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

☑ Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

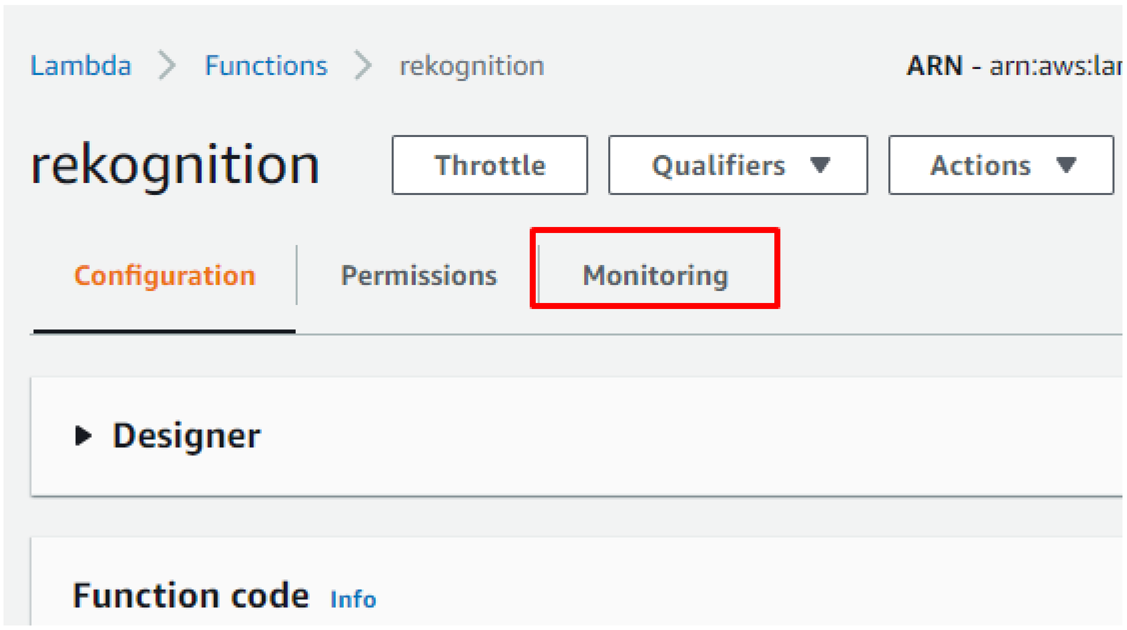
Cancel Add

15. Now you're ready to test it out. Go to your S3 bucket that you selected for the S3 Trigger in the lambda function and upload **people01.jpg**

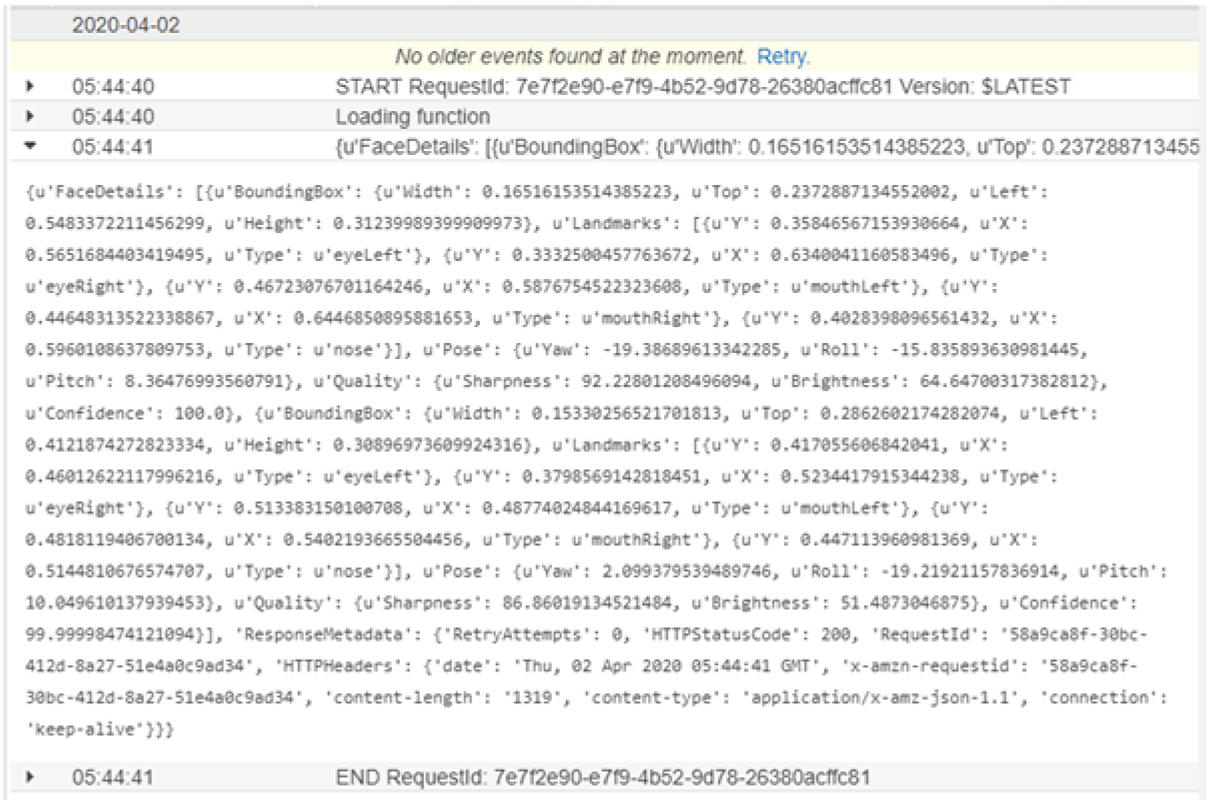


a01041926-rekognition

16. Once uploaded, the lambda function should trigger as it's set to run on "All object create events" in the S3 bucket. Go to your lambda function's Monitoring tab and click "View logs in CloudWatch".



17. Look at the logs to see the following output:

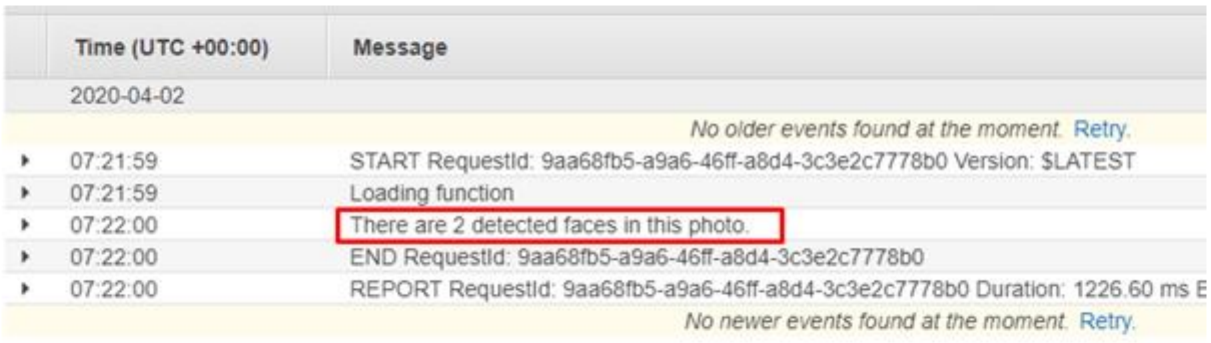


```
    2020-04-02
                      No older events found at the moment. Retry.
▸   05:44:40          START RequestId: 7e7f2e90-e7f9-4b52-9d78-26380acffc81 Version: $LATEST
▸   05:44:40          Loading function
▾   05:44:41          {u'FaceDetails': [{u'BoundingBox': {u'Width': 0.16516153514385223, u'Top': 0.237288713455

{u'FaceDetails': [{u'BoundingBox': {u'Width': 0.16516153514385223, u'Top': 0.2372887134552002, u'Left':
0.5483372211456299, u'Height': 0.31239989399909973}, u'Landmarks': [{u'Y': 0.35846567153930664, u'X':
0.5651684403419495, u'Type': u'eyeLeft'}, {u'Y': 0.3332500457763672, u'X': 0.6340041160583496, u'Type':
u'eyeRight'}, {u'Y': 0.46723076701164246, u'X': 0.5876754522323608, u'Type': u'mouthLeft'}, {u'Y':
0.44648313522338867, u'X': 0.6446850895881653, u'Type': u'mouthRight'}, {u'Y': 0.4028398096561432, u'X':
0.5960108637809753, u'Type': u'nose'}], u'Pose': {u'Yaw': -19.38689613342285, u'Roll': -15.835893630981445,
u'Pitch': 8.36476993560791}, u'Quality': {u'Sharpness': 92.22801208496094, u'Brightness': 64.64700317382812},
u'Confidence': 100.0}, {u'BoundingBox': {u'Width': 0.15330256521701813, u'Top': 0.2862602174282074, u'Left':
0.4121874272823334, u'Height': 0.30896973609924316}, u'Landmarks': [{u'Y': 0.417055606842041, u'X':
0.46012622117996216, u'Type': u'eyeLeft'}, {u'Y': 0.3798569142818451, u'X': 0.5234417915344238, u'Type':
u'eyeRight'}, {u'Y': 0.513383150100708, u'X': 0.48774024844169617, u'Type': u'mouthLeft'}, {u'Y':
0.4818119406700134, u'X': 0.5402193665504456, u'Type': u'mouthRight'}, {u'Y': 0.447113960981369, u'X':
0.5144810676574707, u'Type': u'nose'}], u'Pose': {u'Yaw': 2.099379539489746, u'Roll': -19.21921157836914, u'Pitch':
10.049610137939453}, u'Quality': {u'Sharpness': 86.86019134521484, u'Brightness': 51.4873046875}, u'Confidence':
99.99998474121094}], 'ResponseMetadata': {'RetryAttempts': 0, 'HTTPStatusCode': 200, 'RequestId': '58a9ca8f-30bc-
412d-8a27-51e4a0c9ad34', 'HTTPHeaders': {'date': 'Thu, 02 Apr 2020 05:44:41 GMT', 'x-amzn-requestid': '58a9ca8f-
30bc-412d-8a27-51e4a0c9ad34', 'content-length': '1319', 'content-type': 'application/x-amz-json-1.1', 'connection':
'keep-alive'}}}

▸   05:44:41          END RequestId: 7e7f2e90-e7f9-4b52-9d78-26380acffc81
```

How can we take this information to count the number of faces in a photo?

