REPORT Final Report

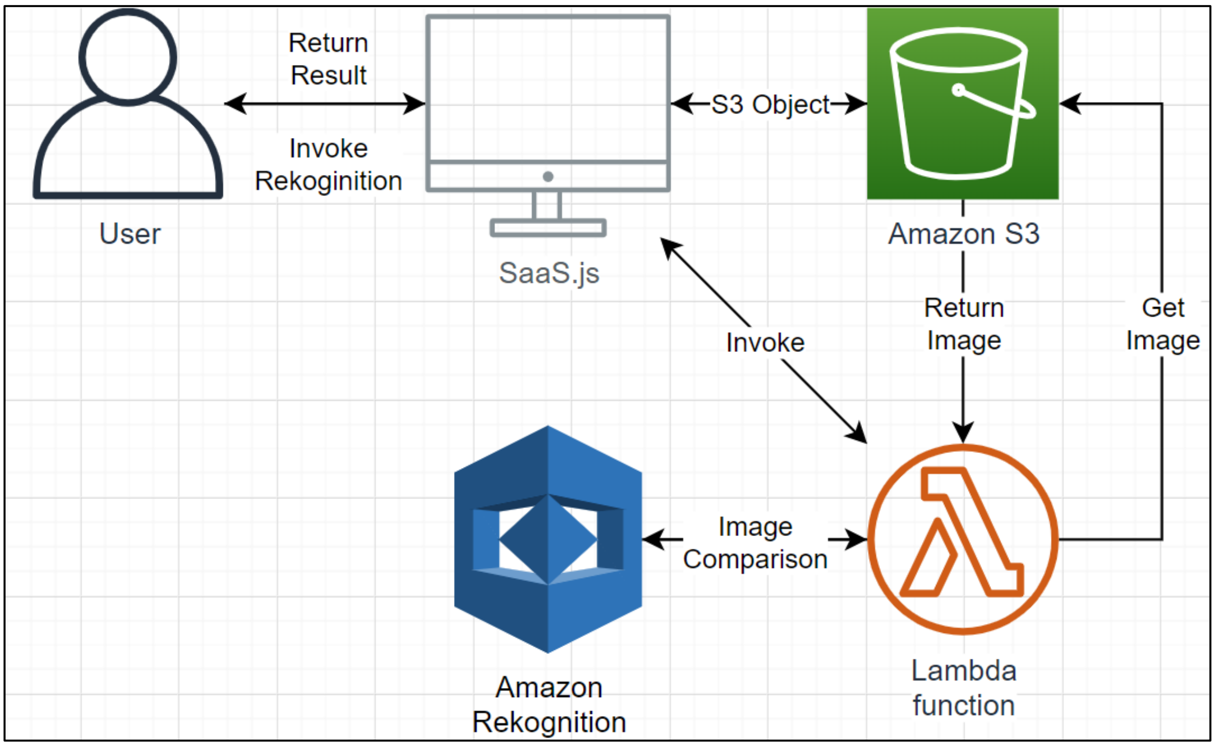
과목명	클라우드IOT서비스
담당교수	정 갑 주 교수님
학과	컴퓨터공학부
학년	4학년
학번	201714151
이름	박 민 기
제출일	2020. 06. 15

Contents

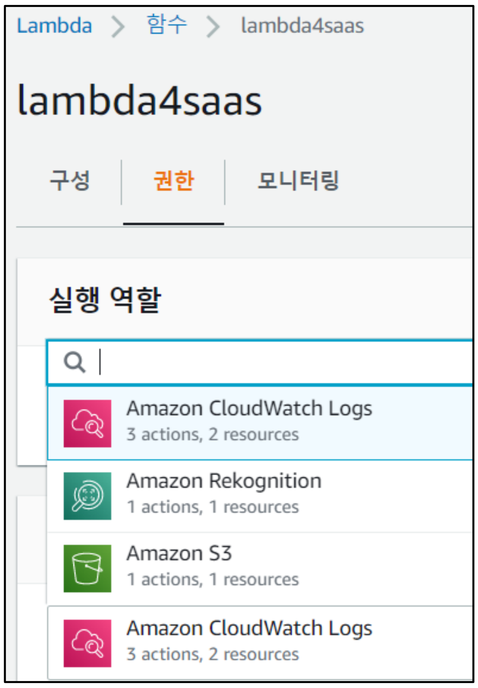
1. Task #1

1-1. System Design Document

1-1-1.Diagram



1-1-2. Give Permission to Lambda Function


Lambda > 함수 > lambda4saas

lambda4saas

구성 | 권한 | 모니터링

실행 역할

Amazon CloudWatch Logs
3 actions, 2 resources

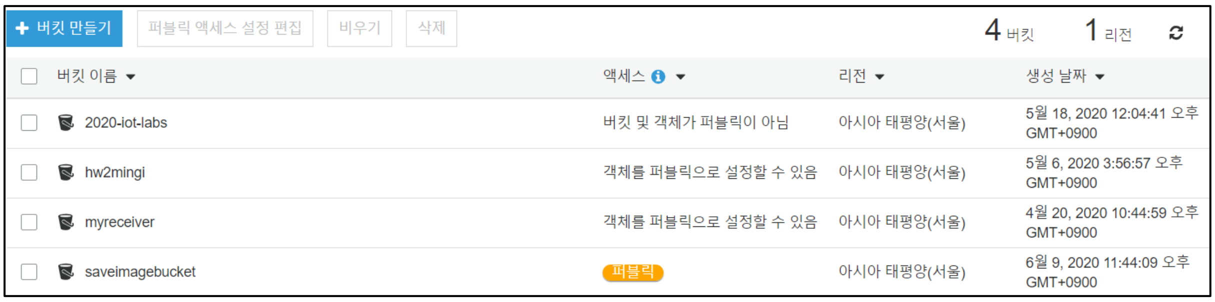
Amazon Rekognition
1 actions, 1 resources

Amazon S3
1 actions, 1 resources

Amazon CloudWatch Logs
3 actions, 2 resources

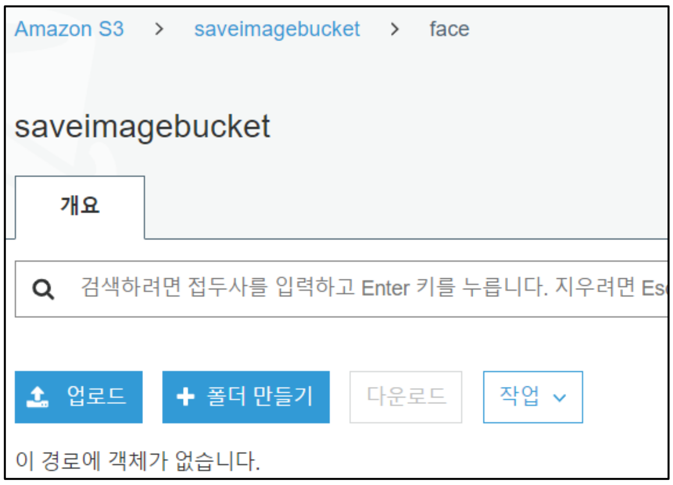
1-2. System Implementation Document

1-2-1. Create S3 bucket for S3 objects(saveimagebucket)



버킷 이름	액세스	리전	생성 날짜
2020-iot-labs	버킷 및 객체가 퍼블릭이 아님	아시아 태평양(서울)	5월 18, 2020 12:04:41 오후 GMT+0900
hw2mingi	객체를 퍼블릭으로 설정할 수 있음	아시아 태평양(서울)	5월 6, 2020 3:56:57 오후 GMT+0900
myreceiver	객체를 퍼블릭으로 설정할 수 있음	아시아 태평양(서울)	4월 20, 2020 10:44:59 오후 GMT+0900
saveimagebucket	퍼블릭	아시아 태평양(서울)	6월 9, 2020 11:44:09 오후 GMT+0900

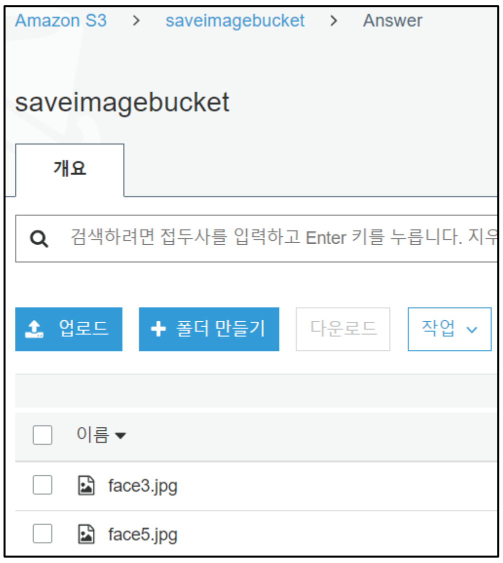
1-2-2. Create S3 folder for Upload visiter's image file



Amazon S3 > saveimagebucket > face
saveimagebucket
개요
검색하려면 접두사를 입력하고 Enter 키를 누릅니다. 지우려면 Es
업로드
+ 폴더 만들기
다운로드
작업

이 경로에 객체가 없습니다.

