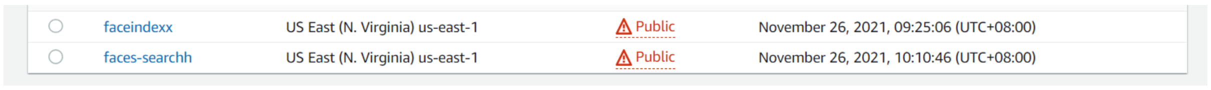
Configuration Guide 😊

1. Create 2 buckets

(Change bucket names accordingly because i have used this bucket name already so need to change it if you want to test my codes)

* AWS S3 don’t allow similar bucket names that others use so change accordingly for all the codes

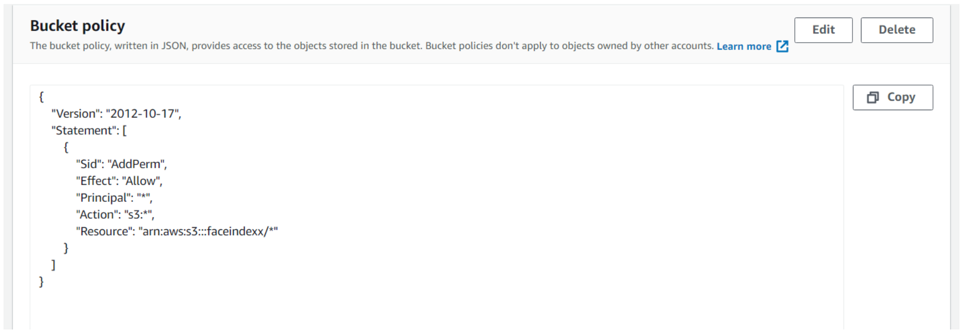
1. faceindexx



○	faceindexx	US East (N. Virginia) us-east-1	⚠ Public	November 26, 2021, 09:25:06 (UTC+08:00)
○	faces-searchh	US East (N. Virginia) us-east-1	⚠ Public	November 26, 2021, 10:10:46 (UTC+08:00)

Uncheck all boxes for public access

Bucket Policy for faceindexx

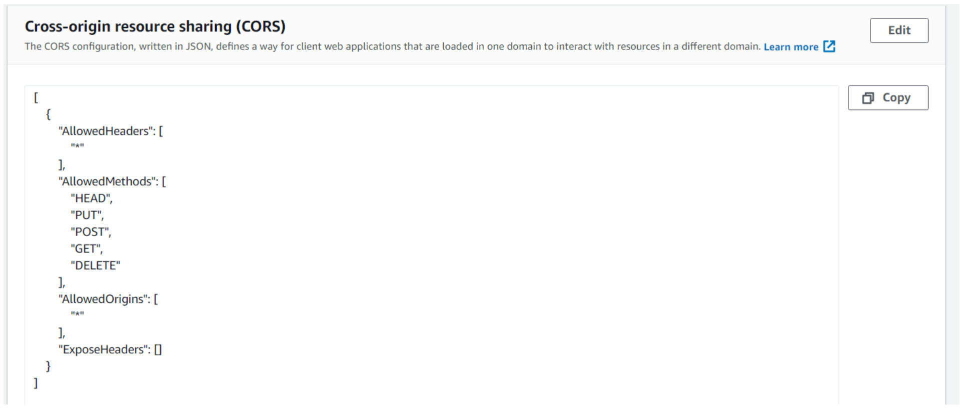


Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. Learn more

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AddPerm",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::faceindexx/*"
        }
    ]
}
```

CORS



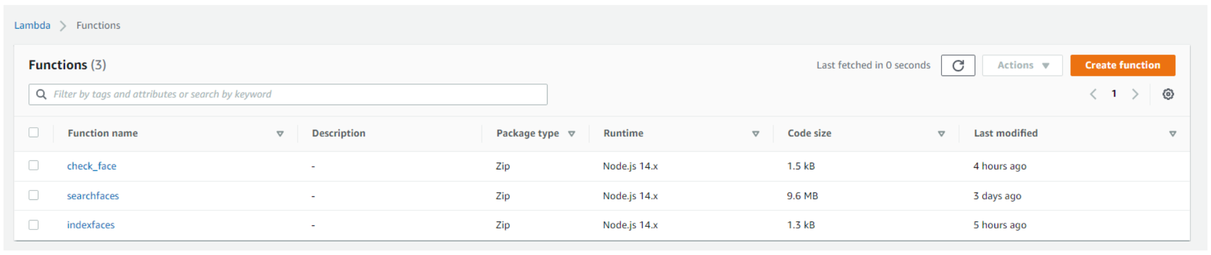
Cross-origin resource sharing (CORS)

The CORS configuration, written in JSON, defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. Learn more

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "HEAD",
            "PUT",