Try to create the same output as below:

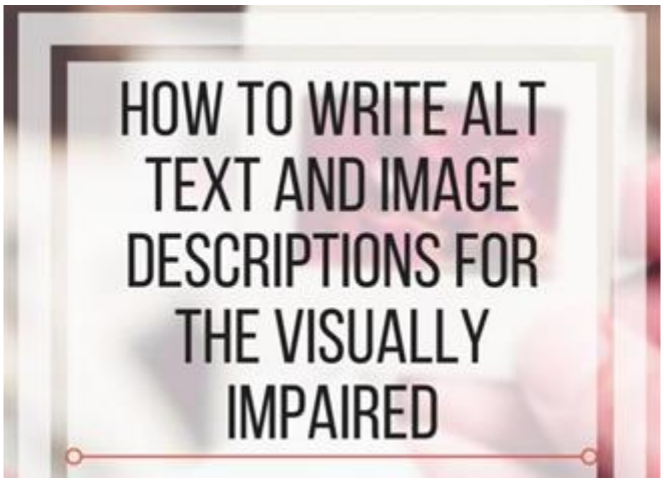
people01.jpg

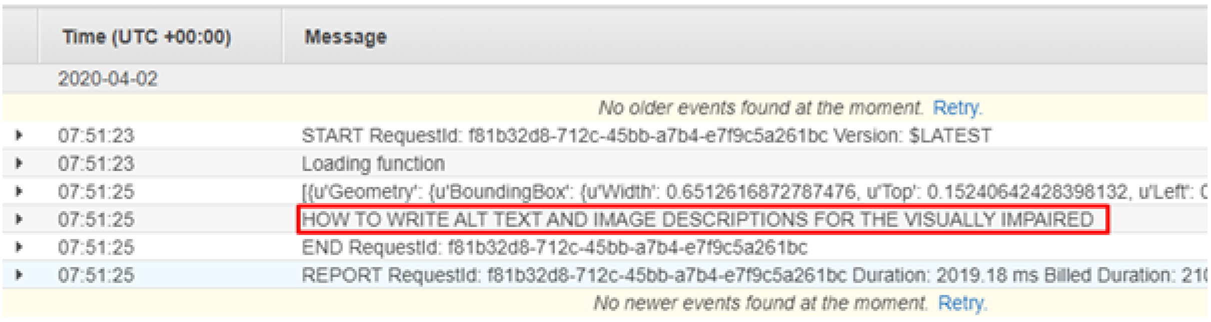


```
     Time (UTC +00:00)    Message
     2020-04-02
                                        No older events found at the moment. Retry.
▸    07:21:59             START RequestId: 9aa68fb5-a9a6-46ff-a8d4-3c3e2c7778b0 Version: $LATEST
▸    07:21:59             Loading function
▸    07:22:00             There are 2 detected faces in this photo.
▸    07:22:00             END RequestId: 9aa68fb5-a9a6-46ff-a8d4-3c3e2c7778b0
▸    07:22:00             REPORT RequestId: 9aa68fb5-a9a6-46ff-a8d4-3c3e2c7778b0 Duration: 1226.60 ms E
                                        No newer events found at the moment. Retry.
```

18. Now let’s explore other features of AWS Rekognition. How can you use one of its other features to detect text?
HINT: Check the AWS Rekognition Developer Guide’s ‘**Detecting Text’** section. https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html

img01.jpg





```
Time (UTC +00:00)    Message
2020-04-02
                     No older events found at the moment. Retry.
07:51:23             START RequestId: f81b32d8-712c-45bb-a7b4-e7f9c5a261bc Version: $LATEST
07:51:23             Loading function
07:51:25             [{u'Geometry': {u'BoundingBox': {u'Width': 0.6512616872787476, u'Top': 0.15240642428398132, u'Left': 0
07:51:25             HOW TO WRITE ALT TEXT AND IMAGE DESCRIPTIONS FOR THE VISUALLY IMPAIRED
07:51:25             END RequestId: f81b32d8-712c-45bb-a7b4-e7f9c5a261bc
07:51:25             REPORT RequestId: f81b32d8-712c-45bb-a7b4-e7f9c5a261bc Duration: 2019.18 ms Billed Duration: 21
                     No newer events found at the moment. Retry.
```

19. Once you can print out the text to CloudWatch, send an e-mail to yourself with the detected text through the SNS topic we created.

> HINTS
>
> SNS documentation for Boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sns.html
>
> You will also need the **SNS Topic ARN** and to use the **publish()** method for SNS.

If successful, it should look like this:

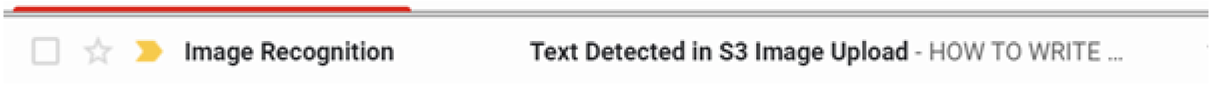
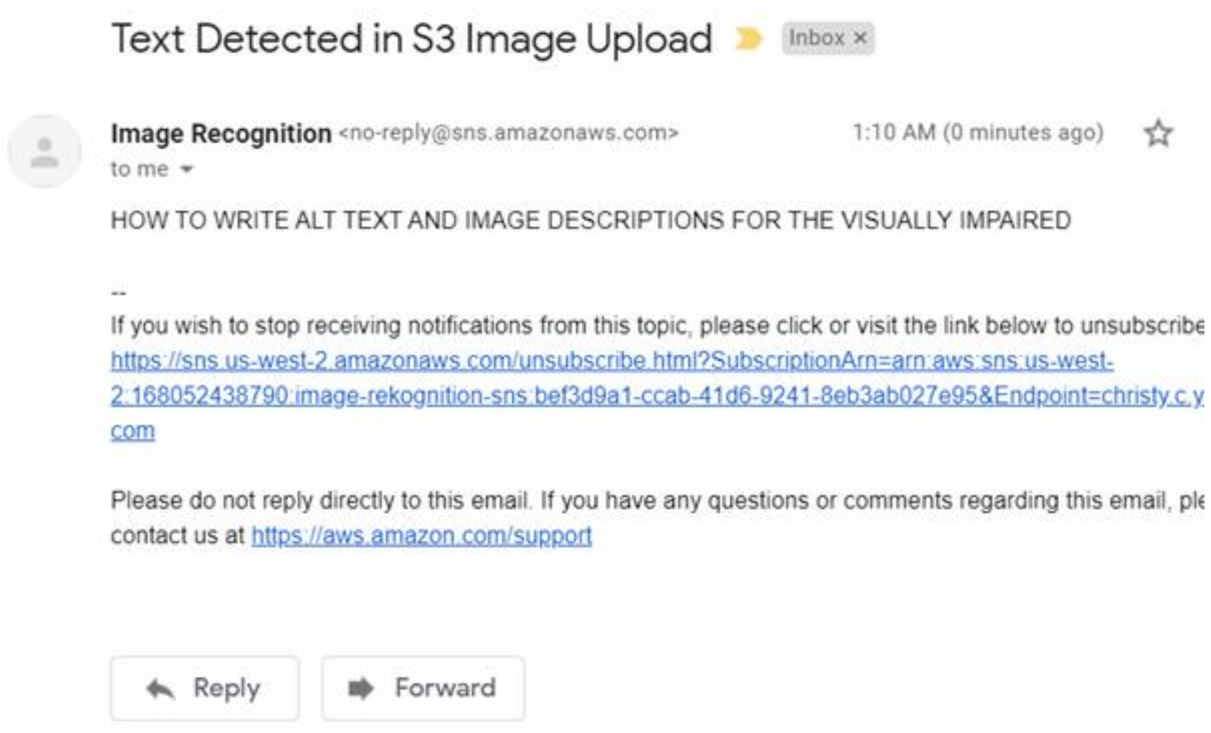


Image Recognition   **Text Detected in S3 Image Upload** - HOW TO WRITE ...



Text Detected in S3 Image Upload Inbox ×

Image Recognition <no-reply@sns.amazonaws.com> 1:10 AM (0 minutes ago)
to me

HOW TO WRITE ALT TEXT AND IMAGE DESCRIPTIONS FOR THE VISUALLY IMPAIRED

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe
https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:168052438790:image-rekognition-sns:bef3d9a1-ccab-41d6-9241-8eb3ab027e95&Endpoint=christy.c.y com

Please do not reply directly to this email. If you have any questions or comments regarding this email, ple contact us at https://aws.amazon.com/support

20. Remember to disable your lambda function before proceeding to part03, or you will end up receiving e-mails with each upload.