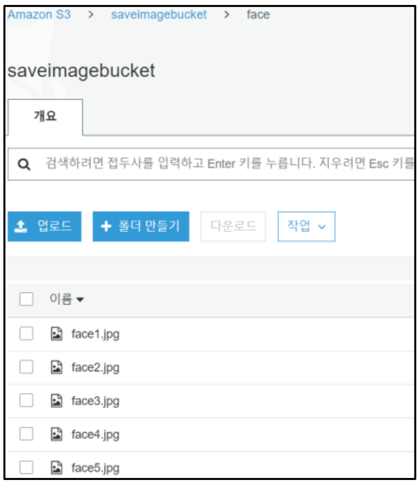
1-2-3. Create S3 objects for Registered Image(Answer/face3.jpg & Answer/face5.jpg)



Amazon S3 > saveimagebucket > Answer
saveimagebucket
개요
검색하려면 접두사를 입력하고 Enter 키를 누릅니다. 지우
업로드
+ 폴더 만들기
다운로드
작업
이름

face3.jpg
face5.jpg

1-2-4. Upload Complete Visiter's Image(S3 object in S3 bucket)



Amazon S3 > saveimagebucket > face

saveimagebucket

검색하려면 접두사를 입력하고 Enter 키를 누릅니다. 지우려면 Esc 키를

face1.jpg
face2.jpg
face3.jpg
face4.jpg
face5.jpg

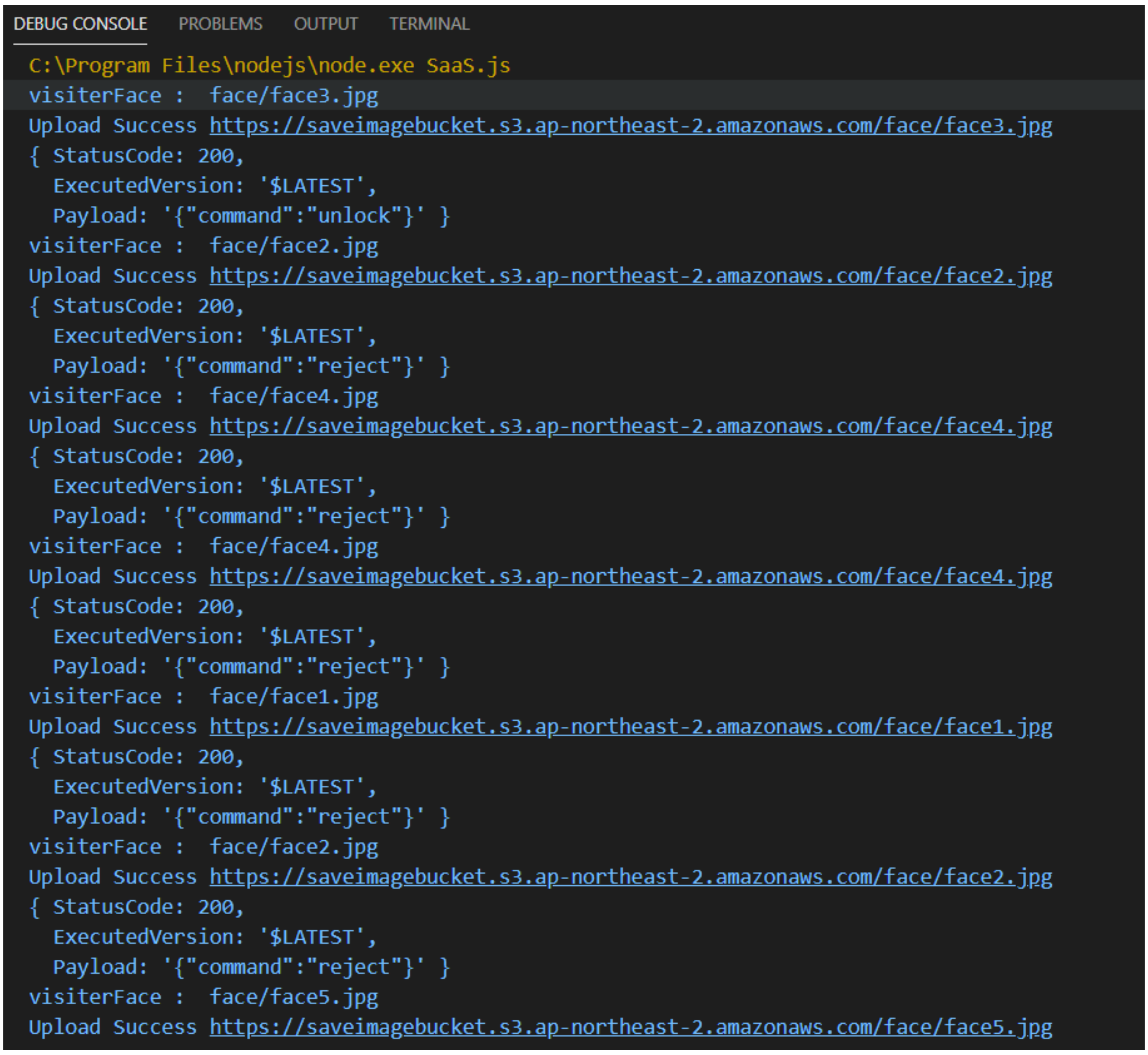
1-3. SaaS.js Node js code

```
var AWS = require('aws-sdk');
AWS.config.region = 'ap-northeast-2';
var lambda = new AWS.Lambda ({"apiVersion" : '2015-03-31'});

AWS.config.update({region:'ap-northeast-2'});
s3 = new AWS.S3({apiVersion: '2006-03-01'});


function imageRekognitionFunc(visiterFace){

    var bucket = 'saveimagebucket';
    var uploadParams = {Bucket: bucket, Key: '', Body: ''};
    var localImageFolder = './face/';

    var fs = require('fs');
    var visiterFolder = 'face/';
    var file = localImageFolder + visiterFace;
    var visiter = visiterFolder + visiterFace;
    var fileStream = fs.createReadStream(file);

    fileStream.on('error', function(err) {
      console.log('File Error', err);
    });

    uploadParams.Body = fileStream;
    var path = require('path');
    uploadParams.Key = visiterFolder + path.basename(file);
    s3.upload (uploadParams, function (err, data) {
      if (err) {
        console.log("Error", err);
      } if (data) {
        console.log("Upload Success", data.Location);
      }
    });

    console.log("visiterFace : ", visiter)

    const event = { visiterFace : visiter, bucket : bucket};

    lambda.invoke(
        {
            FunctionName: 'lambda4saas',
            Payload: JSON.stringify(event, null, 2)
        },
        function(error, data) {
            if (error) { console.info(error);}
            else { console.info(data);}
        }
    );

}

setInterval(function () {
    var images = ['face1.jpg','face2.jpg','face3.jpg','face4.jpg','face5.jpg']
;
    var randomImage = images[Math.ceil(Math.random()*5)-1];
    imageRekognitionFunc(randomImage);

}, 3000);
```