            "POST",
            "GET",
            "DELETE"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

Do the same for face-searchh bucket

2.Create 3 Lambda Functions



Lambda > Functions

Functions (3)

Last fetched in 0 seconds

Function name	Description	Package type	Runtime	Code size	Last modified
check_face	-	Zip	Node.js 14.x	1.5 kB	4 hours ago
searchfaces	-	Zip	Node.js 14.x	9.6 MB	3 days ago
indexfaces	-	Zip	Node.js 14.x	1.3 kB	5 hours ago

1. *indexfaces*

a. Need to add in imageMagick layer

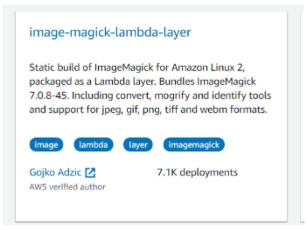


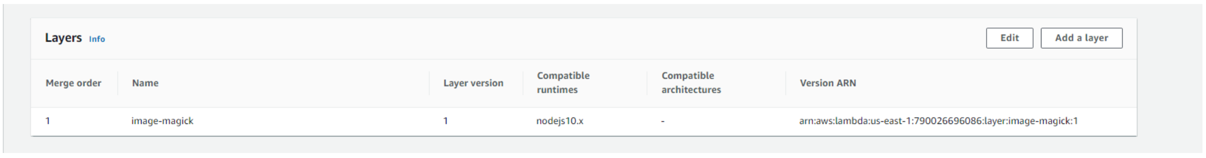
image-magick-lambda-layer

Static build of ImageMagick for Amazon Linux 2, packaged as a Lambda layer. Bundles ImageMagick 7.0.8-45. Including convert, mogrify and identify tools and support for jpeg, gif, png, tiff and webm formats.

https://serverlessrepo.aws.amazon.com/applications

Click this, and you will get a "deploy" option, click deploy.

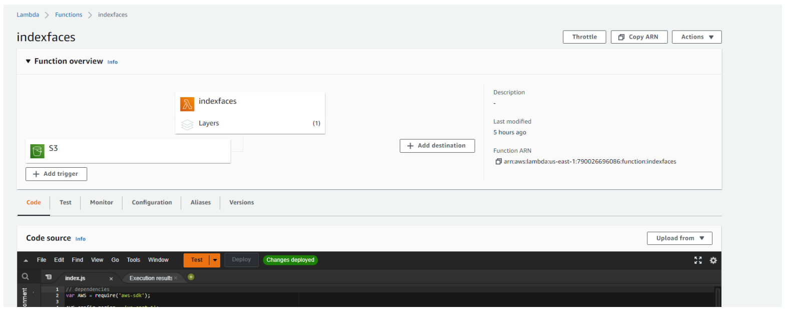
Sign in to console, go to the lambda function , add this layer via the ARN



Layers Info

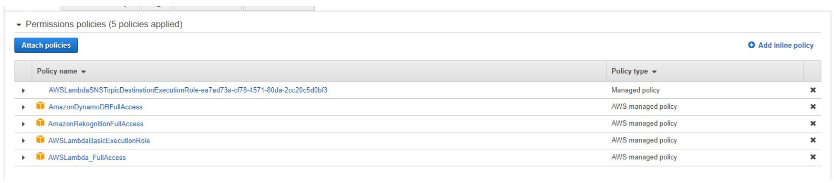
Merge order	Name	Layer version	Compatible runtimes	Compatible architectures	Version ARN