PART C: TWITTER + REKOGNITION

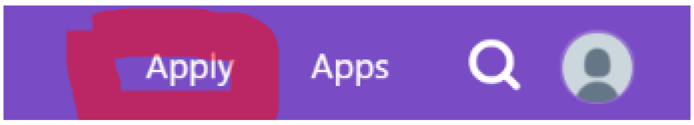
In the next part of the lab, we’ll use AWS Rekognition’s facial recognition ability to analyse celebrities’ faces and tweet at their handles.

To start, we’ll need to have a Twitter account.

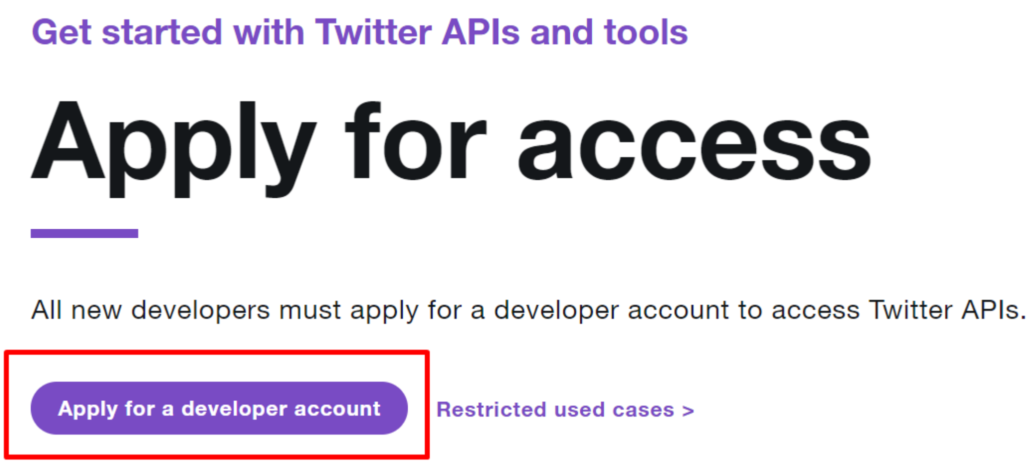
Part 1: Setting up a Developer account for Twitter

1. Go to https://developer.twitter.com/ and sign in.

2. Click on “Apply” in the top right corner.



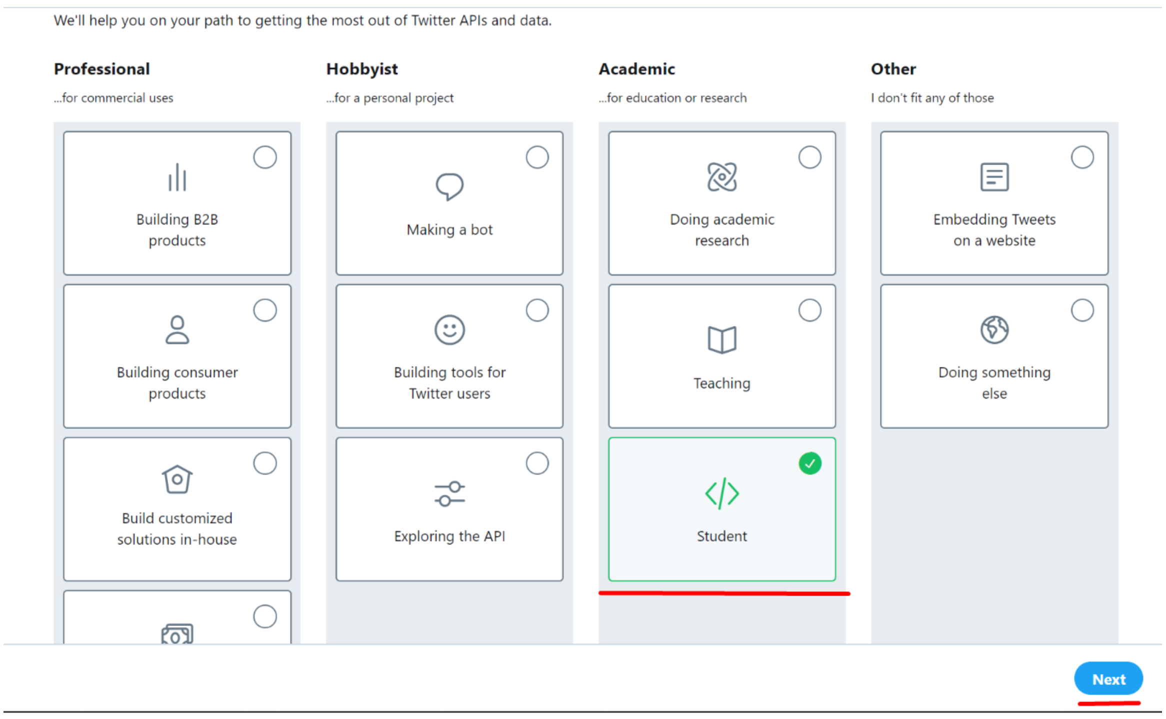
3. Click “Apply for a developer account”


Get started with Twitter APIs and tools

Apply for access

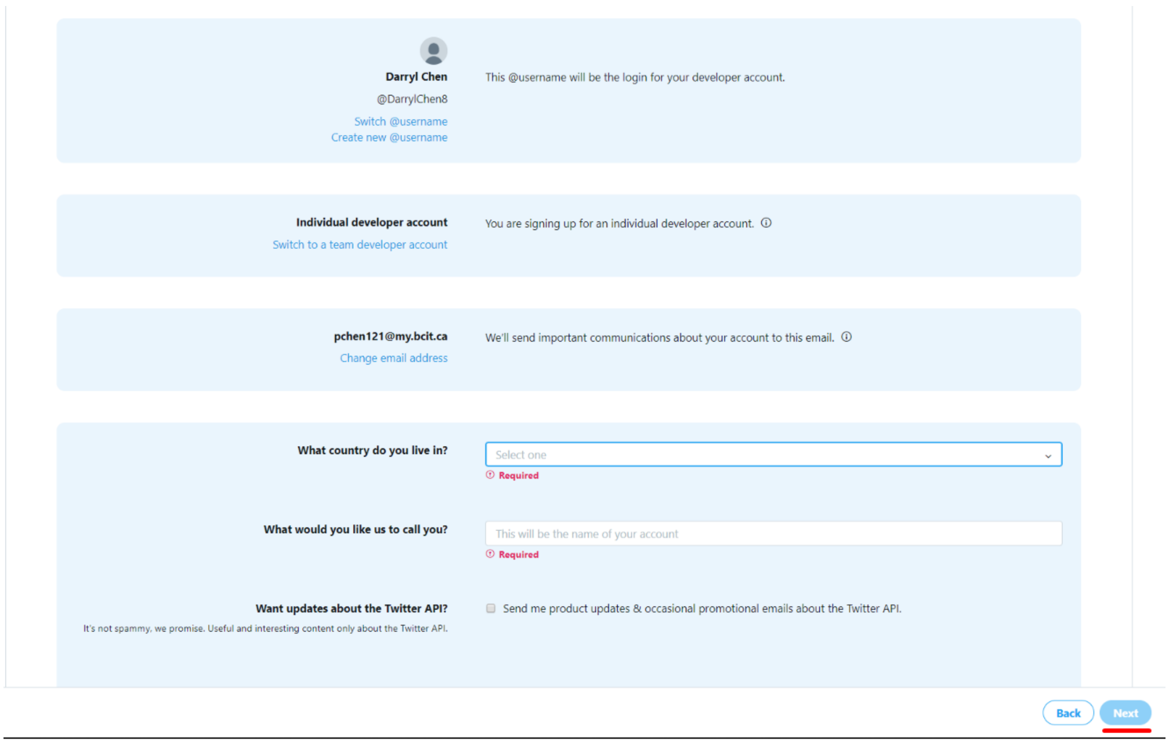
All new developers must apply for a developer account to access Twitter APIs.

4. Select “Student” under the Academic category and click “Next”.


We'll help you on your path to getting the most out of Twitter APIs and data.

Professional	Hobbyist	Academic	Other
...for commercial uses	...for a personal project	...for education or research	I don't fit any of those
Building B2B products	Making a bot	Doing academic research	Embedding Tweets on a website
Building consumer products	Building tools for Twitter users	Teaching	Doing something else
Build customized solutions in-house	Exploring the API	✓ Student	

5. Fill out the mandatory / required fields and click “Next”.



Darryl Chen
@DarrylChen8
Switch @username
Create new @username

This @username will be the login for your developer account.

Individual developer account
Switch to a team developer account

You are signing up for an individual developer account.

pchen121@my.bcit.ca
Change email address

We'll send important communications about your account to this email.