1-4. Lambda index.js Code

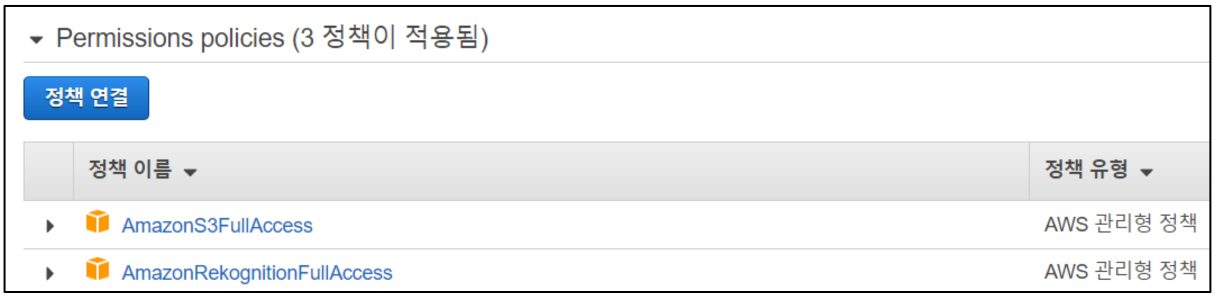
```
var AWS = require('aws-sdk');
AWS.config.update({region : 'ap-northeast-2'});

var imageRecog = new AWS.Rekognition();

exports.handler = async function (event, context){

    var bucket = 'saveimagebucket';
    var answer = false;
    var answerImage = ['Answer/face3.jpg', 'Answer/face5.jpg'];

    var visiterImage = event.visiterFace;
    var command = "";

    for(var i=0; i<answerImage.length; i++) {

        var Params = {
            "QualityFilter" : "AUTO",
            "SimilarityThreshold": 70,

        "SourceImage": {
            "S3Object": {
                "Bucket": bucket,
                "Name" : answerImage[i]
            }
        },

        "TargetImage":  {
            "S3Object": {
                "Bucket": event.bucket,
                "Name" : visiterImage
            }
        }
        };

        var result = await imageRecog.compareFaces(Params, function(err,data)
{
            if (err) console.log(err, err.stack);
            else { answer = Boolean(data.FaceMatches.length); }
        }).promise();

        command = (answer) ? 'unlock' : 'reject';

        if (answer) {
            //console.log(command);
            return {"command" : command };
        }
    }

    console.log(command);
    return {"command" : command };
};
```

1-5. SnapShots Debug Console(result)



```
C:\Program Files\nodejs\node.exe SaaS.js
visiterFace :  face/face3.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face3.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"unlock"}' }
visiterFace :  face/face2.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face2.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"reject"}' }
visiterFace :  face/face4.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face4.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"reject"}' }
visiterFace :  face/face4.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face4.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"reject"}' }
visiterFace :  face/face1.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face1.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"reject"}' }
visiterFace :  face/face2.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face2.jpg
{ StatusCode: 200,
  ExecutedVersion: '$LATEST',
  Payload: '{"command":"reject"}' }
visiterFace :  face/face5.jpg
Upload Success https://saveimagebucket.s3.ap-northeast-2.amazonaws.com/face/face5.jpg
```

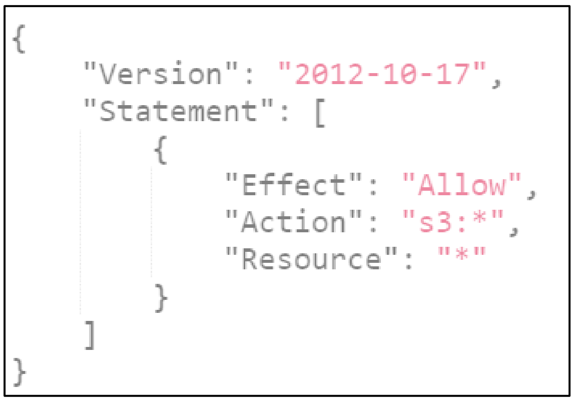
1-6. Permission JSON Documentation



Permissions policies (3 정책이 적용됨)

