

Assignment5 보고서

컴퓨터공학부 201814121 이연희

1. System Design

1) doorCamera.js: 'faceRecog/request'를 Publish하고 S3 버킷의 있는 이미지의 정보를 랜덤으로 JSON 객체에 담아서 전송한다.

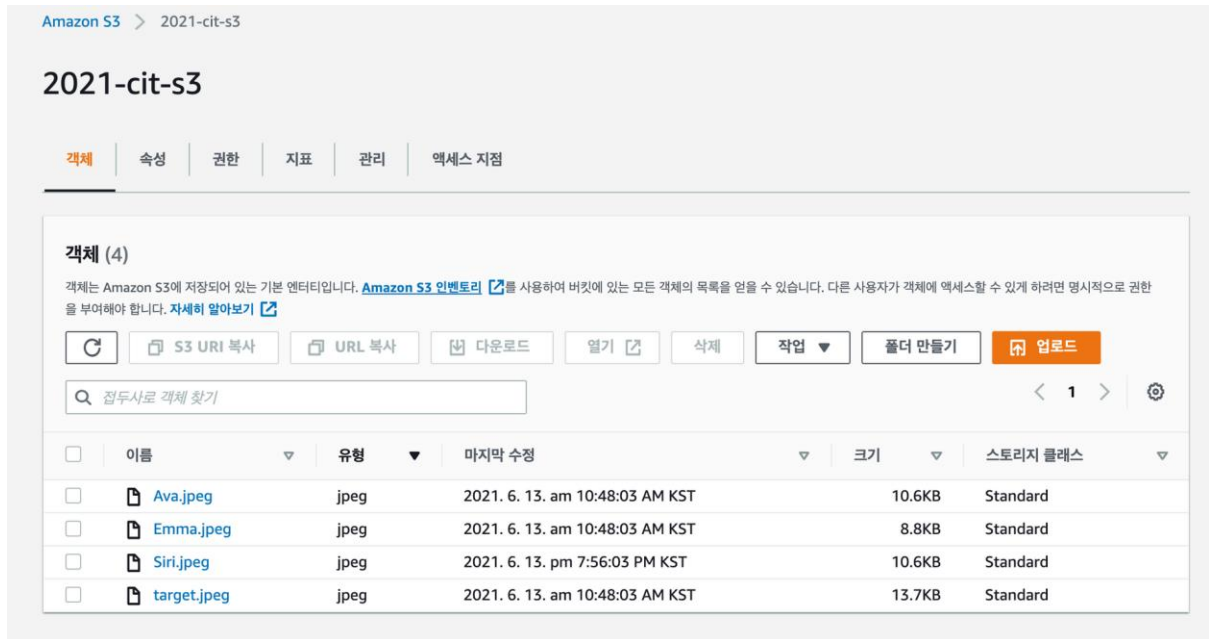
2) index.js: Rekognition 서비스를 이용하여 event값으로 들어온 이미지의 정보와 타겟 이미지의 정보를 변수로 담아 compareFaces() 함수를 호출한다. 결과 값을 받아서 토픽 'faceRecog/notify/door1'에 보낼 image, command 정보를 생성한다. 그리고 iotRule을 생성하여 람다 함수인 index.js에 트리거로 설정한다.

3) doorLock.js: 'faceRecog/notify/door1'을 Subscribe하고 람다함수에서 처리된 image, command 값을 받는다. 만약 command가 등록된 얼굴이라면(unlock) 문이 열림을 출력하고 아니라면(reject) 권한없는 얼굴임을 출력한다.