What country do you live in? Select one
Required

What would you like us to call you? This will be the name of your account
Required

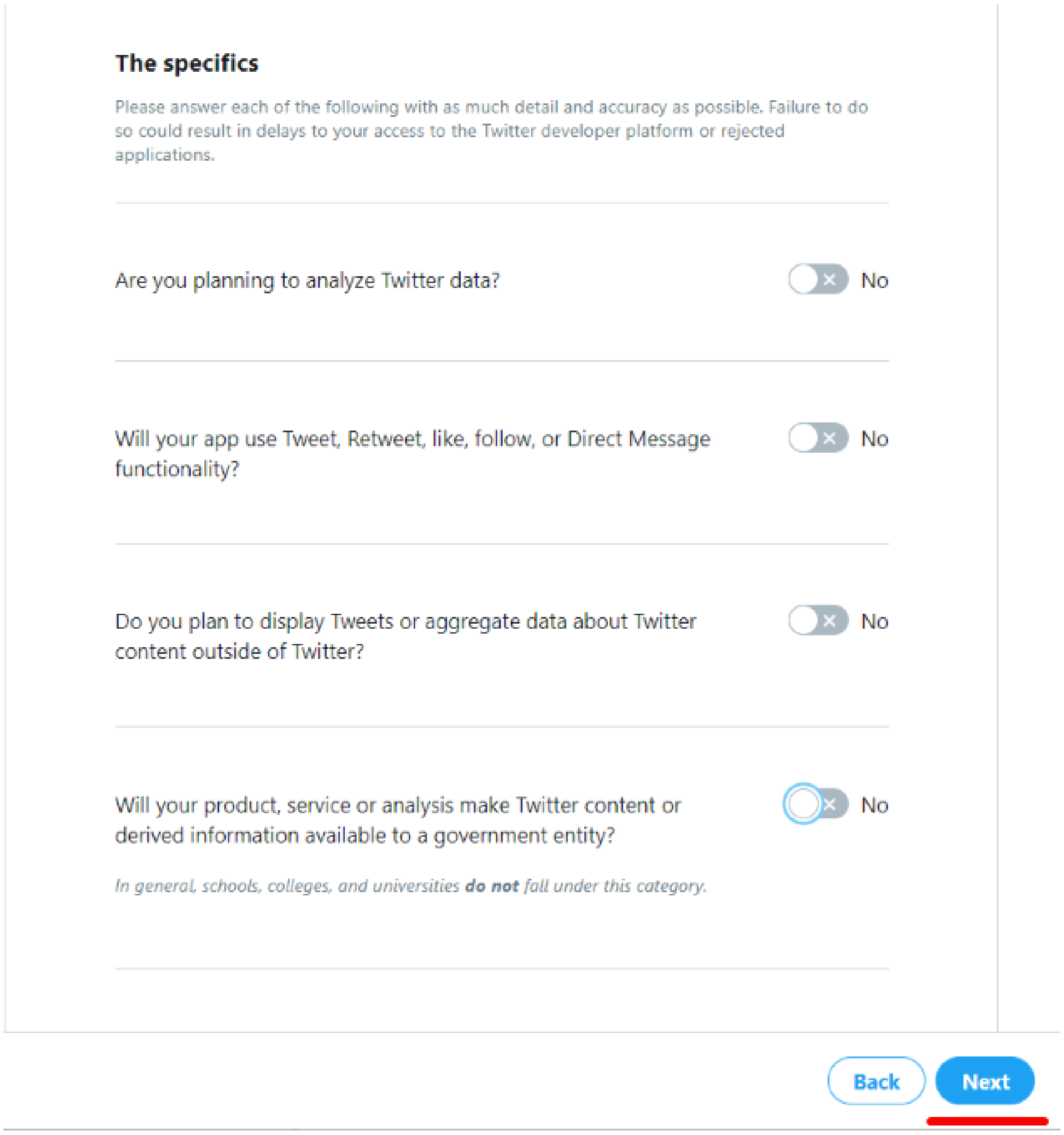
Want updates about the Twitter API?
It's not spammy, we promise. Useful and interesting content only about the Twitter API.

Send me product updates & occasional promotional emails about the Twitter API.

6. On the next screen, In the area where it says in “In your words”, copy and paste the following:

   “I am a computer science student working on a cloud computing project that integrates Twitter and AWS serverless microservices. I will be primarily using the Twitter API with AWS Rekognition, an image and video analysis service. After Rekognition performs an analysis, I will use the Twitter API to tweet from my developer account.“

7. In “The Specifics”, make sure everything is set to “No” and click “Next”.



The specifics

Please answer each of the following with as much detail and accuracy as possible. Failure to do so could result in delays to your access to the Twitter developer platform or rejected applications.

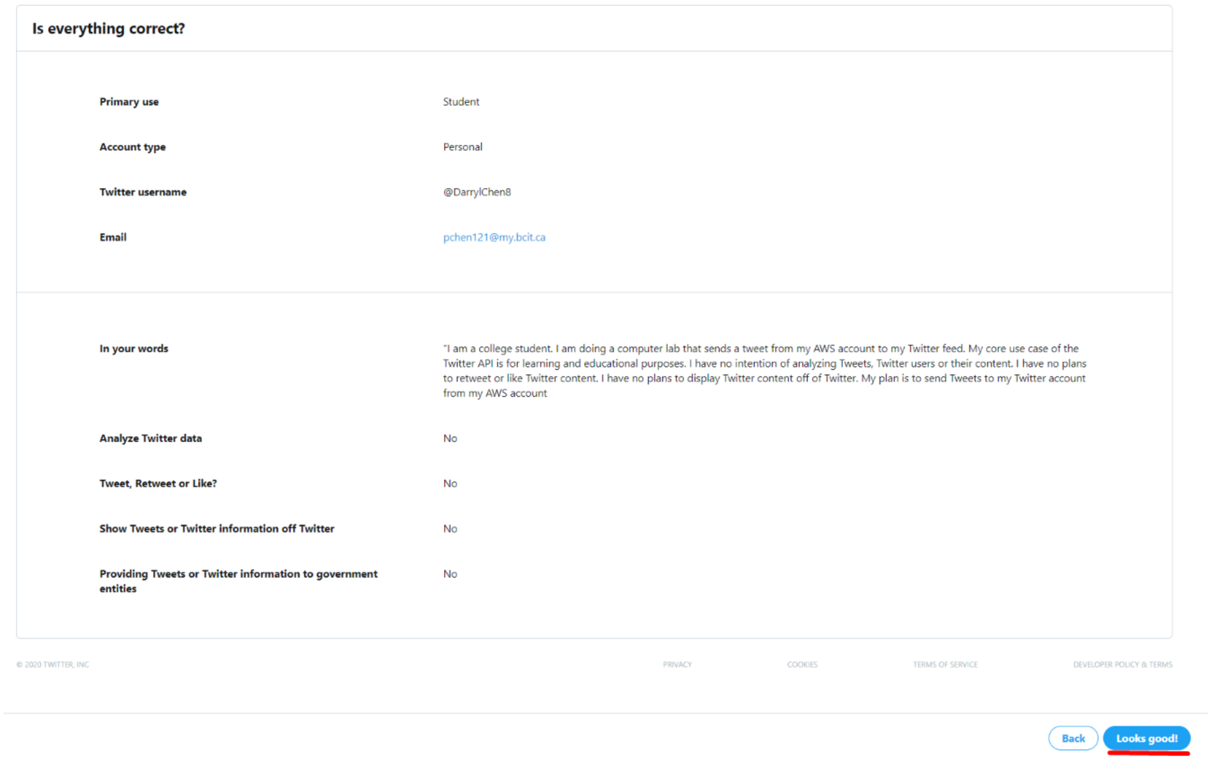
Are you planning to analyze Twitter data? No

Will your app use Tweet, Retweet, like, follow, or Direct Message functionality? No

Do you plan to display Tweets or aggregate data about Twitter content outside of Twitter? No

Will your product, service or analysis make Twitter content or derived information available to a government entity? No

*In general, schools, colleges, and universities **do not** fall under this category.*



Is everything correct?

Primary use	Student
Account type	Personal
Twitter username	@DarrylChen8
Email	pchen121@my.bcit.ca
In your words	"I am a college student. I am doing a computer lab that sends a tweet from my AWS account to my Twitter feed. My core use case of the Twitter API is for learning and educational purposes. I have no intention of analyzing Tweets, Twitter users or their content. I have no plans to retweet or like Twitter content. I have no plans to display Twitter content off of Twitter. My plan is to send Tweets to my Twitter account from my AWS account
Analyze Twitter data	No
Tweet, Retweet or Like?	No
Show Tweets or Twitter information off Twitter	No
Providing Tweets or Twitter information to government entities	No

© 2020 TWITTER, INC PRIVACY COOKIES TERMS OF SERVICE DEVELOPER POLICY & TERMS

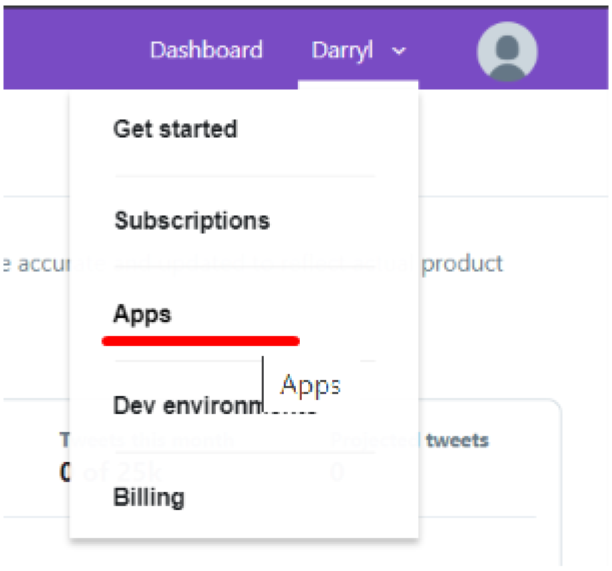
8. Double-check that everything is correct and click "Looks good!"

Agree to the Terms of Service and you should receive an e-mail from Twitter about your application within 24 hours.