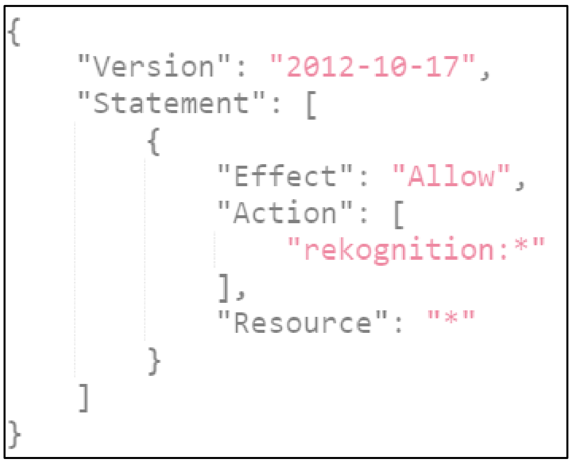
정책 연결

정책 이름	정책 유형
AmazonS3FullAccess	AWS 관리형 정책
AmazonRekognitionFullAccess	AWS 관리형 정책



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

-> AmazonS3FullAccess



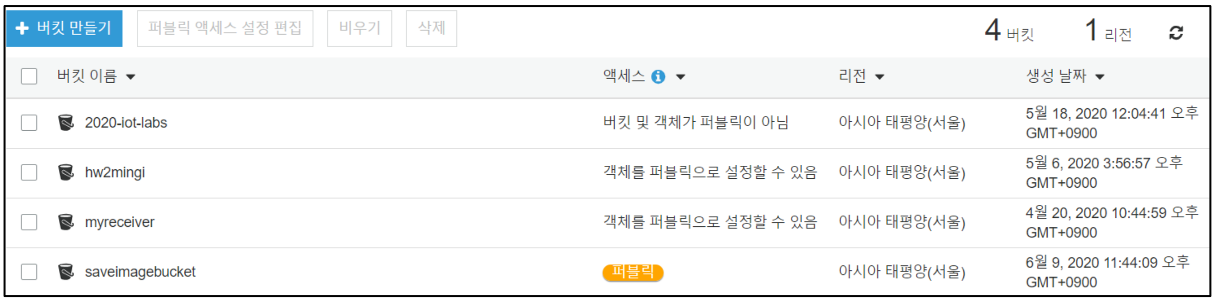
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        }
    ]
}
```

-> AmazonRekognitionFullAccess

2. Task #2

2-1. System Design Document

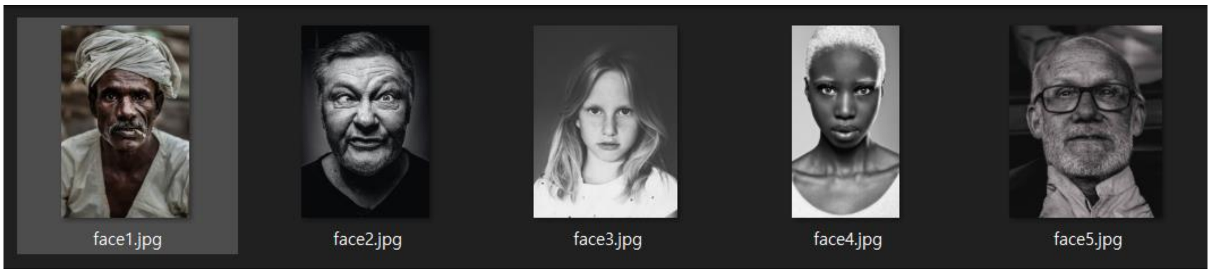
2-1-1. Create S3 bucket for S3 objects(saveimagebucket)

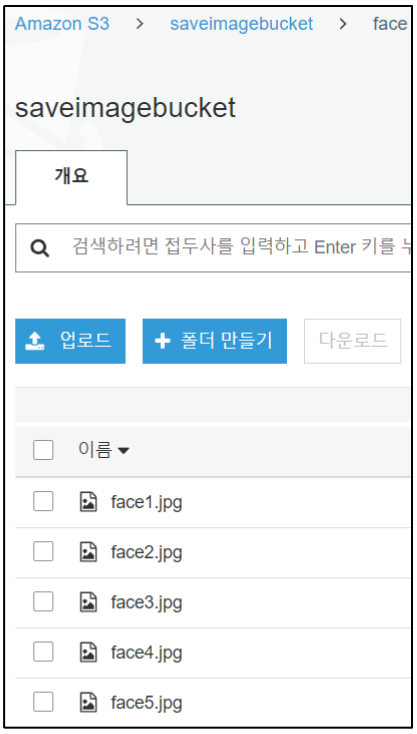


+ 버킷 만들기 | 퍼블릭 액세스 설정 편집 | 비우기 | 삭제 | 4 버킷 | 1 리전

버킷 이름	액세스	리전	생성 날짜
2020-iot-labs	버킷 및 객체가 퍼블릭이 아님	아시아 태평양(서울)	5월 18, 2020 12:04:41 오후 GMT+0900
hw2mingi	객체를 퍼블릭으로 설정할 수 있음	아시아 태평양(서울)	5월 6, 2020 3:56:57 오후 GMT+0900
myreceiver	객체를 퍼블릭으로 설정할 수 있음	아시아 태평양(서울)	4월 20, 2020 10:44:59 오후 GMT+0900
saveimagebucket	퍼블릭	아시아 태평양(서울)	6월 9, 2020 11:44:09 오후 GMT+0900

2-1-2. Create S3 objects for doorCamera1 (Image file to be confirmed: face/face1~5.jpg)





Amazon S3 > saveimagebucket > face

saveimagebucket

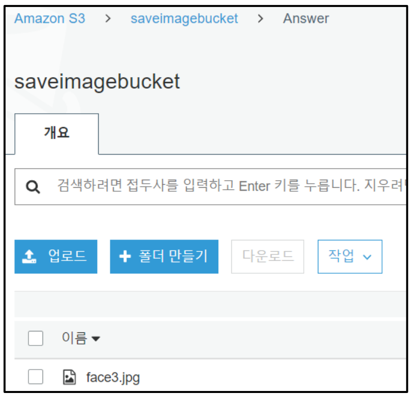
개요

검색하려면 접두사를 입력하고 Enter 키를