1	image-magick	1	nodejs10.x	-	arn:aws:lambda:us-east-1:790026696086:layer:image-magick:1

Add trigger (S3 faceindexx trigger to this function)



Description
-
Last modified
5 hours ago
Function ARN
arn:aws:lambda:us-east-1:790026696086:function:indexfaces

```
// dependencies
var AWS = require('aws-sdk');
```



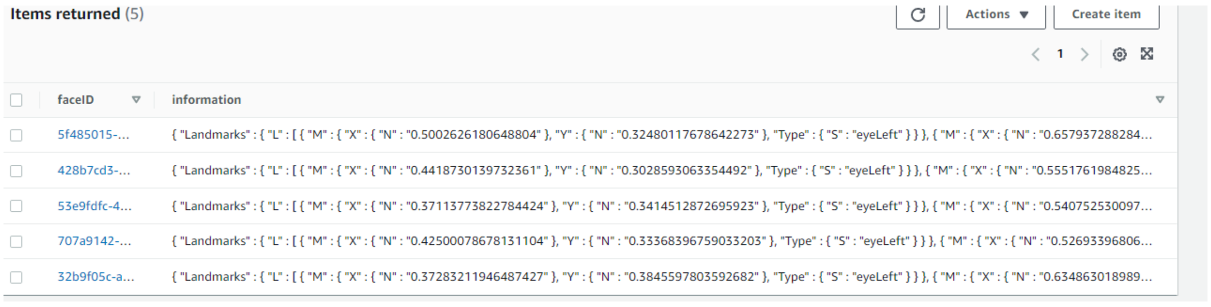
Permissions policies (5 policies applied)

Policy name	Policy type
AWSLambdaSNSTopicDestinationExecutionRole-ea7ad73a-cf78-4571-80da-2cc20c5d0bf3	Managed policy
AmazonDynamoDBFullAccess	AWS managed policy
AmazonRekognitionFullAccess	AWS managed policy
AWSLambdaBasicExecutionRole	AWS managed policy
AWSLambda_FullAccess	AWS managed policy

c. Add the codes in.

Inside the ITAD zip file, there is a file name "index.js". CTRL+A, copy all and paste in the code section.

d. Add the data into DynamoDB



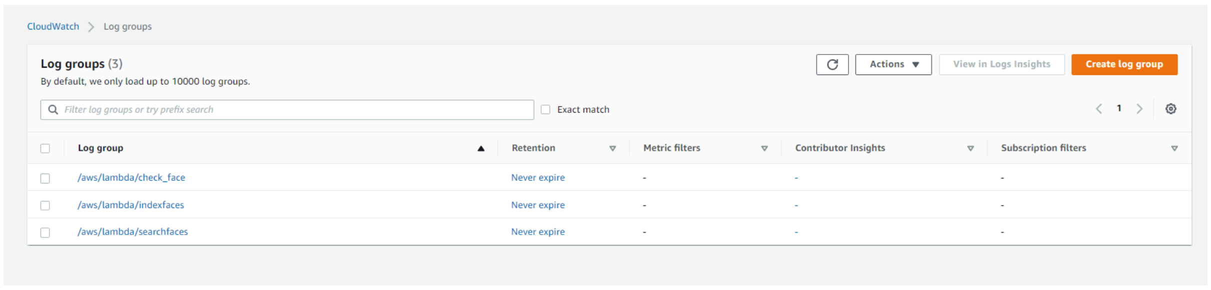
Items returned (5)

faceID	information
5f485015-...	{ "Landmarks" : { "L" : [{ "M" : { "X" : { "N" : "0.5002626180648804" }, "Y" : { "N" : "0.32480117678642273" }, "Type" : { "S" : "eyeLeft" } } }, { "M" : { "X" : { "N" : "0.657937288284...