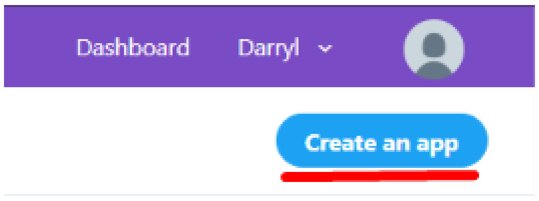
Part 2: Creating the App

1. Login to your Twitter Developer Account (https://developer.twitter.com/)

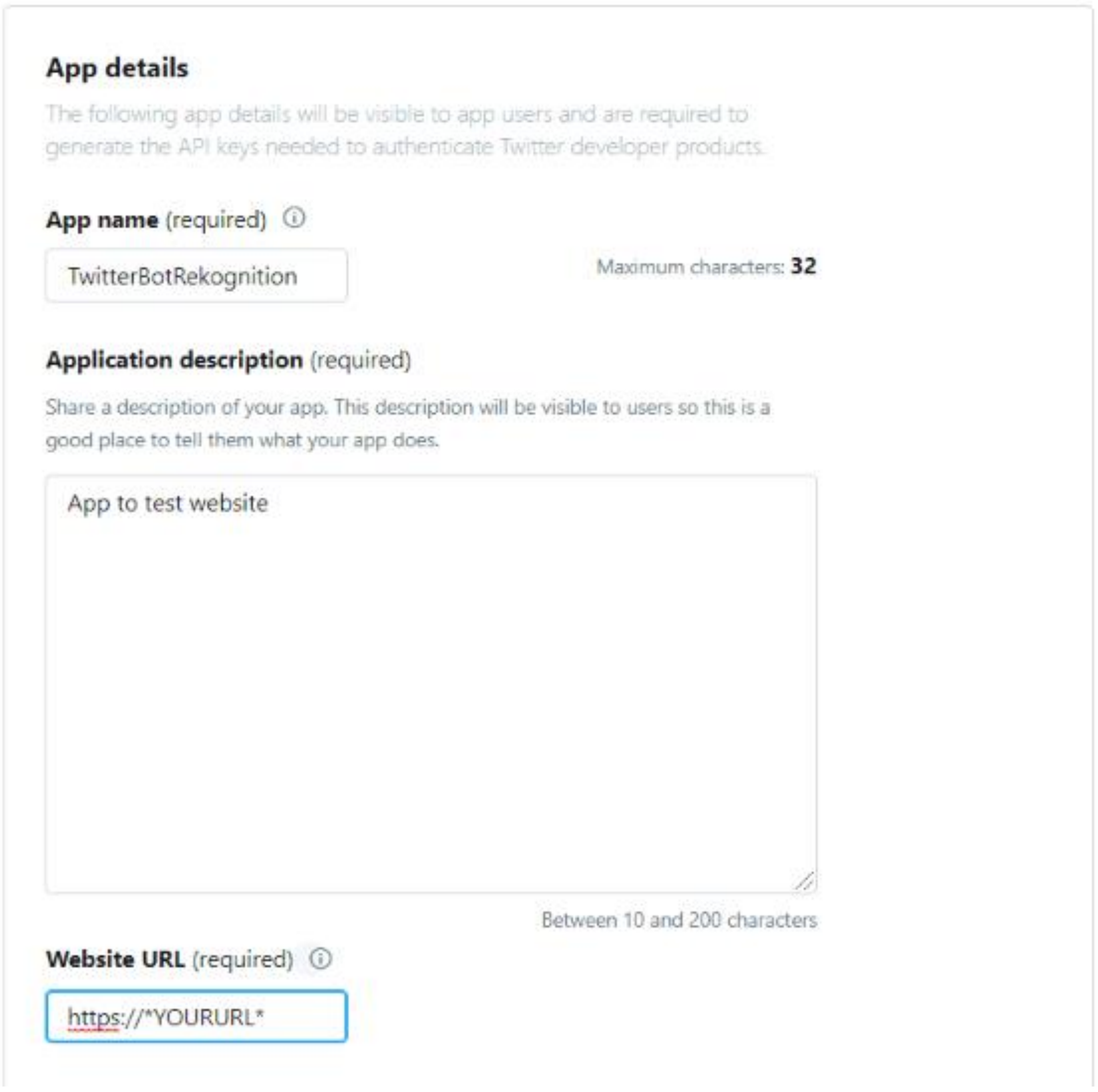
2. In the top right corner of the screen, Click on your user and select Apps from the drop down menu.



3. Click on “Create an App”



4. Fill in the following “required fields” and click create.



App details

The following app details will be visible to app users and are required to generate the API keys needed to authenticate Twitter developer products.

App name (required)

TwitterBotRekognition

Maximum characters: 32

Application description (required)

Share a description of your app. This description will be visible to users so this is a good place to tell them what your app does.

App to test website

Between 10 and 200 characters

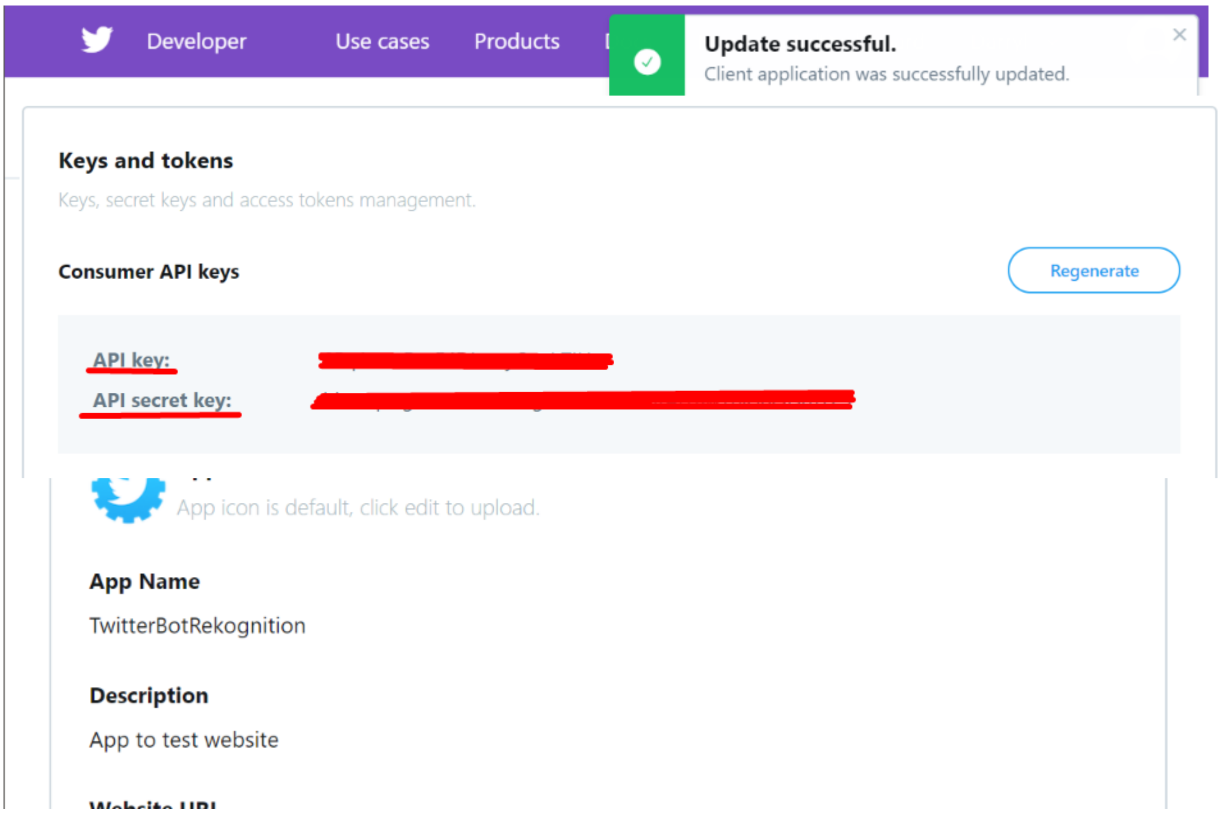
Website URL (required)

https://*YOURURL*

5. In the field “Tell us how this app will be used” copy and paste the following:

   “I am a computer science student working on a cloud computing project that integrates Twitter and AWS serverless microservices. I will be primarily using the Twitter API with AWS Rekognition, an image and video analysis service. After Rekognition performs an analysis, I will use the Twitter API to tweet from my developer account.“

6. After finishing, you should see a screen with your API key and API secret key.



Update successful.
Client application was successfully updated.

Keys and tokens

Keys, secret keys and access tokens management.

Consumer API keys

API key: [redacted]

API secret key: [redacted]

App icon is default, click edit to upload.