업로드 | + 폴더 만들기 | 다운로드

이름
face1.jpg
face2.jpg
face3.jpg
face4.jpg
face5.jpg

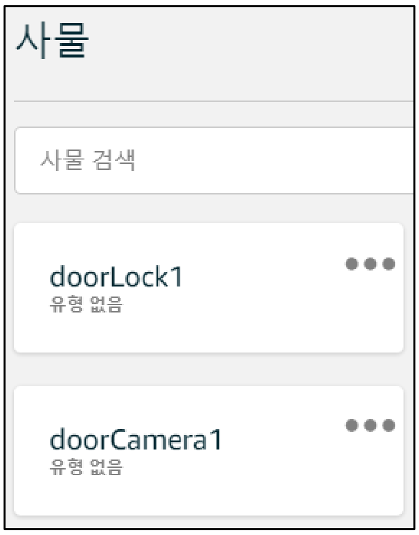
2-1-3. Create S3 objects for Registered Image(Answer/face3.jpg)


Amazon S3 > saveimagebucket > Answer

saveimagebucket

face3.jpg

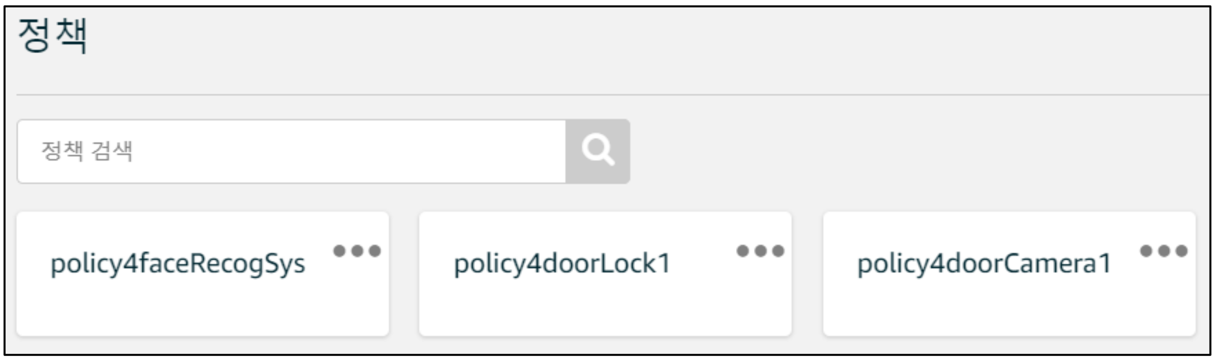
2-1-4. Create IoT Device(doorCamera1 & doorLock1)



doorLock1
유형 없음

doorCamera1
유형 없음

2-1-5. Create Policy for IoT Device

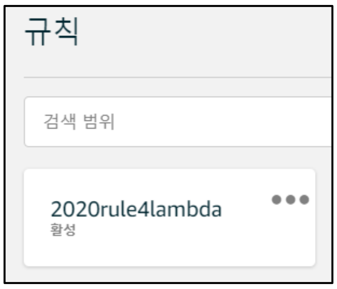


policy4faceRecogSys

policy4doorLock1

policy4doorCamera1

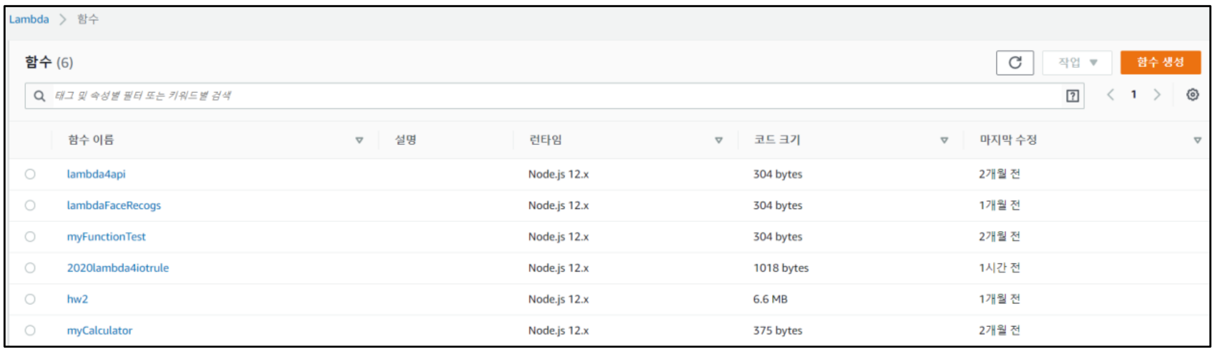
2-1-6. Create IoT Rule for Invoke Lambda


규칙

검색 범위

2020rule4lambda
활성

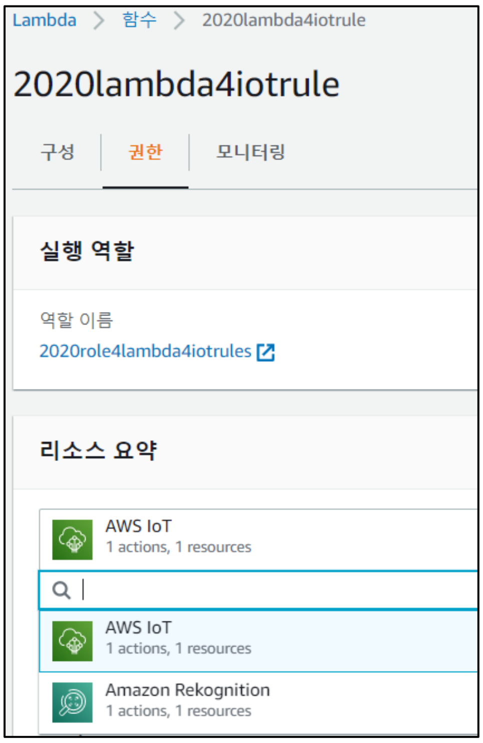
2-1-7. Create Lambda(For Receive message & Image Rekognition)


Lambda > 함수

함수 (6)

함수 이름	설명	런타임	코드 크기	마지막 수정
lambda4api		Node.js 12.x	304 bytes	2개월 전
lambdaFaceRecogs		Node.js 12.x	304 bytes	1개월 전
myFunctionTest		Node.js 12.x	304 bytes	2개월 전
2020lambda4iotrule		Node.js 12.x	1018 bytes	1시간 전
hw2		Node.js 12.x	6.6 MB	1개월 전
myCalculator		Node.js 12.x	375 bytes	2개월 전

2-1-8. Give Permission to Lambda


Lambda > 함수 > 2020lambda4iotrule

2020lambda4iotrule