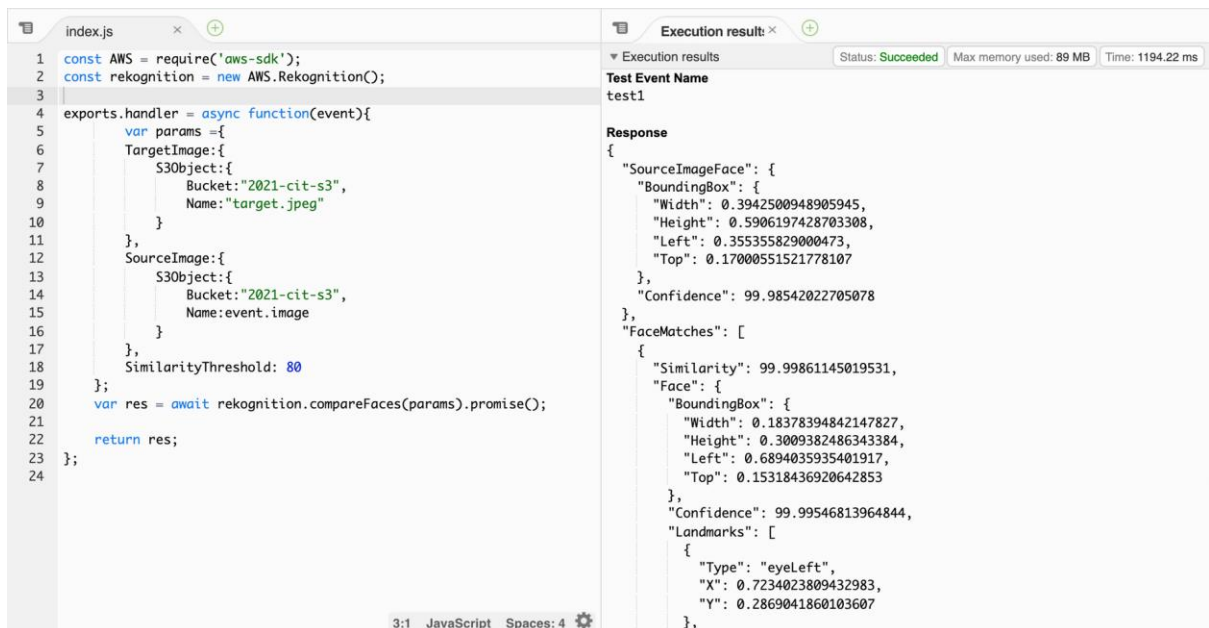
2. Task#1 Develop a Face Recognition Service (SaaS) on the Amazon Rekognition Service (PaaS): 디비에 있는 사진과 현재 방문자의 이미지를 비교하고 결과를 리턴하는 람다서비스

1) 총 4장의 이미지를 2021-cit-s3 버킷에 업로드 했다. 타겟이 되는 이미지(target.jpeg)와 비교 대상의 얼굴 사진(Ava.jpeg, Emma.jpeg, Siri.jpeg)이다.

target.jpeg	Ava.jpeg(o)	Emma.jpeg(o)	Siri.jpeg(x)
			



2) faceRecog라는 람다 함수를 생성했다. {"notify":"faceRecog/notify/door1", "image":"Ava.jpeg"}를 람다함수의 event 테스트 값으로 설정했고, 테스트한 결과이다. Rekognition 서비스를 이용하기 위해 AWS.Rekognition을 불러왔으며, params 변수에 타겟 이미지와 소스 이미지를 지정하였다. SimilarityThreshold값은 80으로 설정했고 compareFaces함수를 호출했다. 오른쪽 응답값의 FaceMatches의 값이 전달된 것을 보아 faceCompare이 제대로 동작한 것을 확인할 수 있다.



3) index.js 코드

3. Task#2 Develop a Face Recognition Platform (PaaS) on the Amazon Rekognition Service (PaaS):

1) 모든 'faceRecog/request' 토픽에 대한 값을 받는 SQL SELECT Clause를 작성하고 IoTRule을 생성했다. 토픽으로 들어오는 값이 존재하면 람다함수를 부를 수 있도록 작업을 설정했다. 이후 람다함수에 IoTRule을 트리거로 등록했다.

AWS IoT > 규칙 > IoTRule

규칙

IoTRule

비활성

작업 ▼

개요

설명

편집

Tags

설명이 없습니다

규칙 쿼리 설명문

편집


이 규칙을 사용하여 처리하고자 하는 메시지의 소스입니다.