App Name

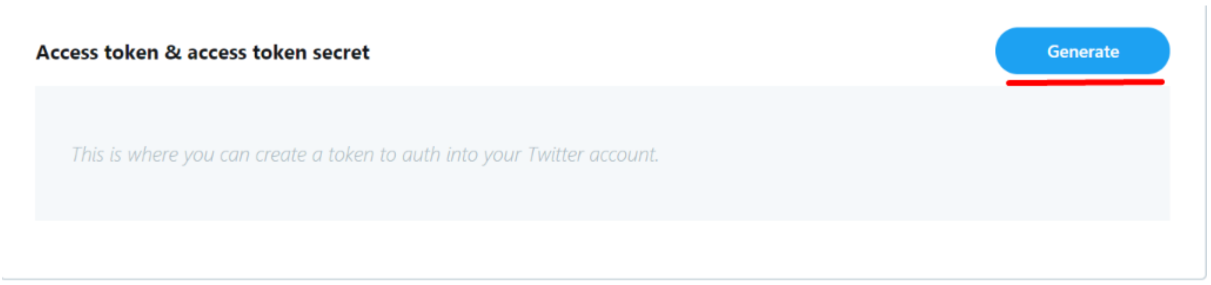
TwitterBotRekognition

Description

App to test website

7. Click on the “Key and Tokens” tab and proceed to copy and save both the Consumer API key and Consumer Secret API key

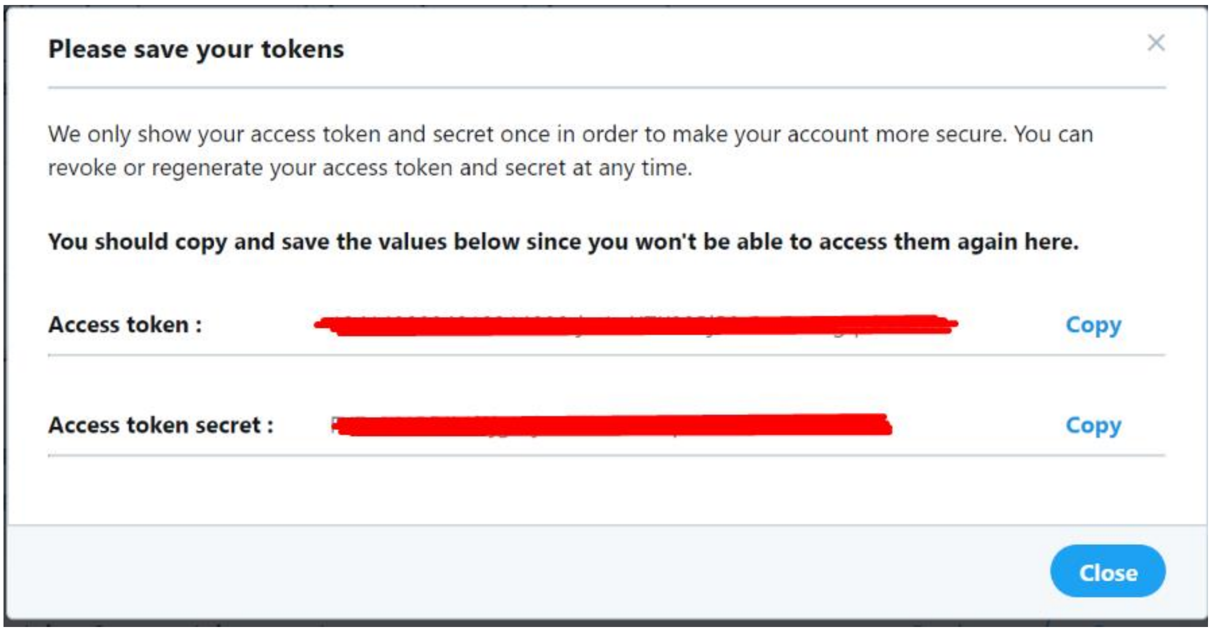
8. Click the “Generate” button



Access token & access token secret

This is where you can create a token to auth into your Twitter account.

9. Copy both the Access Token and Access Token Secret.



Please save your tokens

We only show your access token and secret once in order to make your account more secure. You can revoke or regenerate your access token and secret at any time.

You should copy and save the values below since you won't be able to access them again here.

Access token :	[redacted]	Copy
Access token secret :	[redacted]	Copy

10. Now let's set up the rest of our elements so we know who to tweet when Rekognition recognises a celebrity.

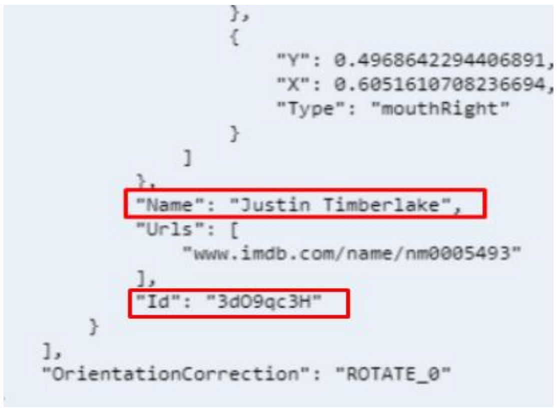
    Upload **c01.jpg** from img/part03 to your S3 bucket from earlier. **c01.jpg** is a picture of Justin Timberlake.

11. Open up Cloud9 so we can use the **recognize-celebrities** feature on the go.

    Enter this command into AWS CLI. Remember to change to your own bucket name.

```
 aws rekognition recognize-celebrities --image
"S3Object={Bucket=a01041926-rekognition,Name=c01.jpg}"
```

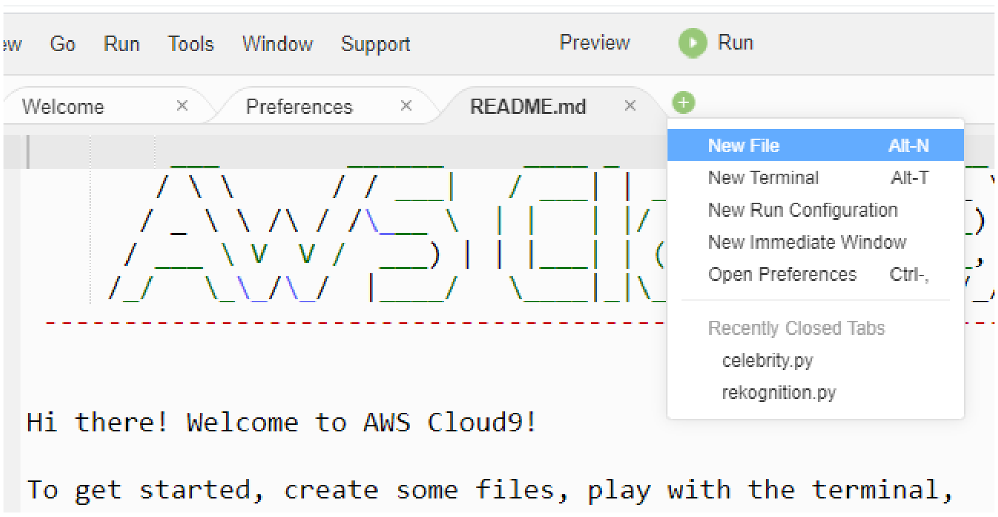
12. Near the bottom of the output you should see the celebrity name and their unique ID.



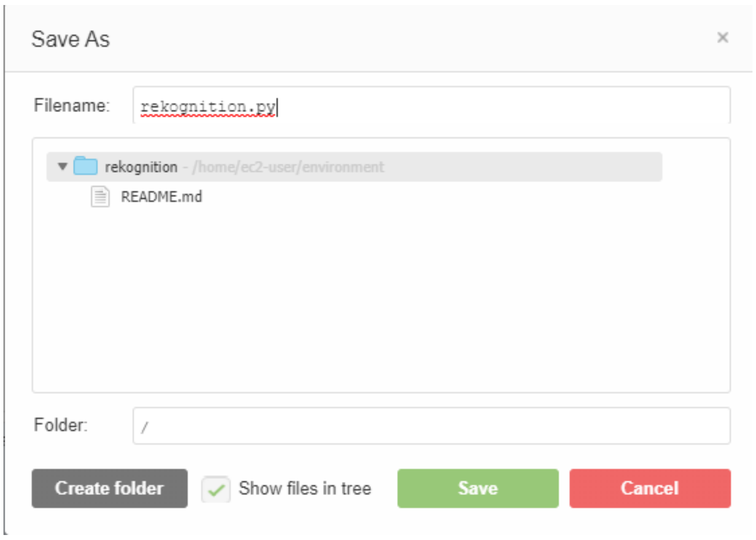
```
                    },
                    {
                        "Y": 0.4968642294406891,
                        "X": 0.6051610708236694,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Justin Timberlake",
            "Urls": [
                "www.imdb.com/name/nm0005493"
            ],
            "Id": "3dO9qc3H"
        }
    ],
    "OrientationCorrection": "ROTATE_0"
```

Keep this in mind as we'll be using the ID later in our database.

13. In Cloud9, create a new python file, **rekognition.py**.



```
Hi there! Welcome to AWS Cloud9!

To get started, create some files, play with the terminal,
```

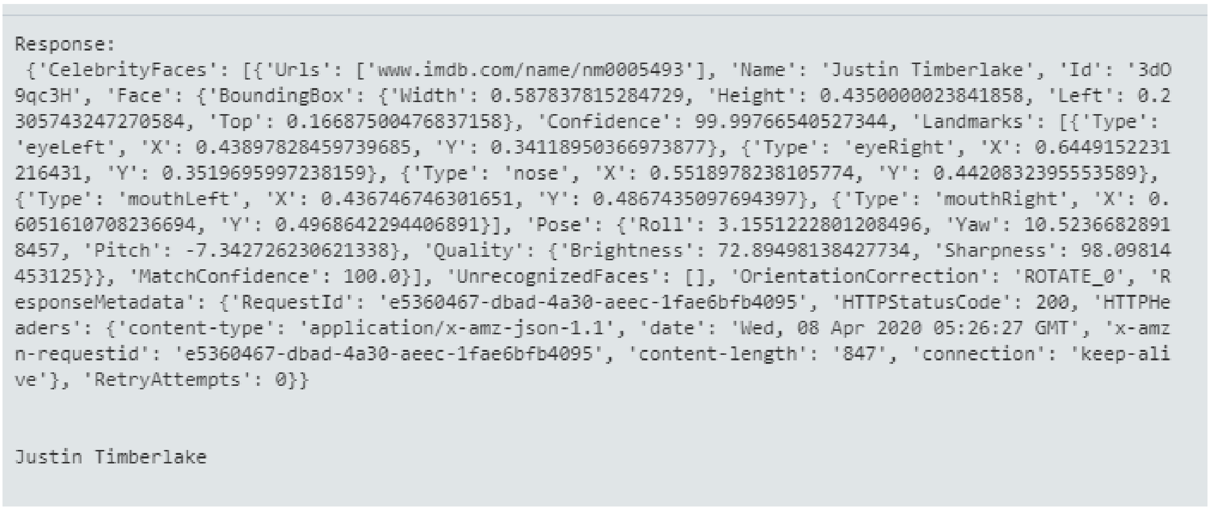


Add the following code snippet and save. Make sure you replace the bucket name with the name of your own.

```
from __future__ import print_function
import boto3

session = boto3.session.Session(region_name="us-west-2")
rek = session.client('rekognition')

response = rek.recognize_celebrities(
    Image={
        'S3Object': {
            'Bucket': 'a01041926-rekognition',
            'Name': 'c04.jpg',
        }
    }
)

print("Response:\n", response, "\n\n")
for celeb in response['CelebrityFaces']:
    print("{0} -- ID {1}".format(celeb["Name"], celeb["Id"]))
```