실행 역할

역할 이름
2020role4lambda4iotrules

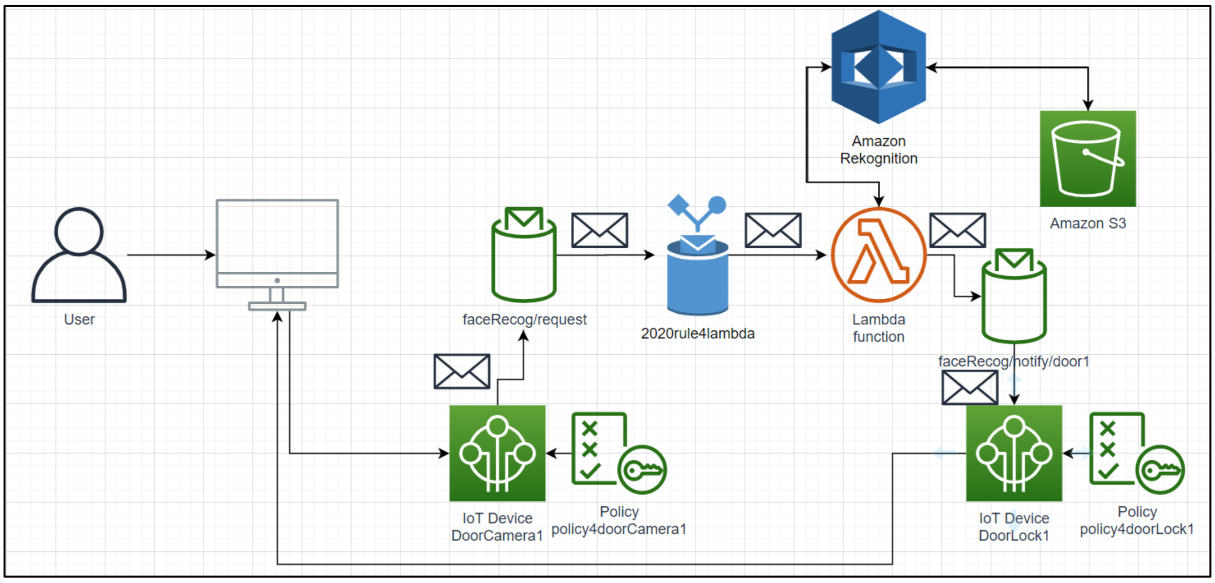
리소스 요약

AWS IoT
1 actions, 1 resources

AWS IoT
1 actions, 1 resources

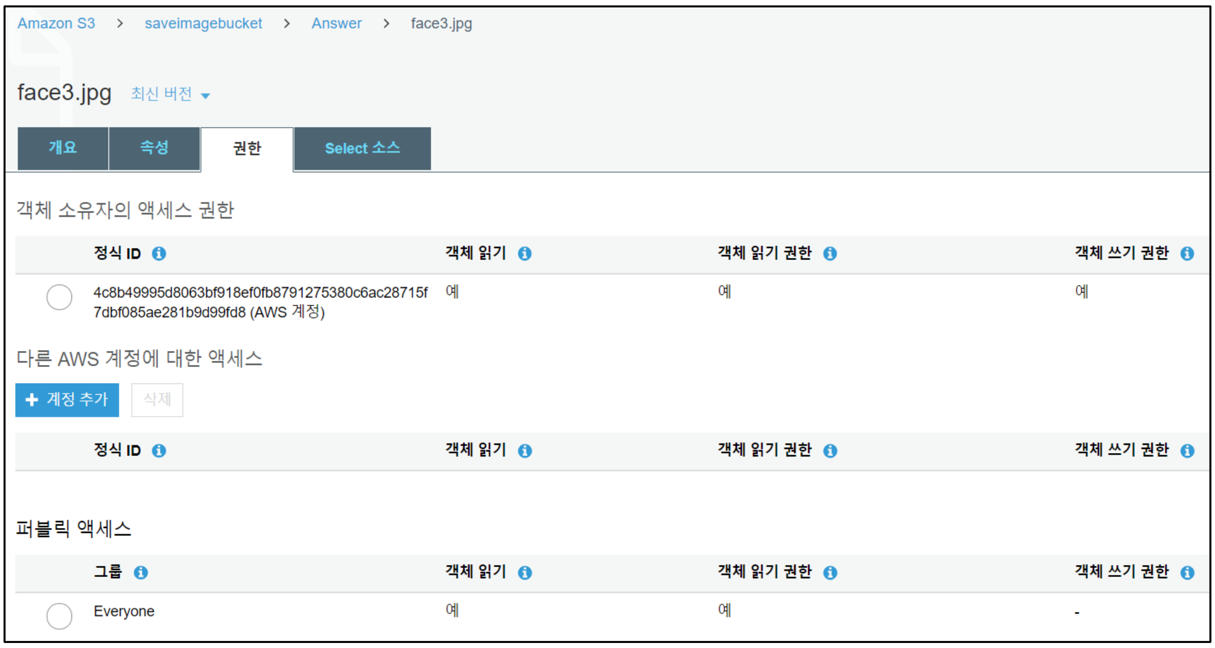
Amazon Rekognition
1 actions, 1 resources

2-1-9. Design Diagram



2-2. System Implementation Document

2-2-1. S3 objects to public for access in Lambda



Amazon S3 > saveimagebucket > Answer > face3.jpg

face3.jpg 최신 버전

객체 소유자의 액세스 권한

정식 ID	객체 읽기	객체 읽기 권한	객체 쓰기 권한
4c8b49995d8063bf918ef0fb8791275380c6ac28715f7dbf085ae281b9d99fd8 (AWS 계정)	예	예	예

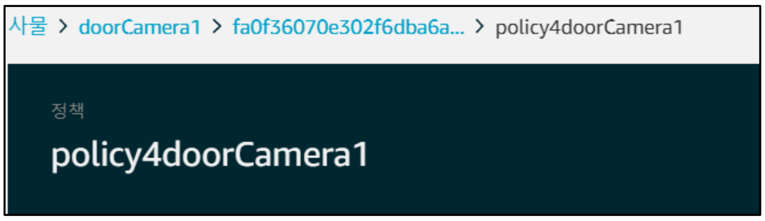
다른 AWS 계정에 대한 액세스

정식 ID	객체 읽기	객체 읽기 권한	객체 쓰기 권한

퍼블릭 액세스

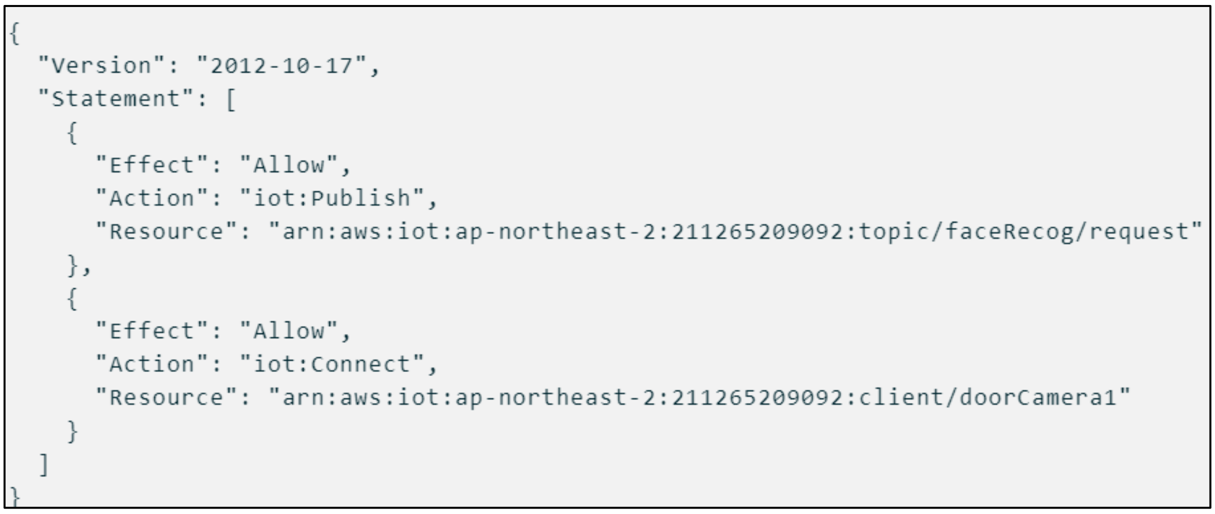
그룹	객체 읽기	객체 읽기 권한	객체 쓰기 권한
Everyone	예	예	-

2-2-2. policy to doorCamera1(Connect & Publish)

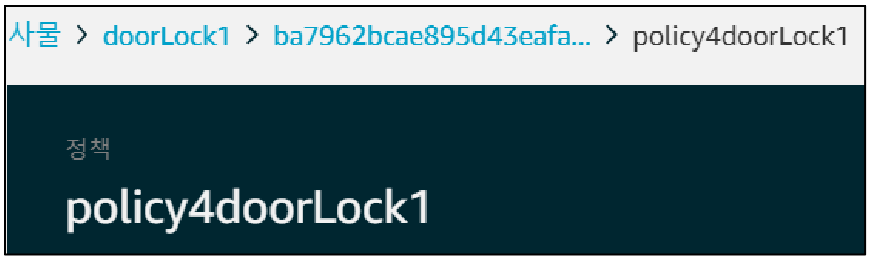

사물 > doorCamera1 > fa0f36070e302f6dba6a... > policy4doorCamera1

정책

policy4doorCamera1

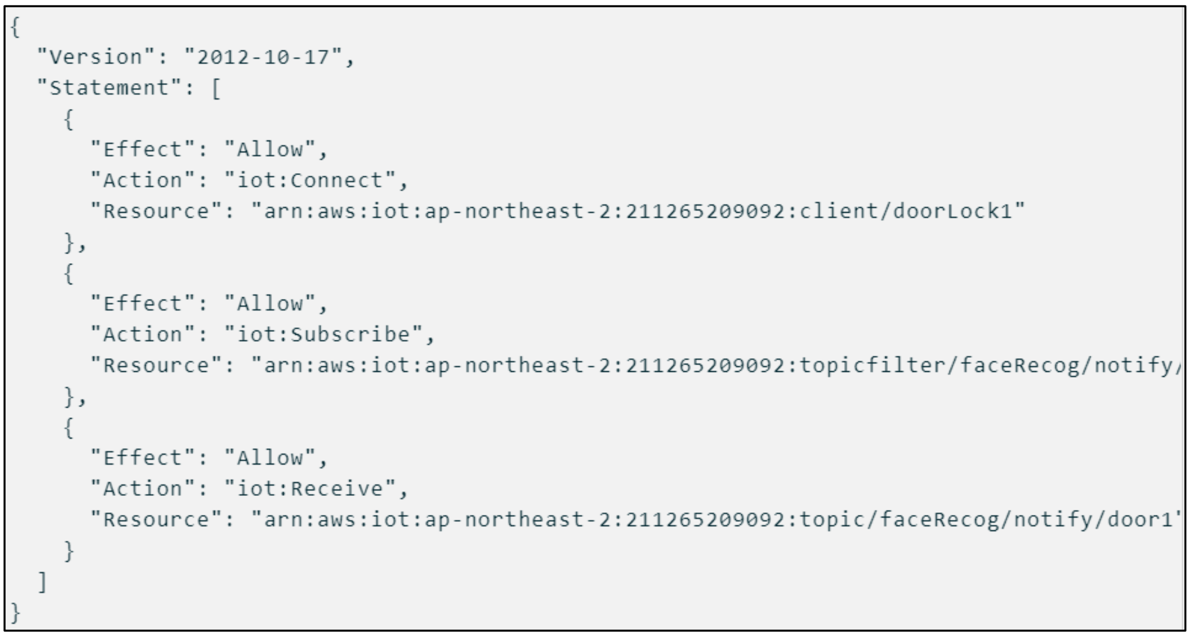

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/faceRecog/request"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:client/doorCamera1"
    }
  ]
}
```