428b7cd3-...	{ "Landmarks" : { "L" : [{ "M" : { "X" : { "N" : "0.4418730139732361" }, "Y" : { "N" : "0.3028593063354492" }, "Type" : { "S" : "eyeLeft" } } }, { "M" : { "X" : { "N" : "0.5551761984825...
53e9fdfc-4...	{ "Landmarks" : { "L" : [{ "M" : { "X" : { "N" : "0.37113773822784424" }, "Y" : { "N" : "0.3414512872695923" }, "Type" : { "S" : "eyeLeft" } } }, { "M" : { "X" : { "N" : "0.540752530097...
707a9142-...	{ "Landmarks" : { "L" : [{ "M" : { "X" : { "N" : "0.42500078678131104" }, "Y" : { "N" : "0.33368396759033203" }, "Type" : { "S" : "eyeLeft" } } }, { "M" : { "X" : { "N" : "0.52693396806...
32b9f05c-a...	{ "Landmarks" : { "L" : [{ "M" : { "X" : { "N" : "0.37283211946487427" }, "Y" : { "N" : "0.3845597803592682" }, "Type" : { "S" : "eyeLeft" } } }, { "M" : { "X" : { "N" : "0.634863018989...

2. *searchFaces*

a. do the same as indexfaces – add the imageMagick layer

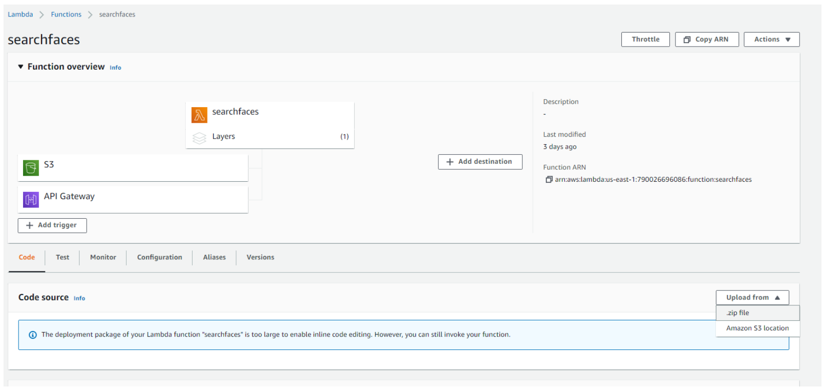
b. create log group, go to cloudWatch log, then create log group. Name group as follows, "aws/lambda/bucketName"



Log groups (3)
By default, we only load up to 10000 log groups.

Log group	Retention	Metric filters	Contributor Insights	Subscription filters
/aws/lambda/check_face	Never expire	-	-	-
/aws/lambda/indexfaces	Never expire	-	-	-
/aws/lambda/searchfaces	Never expire	-	-	-

c. use the same role as above.



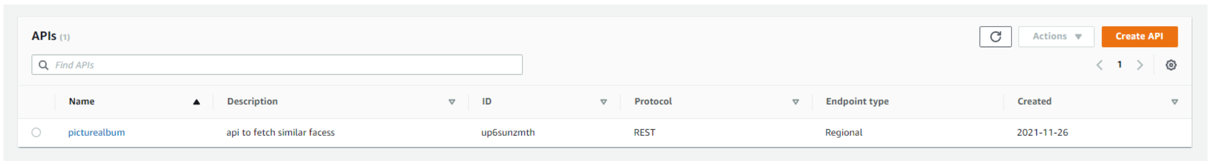
Description
-
Last modified
3 days ago
Function ARN
arn:aws:lambda:us-east-1:790026696086:function:searchfaces

The deployment package of your Lambda function "searchfaces" is too large to enable inline code editing. However, you can still invoke your function.

Add searchFacess zip file into the code source. Add the S3 bucket(faces-search) trigger and and the API Gateway.