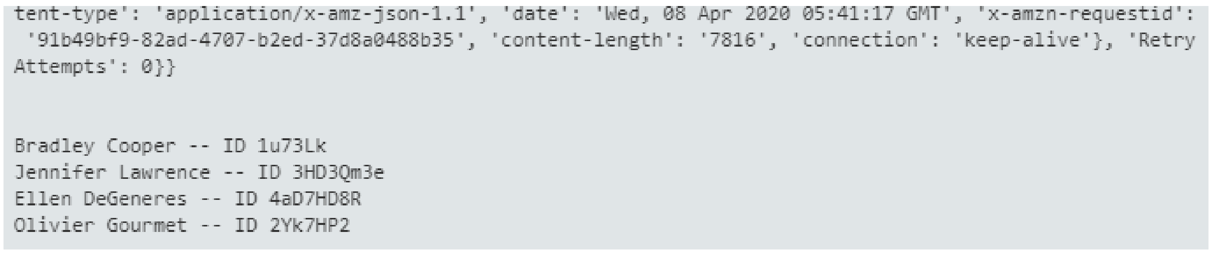
14. What happens when you run it? This may take a few seconds. Be patient. (You can run the code either by clicking the green “Run” button or right clicking your file and selecting “Run”.)

    You should get the following output:

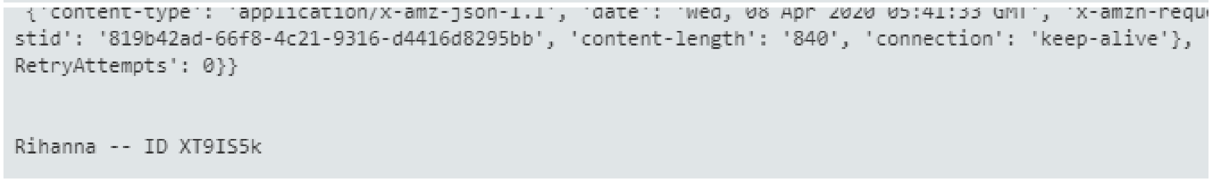


```
Response:
 {'CelebrityFaces': [{'Urls': ['www.imdb.com/name/nm0005493'], 'Name': 'Justin Timberlake', 'Id': '3dO
9qc3H', 'Face': {'BoundingBox': {'Width': 0.587837815284729, 'Height': 0.4350000023841858, 'Left': 0.2
305743247270584, 'Top': 0.16687500476837158}, 'Confidence': 99.99766540527344, 'Landmarks': [{'Type':
'eyeLeft', 'X': 0.43897828459739685, 'Y': 0.34118950366973877}, {'Type': 'eyeRight', 'X': 0.6449152231
216431, 'Y': 0.3519695997238159}, {'Type': 'nose', 'X': 0.5518978238105774, 'Y': 0.4420832395553589},
{'Type': 'mouthLeft', 'X': 0.436746746301651, 'Y': 0.4867435097694397}, {'Type': 'mouthRight', 'X': 0.
6051610708236694, 'Y': 0.4968642294406891}], 'Pose': {'Roll': 3.1551222801208496, 'Yaw': 10.5236682891
8457, 'Pitch': -7.342726230621338}, 'Quality': {'Brightness': 72.89498138427734, 'Sharpness': 98.09814
453125}}, 'MatchConfidence': 100.0}], 'UnrecognizedFaces': [], 'OrientationCorrection': 'ROTATE_0', 'R
esponseMetadata': {'RequestId': 'e5360467-dbad-4a30-aeec-1fae6bfb4095', 'HTTPStatusCode': 200, 'HTTPHe
aders': {'content-type': 'application/x-amz-json-1.1', 'date': 'Wed, 08 Apr 2020 05:26:27 GMT', 'x-amz
n-requestid': 'e5360467-dbad-4a30-aeec-1fae6bfb4095', 'content-length': '847', 'connection': 'keep-ali
ve'}, 'RetryAttempts': 0}}


Justin Timberlake
```

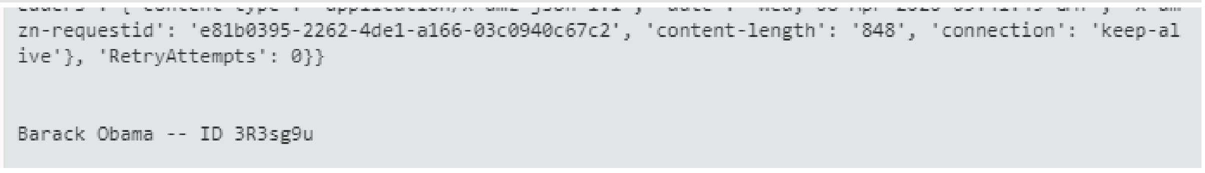
    You can try this on other photos. Upload **c02.jpg**, **c03.jpg**, and **c04.jpg** to try them out:



```
tent-type': 'application/x-amz-json-1.1', 'date': 'Wed, 08 Apr 2020 05:41:17 GMT', 'x-amzn-requestid':
 '91b49bf9-82ad-4707-b2ed-37d8a0488b35', 'content-length': '7816', 'connection': 'keep-alive'}, 'Retry
Attempts': 0}}


Bradley Cooper -- ID 1u73Lk
Jennifer Lawrence -- ID 3HD3Qm3e
Ellen DeGeneres -- ID 4aD7HD8R
Olivier Gourmet -- ID 2Yk7HP2
```



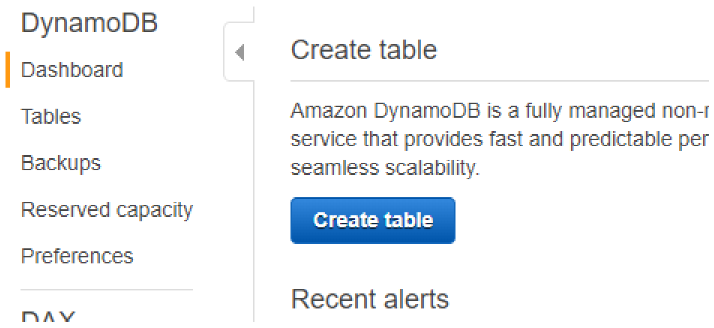
```
 {'content-type': 'application/x-amz-json-1.1', 'date': 'Wed, 08 Apr 2020 05:41:33 GMT', 'x-amzn-requ
stid': '819b42ad-66f8-4c21-9316-d4416d8295bb', 'content-length': '840', 'connection': 'keep-alive'},
RetryAttempts': 0}}


Rihanna -- ID XT9IS5k
```



```
[illegible]
zn-requestid': 'e81b0395-2262-4de1-a166-03c0940c67c2', 'content-length': '848', 'connection': 'keep-al
ive'}, 'RetryAttempts': 0}}


Barack Obama -- ID 3R3sg9u
```

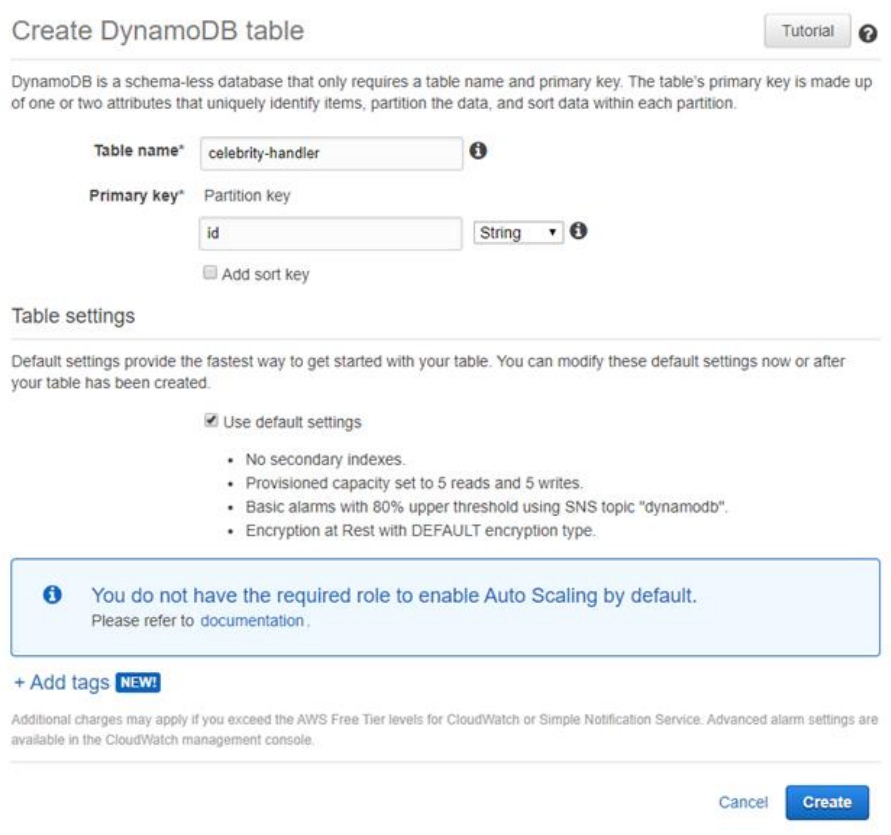
15. Let’s create a dynamoDB table so we can store celebrities and their twitter handles.