2-2-3. policy to doorLock1(Connect & Subscribe & Receive)

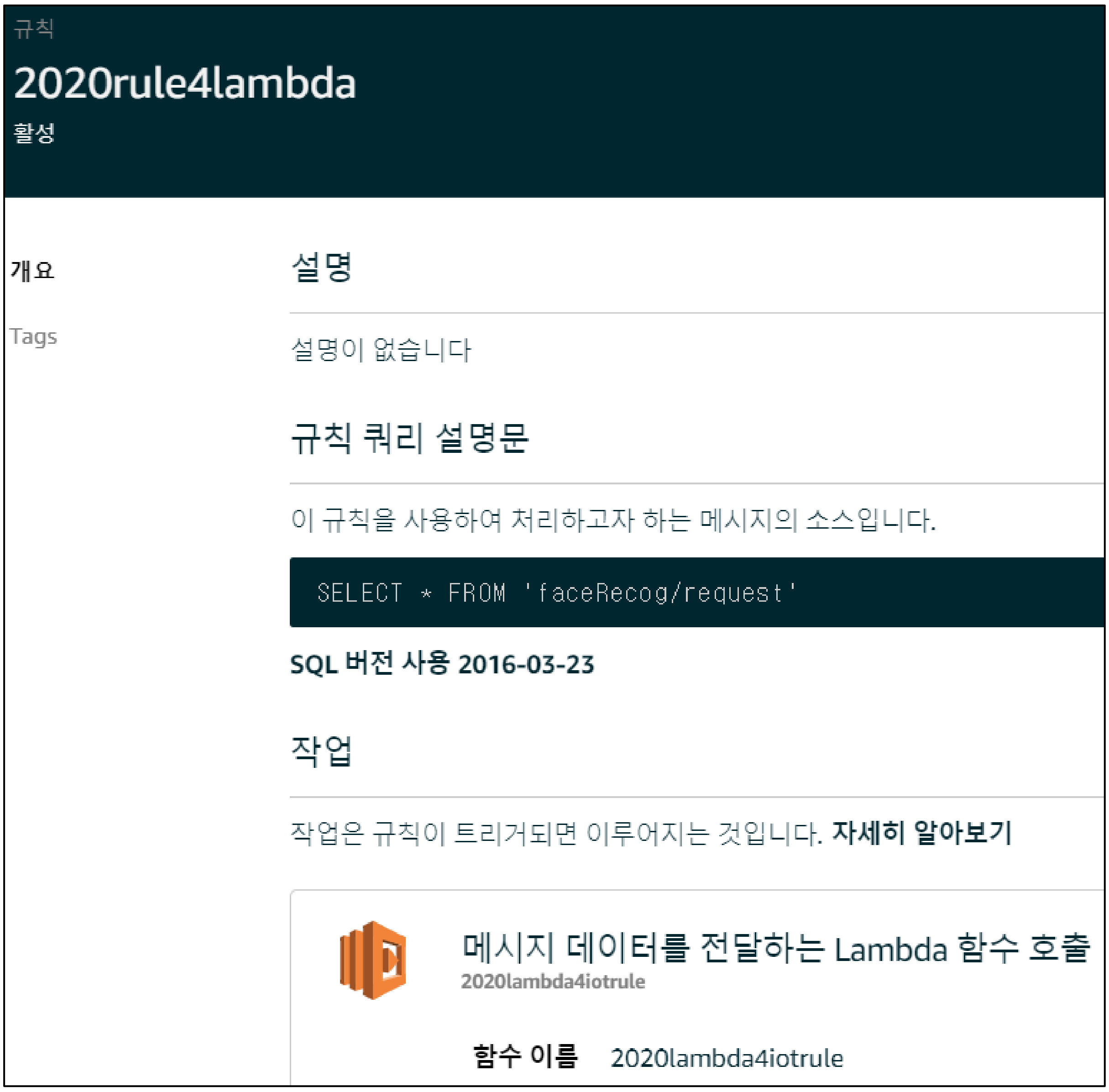

사물 > doorLock1 > ba7962bcae895d43eafa... > policy4doorLock1

정책

policy4doorLock1


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:client/doorLock1"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topicfilter/faceRecog/notify/
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:ap-northeast-2:211265209092:topic/faceRecog/notify/door1
    }
  ]
}
```

2-2-4. IoT Rule(Invoke Lambda : 2020lambda4iotrule)


규칙

2020rule4lambda

활성

개요

Tags

설명

설명이 없습니다

규칙 쿼리 설명문

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

2020lambda4iotrule

함수 이름 2020lambda4iotrule

2-2-5. Lambda (write index.js)

Attached Seprately (2-4-3)

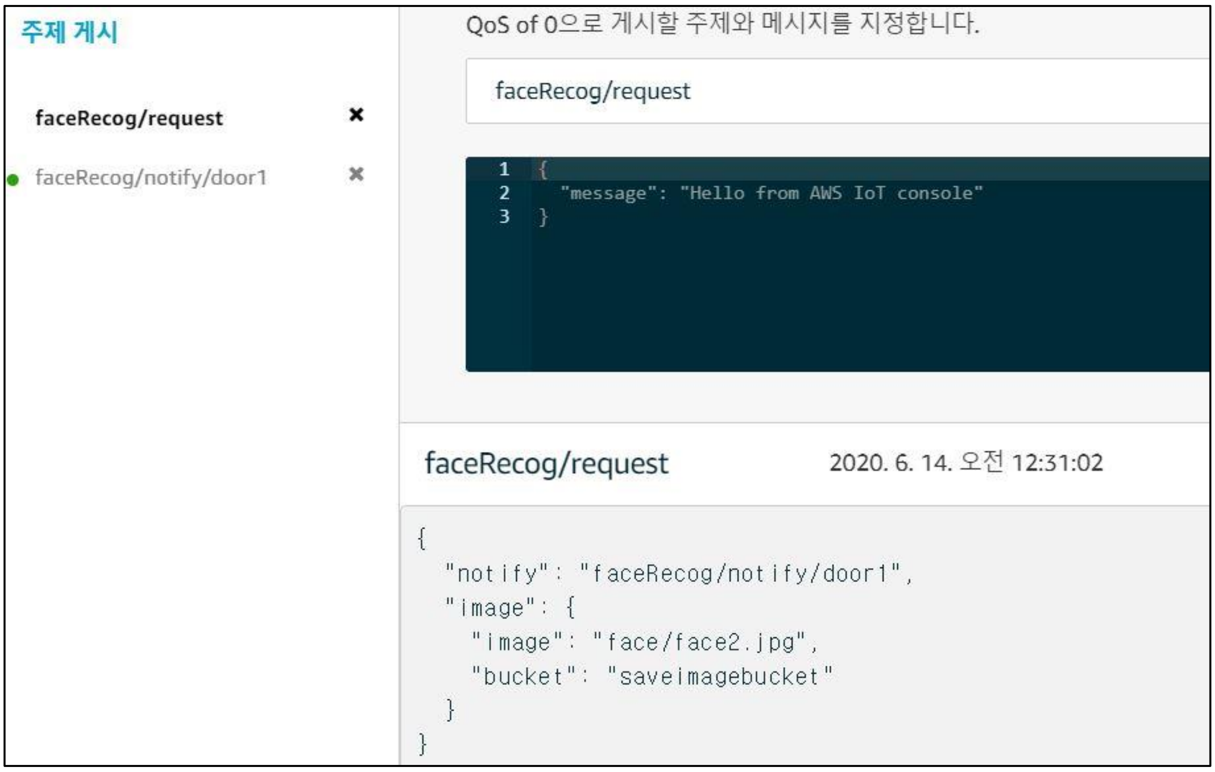
2-2-6. implement awsiot-doorCamera1.js

Attached Seprately (2-3)

2-2-7. implement awsiot-doorLock1.js

Attached Seprately (2-3)

2-2-8. check in the IoT Core Test



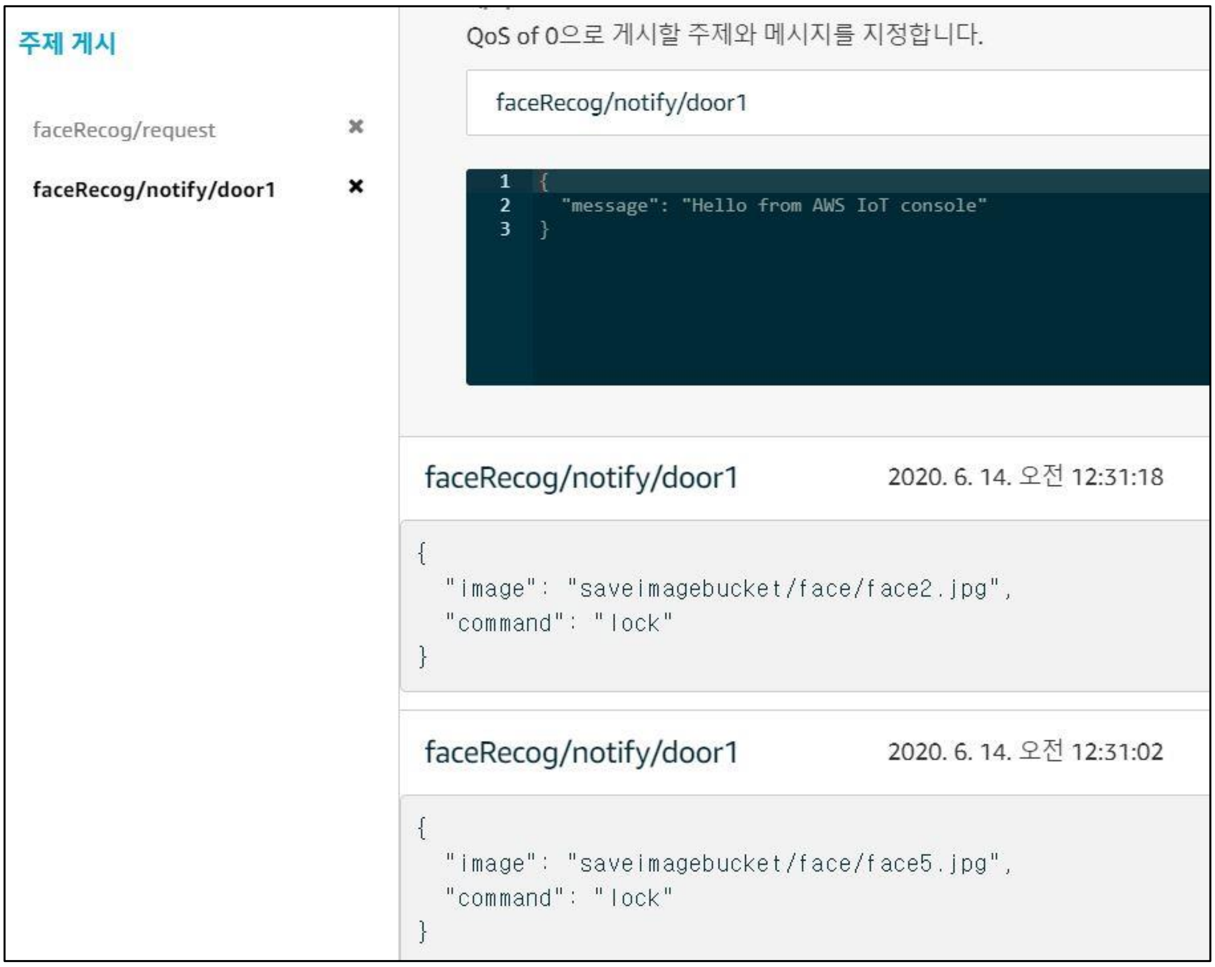
QoS of 0으로 게시할 주제와 메시지를 지정합니다.

faceRecog/request

```
{
  "message": "Hello from AWS IoT console"
}
```

faceRecog/request 2020. 6. 14. 오전 12:31:02

```
{
  "notify": "faceRecog/notify/door1",
  "image": {
    "image": "face/face2.jpg",
    "bucket": "saveimagebucket"
  }
}
```



QoS of 0으로 게시할 주제와 메시지를 지정합니다.

faceRecog/notify/door1

```
{
  "message": "Hello from AWS IoT console"
}
```

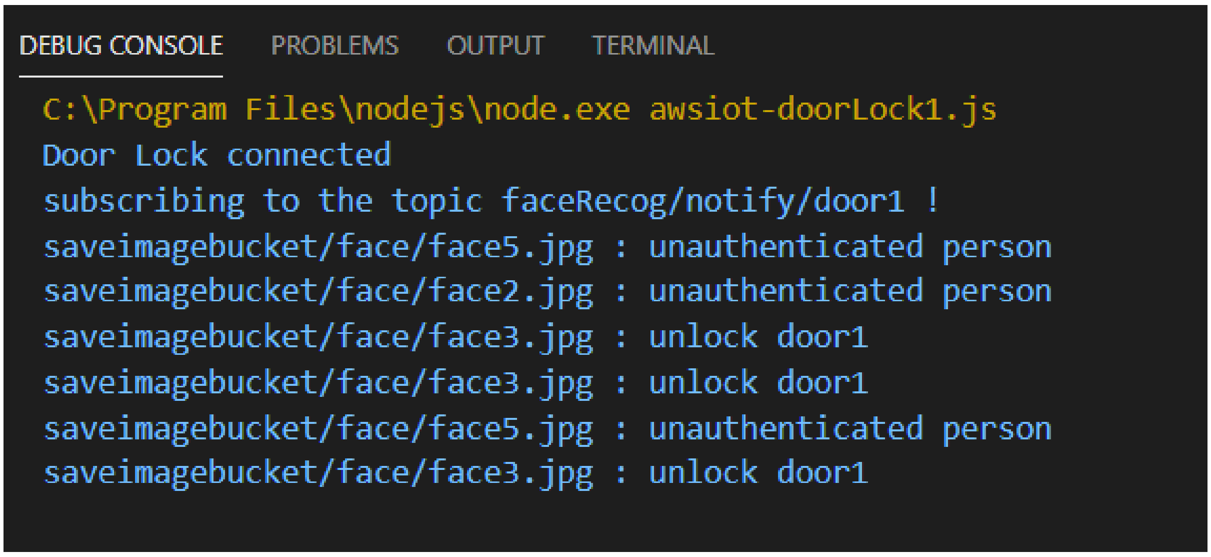
faceRecog/notify/door1 2020. 6. 14. 오전 12:31:18