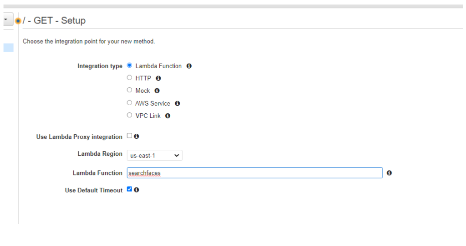
Browse to API gateway. Click on create API -> REST API ->Build-> add the name you want to name the API.

Create Method => go to get request -> click lambda function and add in which function to connect to.



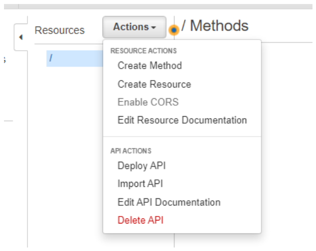
APIs (1)

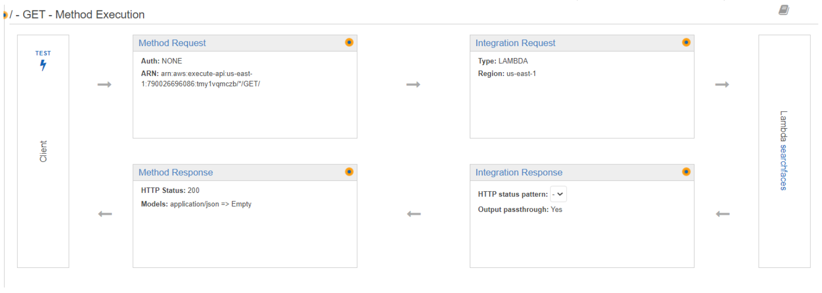
Name	Description	ID	Protocol	Endpoint type	Created
picturealbum	api to fetch similar facess	up6sunzmth	REST	Regional	2021-11-26


/ - GET - Setup

Choose the integration point for your new method.

Integration type: [✓ Lambda Function] HTTP Mock AWS Service VPC Link
Use Lambda Proxy integration: ☐
Lambda Region: us-east-1
Lambda Function: searchfaces
Use Default Timeout: ☑





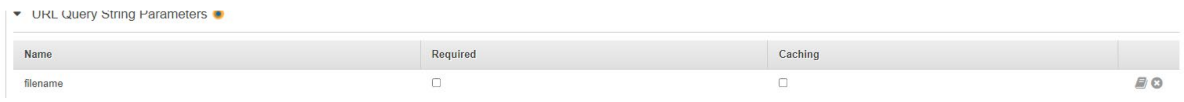
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:790026696086:tmy1vqmczb/*/GET/