    Go to DynamoDB under AWS and click Create Table.



Amazon DynamoDB is a fully managed non-r
service that provides fast and predictable per
seamless scalability.

16. Table name is **celebrity-handler**
    Primary key is **id** (String)
    Accept the default settings and click Create.


Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* celebrity-handler

Primary key* Partition key

id String

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

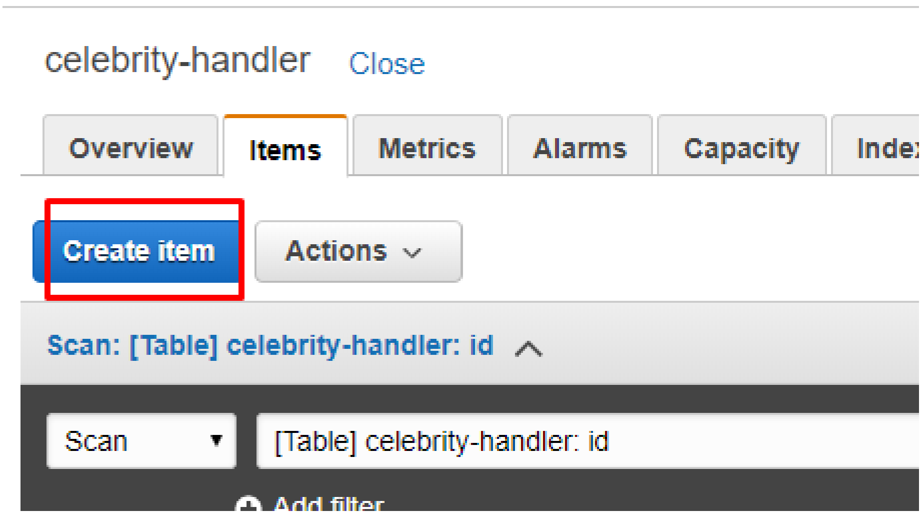
You do not have the required role to enable Auto Scaling by default.
Please refer to documentation.

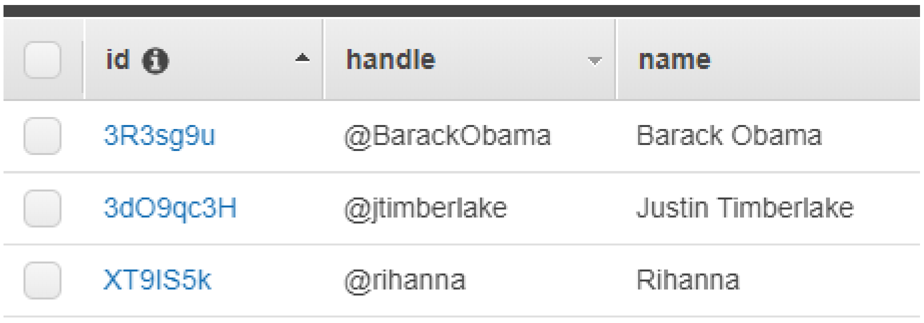
+ Add tags NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

17. We are going to use the celebrity IDs generated by rekognition as our **id**, and store their twitter handles under **handle**. We will also add a **name** column.

    Click Create Item and plug the data in.





	id	handle	name
	3R3sg9u	@BarackObama	Barack Obama
	3dO9qc3H	@jtimberlake	Justin Timberlake
	XT9IS5k	@rihanna	Rihanna

18. Alternatively, you can programatically create the items. Let’s create a new file called **twitterbot.py**.

    Add the following code to set up programmatically accessing and querying db values.

```
from __future__ import print_function

import boto3
import json
from twython import Twython

session = boto3.session.Session(region_name="us-west-2")
rek = session.client('rekognition')
table = 'celebrity-handler'
ddb = session.resource('dynamodb').Table(table)
```

    You can programmatically add items into your **celebrity-handler** database by using the following code:

```
ddb.put_item(Item={"id": "3R3sg9u", "handle" :
"@BarackObama", “name”:”Barack Obama”})
```

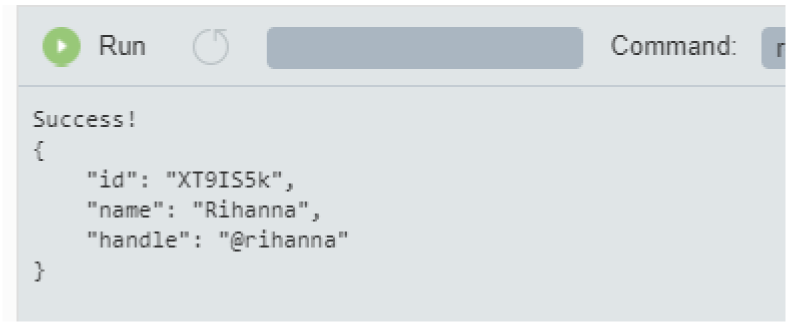
19. Let’s try to query from the database and see if it works.

    The ID for Rihanna is **XT9IS5k**.

    Add the following code to your file:

```
try:
    query = ddb.get_item(TableName=table, Key={
        "id": "XT9IS5k"
    })
except Exception as e:
    print(e)
else:
    print("Success!")
    print(json.dumps(query["Item"], indent=4))
```

You should see this:



```
Success!
{
    "id": "XT9IS5k",
    "name": "Rihanna",
    "handle": "@rihanna"
}
```

20. Now we're ready to bring it all together.

    Add the following snippet after the line that declares
    ddb = session.resource('dynamob').Table(table)

```
with open('creds.json') as f:
    c = json.loads(f.read())

twitter = Twython(c["consumer_key"],
c["consumer_secret"],
                  c["access_token_key"],
c["access_token_secret"])
```

21. Notice that our code will attempt to open a file called **creds.json** that hasn't been created yet.

    This should sound familiar as we did a similar step in our Twitter lab.

The creds.json is a file that stores the credentials necessary to authenticate and authorise access to your Twitter app. We copied down these values earlier, and they will come in handy now.

NOTE: It is a VERY bad idea to store your access keys unencrypted. But the KMS is not the focus of our lab today. We will be storing our keys directly in creds.json simply as a proof of concept and as a way to explore AWS Rekognition's features and capabilities.

Best practices involve using an encrypted key that refers back to AWS KMS which will manage your access keys for you. You can follow these steps starting from **page 12** in the **Twitter lab**.

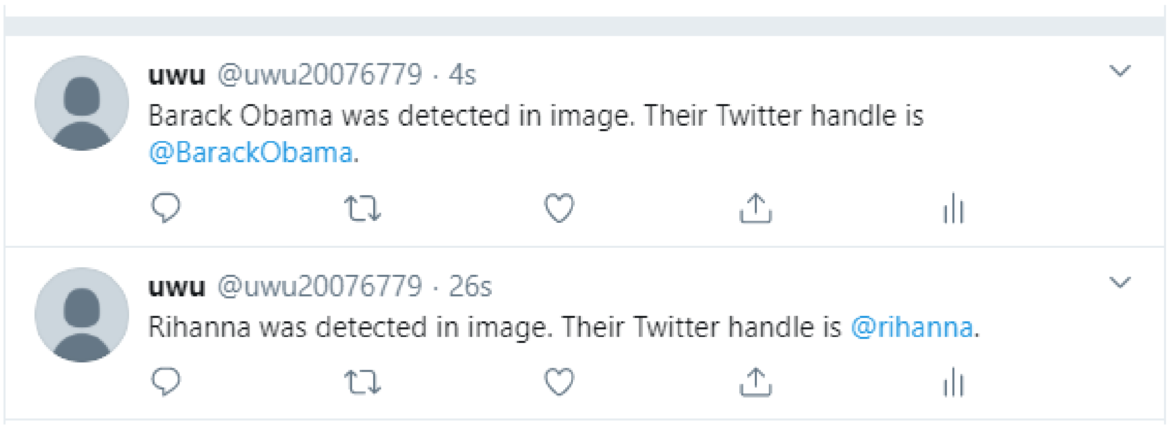
22. Copy and paste your access keys and save **creds.json** on the same level as **twitterbot.py**

```
{
    "consumer_key": "THE_API_KEY_VALUE",
    "consumer_secret": "THE_API_SECRET_VALUE",
    "access_token_key": "TOKEN_KEY_VALUE",
    "access_token_secret": "TOKEN_SECRET_VALUE"
}
```

23. We're ready to bring everything together.
    Try and create the following methods so that a tweet will be posted whenever you change the value in main().

```
def use_rekognition(bucket, filename)
def query_db(id)
def parse_queries(query_list)
def post_msglist(content_list)
def main()
```

Your results should look something like this!


uwu @uwu20076779 · 4s
Barack Obama was detected in image. Their Twitter handle is @BarackObama.

uwu @uwu20076779 · 26s
Rihanna was detected in image. Their Twitter handle is @rihanna.

If you want to make it so that these tweets are sent out whenever an image is uploaded, you can easily place the logic in your file into a lambda function set to trigger on all create object events in your S3 bucket.

FINAL NOTES

The code used to achieve these tasks in the various prompts throughout this lab is provided in the .zip file, under the code directory.