SELECT * FROM 'faceRecog/request'

SQL 버전 사용 2016-03-23

작업

작업은 규칙이 트리거되면 이루어지는 것입니다. 자세히 알아보기

 메시지 데이터를 전달하는 Lambda 함수 호출

faceRecog

제거 편집 ▶

작업 추가

faceRecog

▼ 함수 개요 Info



AWS IoT

+ 트리거 추가



faceRecog



Layers

(0)

+ 대상 추가

3) event로 받은 값의 image를 소스이미지로 지정하고, compareFaces 함수를 호출했다. 만약 타겟 이미지에 없는 얼굴이라면 command를 reject로 설정하고, 있는 얼굴이라면 unlock으로 설정한다.

```
index.js
1 const AWS = require('aws-sdk');
2 const rekognition = new AWS.Rekognition();
3 var iotdata = new AWS.IotData({endpoint: 'a19tksa9tfqz8n-ats.iot.ap-northeast-2.amazonaws.com'});
4
5 exports.handler = async function(event){
6     var params = {
7         TargetImage: {
8             S3Object: {
9                 Bucket: '2021-cit-s3',
10                 Name: 'target.jpeg'
11             }
12         },
13         SourceImage: {
14             S3Object: {
15                 Bucket: '2021-cit-s3',
16                 Name: event.image
17             }
18         },
19         SimilarityThreshold: 80
20     };
21     var res = await rekognition.compareFaces(params).promise();
22     var command = (res.FaceMatches.length==0) ? 'reject':'unlock';
23     var params2 = {
24         topic: event.notify,
25         payload: JSON.stringify({'image':event.image, 'command':command}),
26         qos:0
27     };
28     console.log(params2);
29     var res2 = await iotdata.publish(params2).promise();
30
31     return res2;
32 }
```

4) doorCamera1.js에선 notify와 image 정보를 담은 메시지를 'faceRecog/request'토픽에 대해 반복적으로 publish한다. index.js에서 받은 이벤트 값에 대한 처리를 한 후 결과 메시지를 doorLock1에 필요한 형식(image, command)으로 토픽에 보낸다. doorLock1.js에선 'faceRecog/notify/door1'토픽에 대한 메시지인 image와 command를 받아서 등록된 얼굴에 대해 unlock door1을 출력한다.

오른쪽 출력 결과를 보면 알 수 있듯이 등록되어 있는 Emma에 대해서는 unlock door1을 출력하지만 Siri에 대해서는 unauthenticated person이라고 출력한다.


```

    };
    var res = await rekognition.compareFaces(params).promise();
    var command = (res.FaceMatches.length==0) ? 'reject':'unlock';
    var params2 = {
        topic: event.notify,
        payload:JSON.stringify({'image':event.image, 'command':command}),
        qos:0
    };
    console.log(params2);
    var res2 = await iotdata.publish(params2).promise();

    return res2;
};

```

#doorCamera1.js

```

var awsIot = require('aws-iot-device-sdk');
var doorCamera = awsIot.device({
    keyPath: "./credentials/camera/c786861c38-private.pem.key",
    certPath: "./credentials/camera/c786861c38-certificate.pem.crt",
    caPath: "./credentials/camera/AmazonRootCA1.pem",
    clientId:"doorCamera1",
    host:"a19tksa9tfqz8n-ats.iot.ap-northeast-2.amazonaws.com"
});
doorCamera.on('connect',function(){
    console.log('doorCamera connected');
    var imgStr = ['Ava.jpeg','Siri.jpeg','Emma.jpeg'];

    var idx = Math.floor(Math.random()*3);
    setInterval(function(){
        var message={'notify':'faceRecog/notify/door1','image':imgStr[idx]};
        doorCamera.publish('faceRecog/request',JSON.stringify(message));
        console.log('publish to faceRecog/request'+JSON.stringify(message));
    },6000);
});

```

#doorLock1.js

```
var awsIot = require('aws-iot-device-sdk');
var doorLock = awsIot.device({
  keyPath: "./credentials/lock/cebb794eea-private.pem.key",
  certPath: "./credentials/lock/cebb794eea-
certificate.pem.crt",
  caPath: "./credentials/lock/AmazonRootCA1.pem",
  clientId:"doorLock1",
  host:"a19tksa9tfqz8n-ats.iot.ap-northeast-2.amazonaws.com"
});
doorLock.on('connect',function(){
  console.log('doorLock connected');
  doorLock.subscribe('faceRecog/notify/door1',function(){
    console.log('subscribe to the topic faceRecog/notify/door1
');
  });

  doorLock.on('message',function(topic,message){
    if(topic == 'faceRecog/notify/door1'){
      var noti=JSON.parse(message.toString());
      if(noti.command == 'unlock') console.log(noti.image,': u
nlock door1');
      else console.log(noti.image, ': unauthenticated person');
    }
  })
});
```