Type: LAMBDA
Region: us-east-1

HTTP Status: 200
Models: application/json => Empty

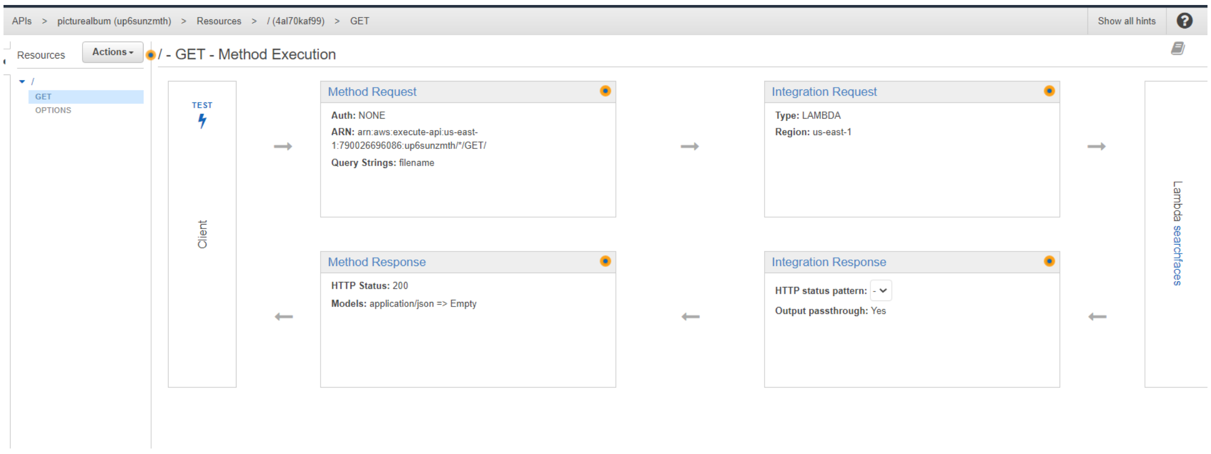
HTTP status pattern: -
Output passthrough: Yes

click Method Request, Under URL Query Parameters, add query string as filename


URL Query String Parameters

Name	Required	Caching
filename	☐	☐

Once successfully done, it should look like this:


APIs > picturealbum (up6sunzmth) > Resources > / (4al70kaf99) > GET

/ - GET - Method Execution

Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:790026696086:up6sunzmth/*/GET/
Query Strings: filename

Integration Request
Type: LAMBDA
Region: us-east-1

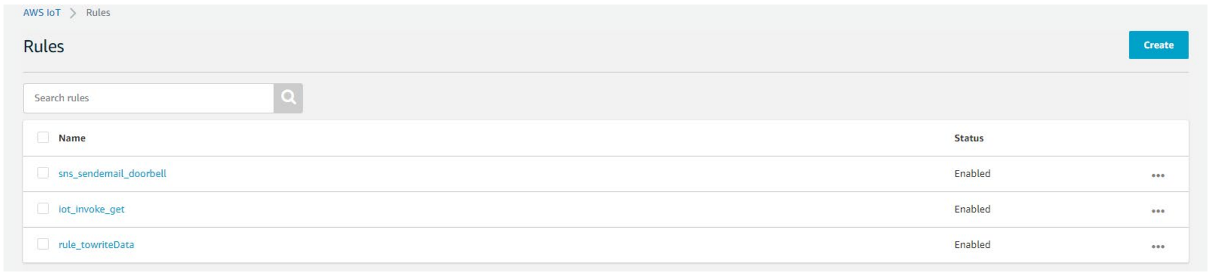
Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

3. *check_face*

a. open up iot_check_button.

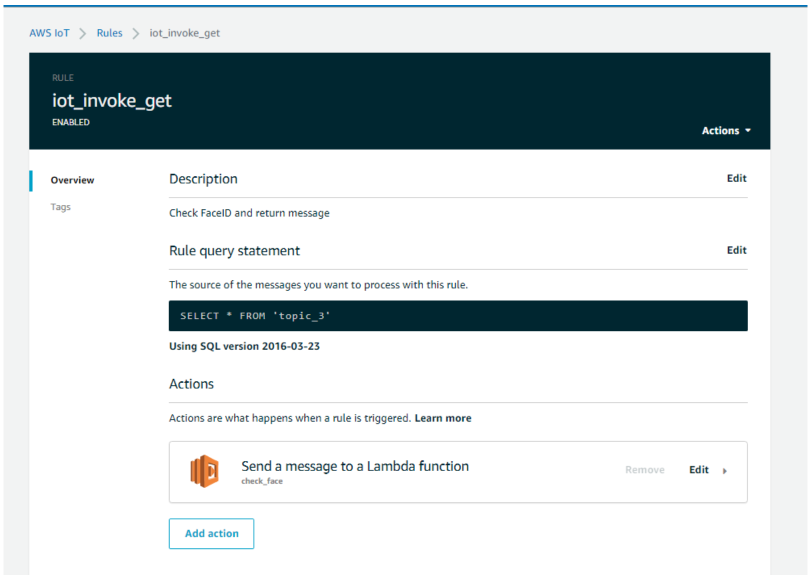
b. create thing, and rule.


AWS IoT > Rules

Rules

Name	Status
sns_sendemail_doorbell	Enabled
iot_invoke_get	Enabled
rule_towriteData	Enabled

Configure like this:


AWS IoT > Rules > iot_invoke_get

RULE
iot_invoke_get
ENABLED

Description
Check FaceID and return message

Rule query statement
The source of the messages you want to process with this rule.

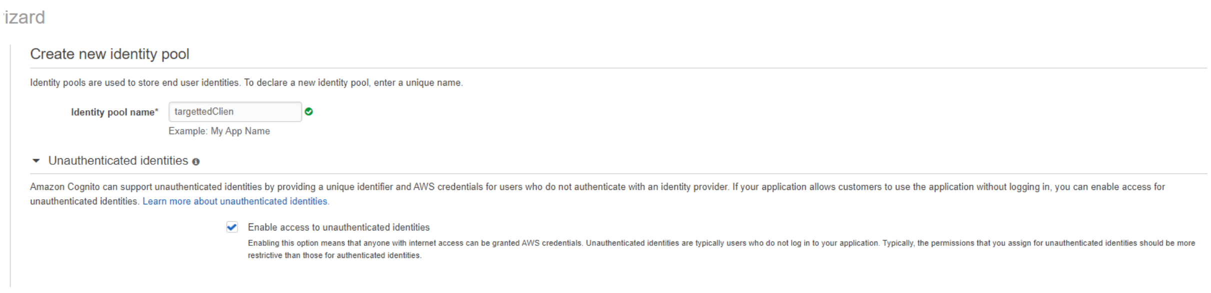
```
SELECT * FROM 'topic_3'
```

Using SQL version 2016-03-23

Actions
Actions are what happens when a rule is triggered. Learn more

Send a message to a Lambda function
check_face

3. To Allow Unauthenticated Access, navigate to amazon Cognito -> manage Identity Pools -> create new identity pool


izard

Create new identity pool

Identity pools are used to store end user identities. To declare a new identity pool, enter a unique name.

Identity pool name* targettedClien

Example: My App Name

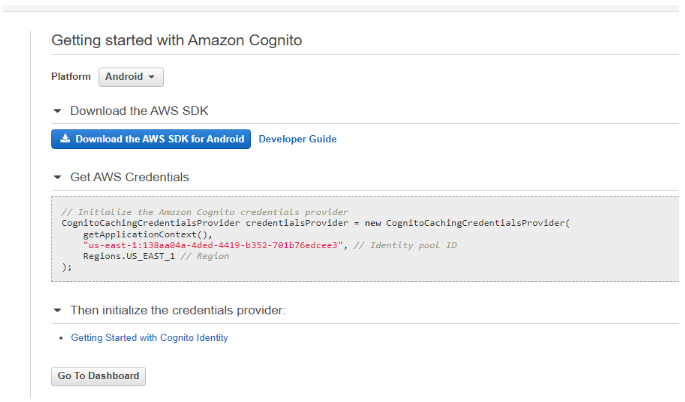
▾ Unauthenticated identities

Amazon Cognito can support unauthenticated identities by providing a unique identifier and AWS credentials for users who do not authenticate with an identity provider. If your application allows customers to use the application without logging in, you can enable access for unauthenticated identities. Learn more about unauthenticated identities.

✓ Enable access to unauthenticated identities

Enabling this option means that anyone with internet access can be granted AWS credentials. Unauthenticated identities are typically users who do not log in to your application. Typically, the permissions that you assign for unauthenticated identities should be more restrictive than those for authenticated identities.

Allow -> OK -> go under sample code and you get your identity pool Id


Getting started with Amazon Cognito

Platform: Android

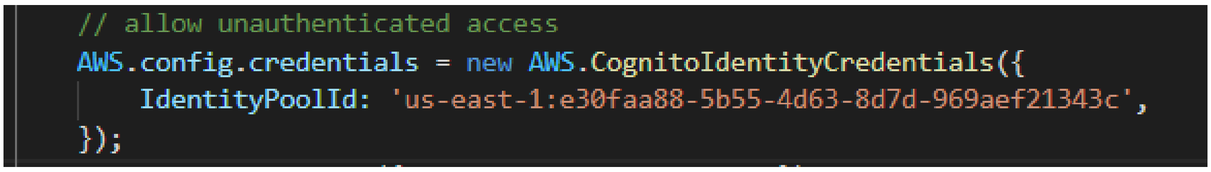
▾ Download the AWS SDK

▾ Get AWS Credentials

```
// Initialize the Amazon Cognito credentials provider
CognitoCachingCredentialsProvider credentialsProvider = new CognitoCachingCredentialsProvider(
    getApplicationContext(),
    "us-east-1:138aa04a-4ded-4419-b352-701b76edcee3", // Identity pool ID
    Regions.US_EAST_1 // Region
);
```

▾ Then initialize the credentials provider:

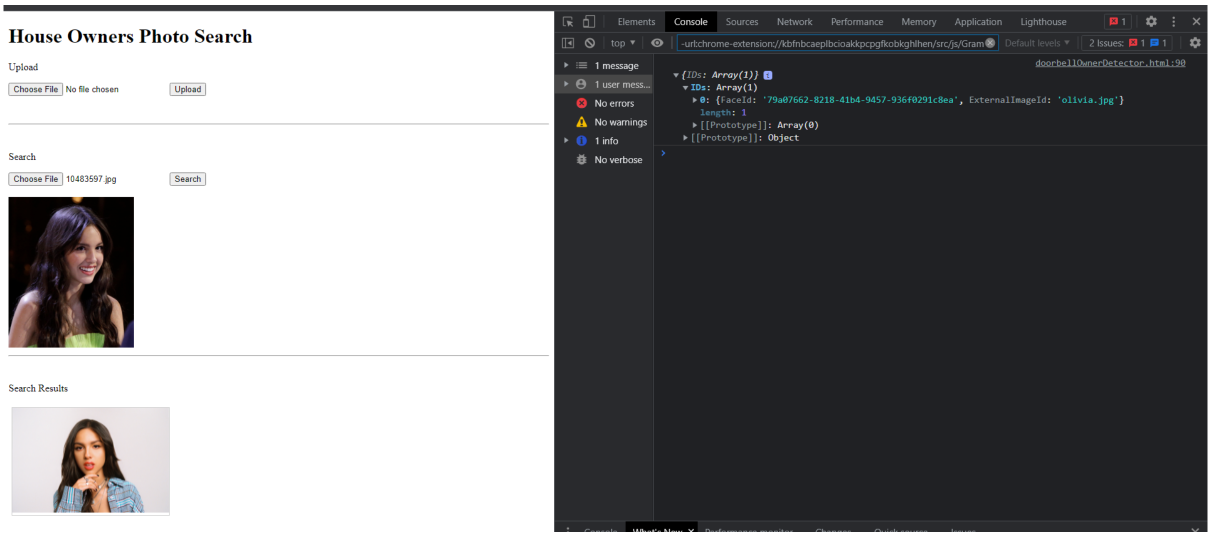
- Getting Started with Cognito Identity


```
// allow unauthenticated access
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:e30faa88-5b55-4d63-8d7d-969aef21343c',
});
```

Inside the html -> change the identity Id to your ID.

4. SNS -> create topic -> create subscription and add any valid email.

Testing :


House Owners Photo Search

Upload

Search

Search Results

-url:chrome-extension://kbfnbcaeplbcioakkpcpgfkobkghlhen/src/js/Gram

```
                                                        doorbellOwnerDetector.html:90
▾ {IDs: Array(1)}
  ▾ IDs: Array(1)
    ▸ 0: {FaceId: '79a07662-8218-41b4-9457-936f0291c8ea', ExternalImageId: 'olivia.jpg'}
      length: 1
    ▸ [[Prototype]]: Array(0)
  ▸ [[Prototype]]: Object
>
```

Copy the faceid from console, subscribe topic in MQTT client.