```
{
  "image": "saveimagebucket/face/face2.jpg",
  "command": "lock"
}
```

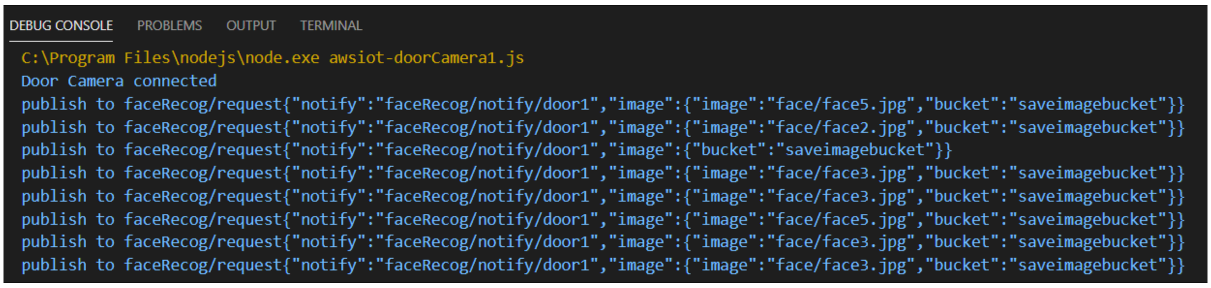
faceRecog/notify/door1 2020. 6. 14. 오전 12:31:02

```
{
  "image": "saveimagebucket/face/face5.jpg",
  "command": "lock"
}
```

2-3. Screen SnapShots (Implemenatation Task)



```
C:\Program Files\nodejs\node.exe awsiot-doorLock1.js
Door Lock connected
subscribing to the topic faceRecog/notify/door1 !
saveimagebucket/face/face5.jpg : unauthenticated person
saveimagebucket/face/face2.jpg : unauthenticated person
saveimagebucket/face/face3.jpg : unlock door1
saveimagebucket/face/face3.jpg : unlock door1
saveimagebucket/face/face5.jpg : unauthenticated person
saveimagebucket/face/face3.jpg : unlock door1
```



```
C:\Program Files\nodejs\node.exe awsiot-doorCamera1.js
Door Camera connected
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face5.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face2.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face3.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face3.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face5.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face3.jpg","bucket":"saveimagebucket"}}
publish to faceRecog/request{"notify":"faceRecog/notify/door1","image":{"image":"face/face3.jpg","bucket":"saveimagebucket"}}
```

2-4. Node.js Code

2-4-1.awsiot-doorCamera1.js

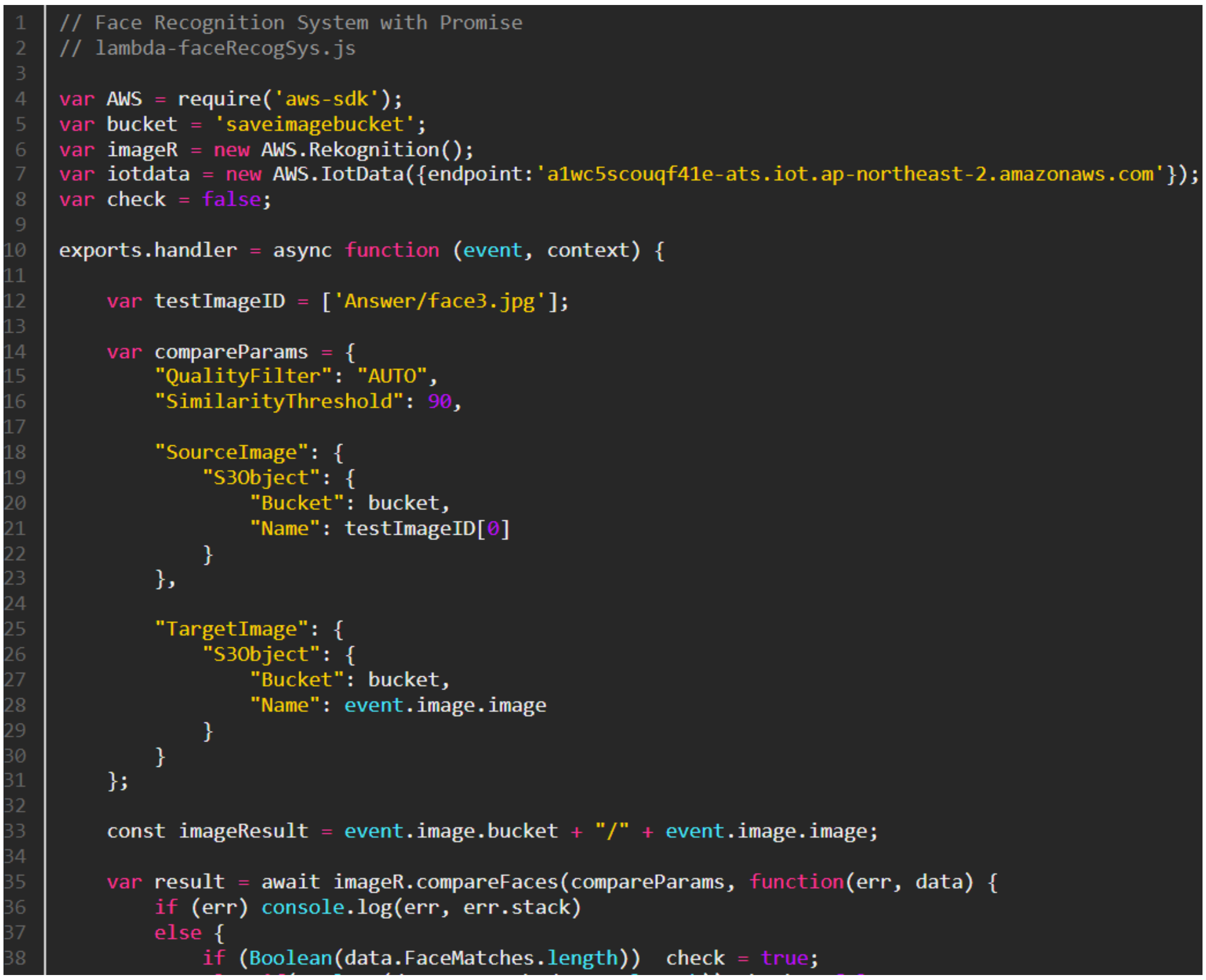
```javascript
// Door Camera Device Example
// awsiot-camera.js

var awsIot = require('aws-iot-device-sdk');

var doorCamera = awsIot.device({
  keyPath: "./credentials/Camera/fa0f36070e-private.pem.key",
  certPath: "./credentials/Camera/fa0f36070e-certificate.pem.crt",
  caPath: "./credentials/Camera/AmazonRootCA1.pem",
  clientId: "doorCamera1",
  host: "a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com"
});
```

```
// Device is an instance returned by mqtt.Client(), see mqtt.js for full docum
entation.
doorCamera.on('connect', function () {
  console.log('Door Camera connected');

var images = ['face/face1.jpg','face/face2.jpg','face/face3.jpg','face/face4.j
pg','face/face5.jpg'];
var sourceBucket = 'saveimagebucket';

  setInterval(function () {
    // randomly select one of the five images
    var imageParam = {
      image : images[Math.ceil(Math.random()*5)],
      bucket : sourceBucket
    }

    var message = { 'notify': 'faceRecog/notify/door1', 'image': imageParam };
    console.log('publish to faceRecog/request' + JSON.stringify(message));
    doorCamera.publish('faceRecog/request', JSON.stringify(message));
  }, 15000);//15 초마다 함수 실행
});
```

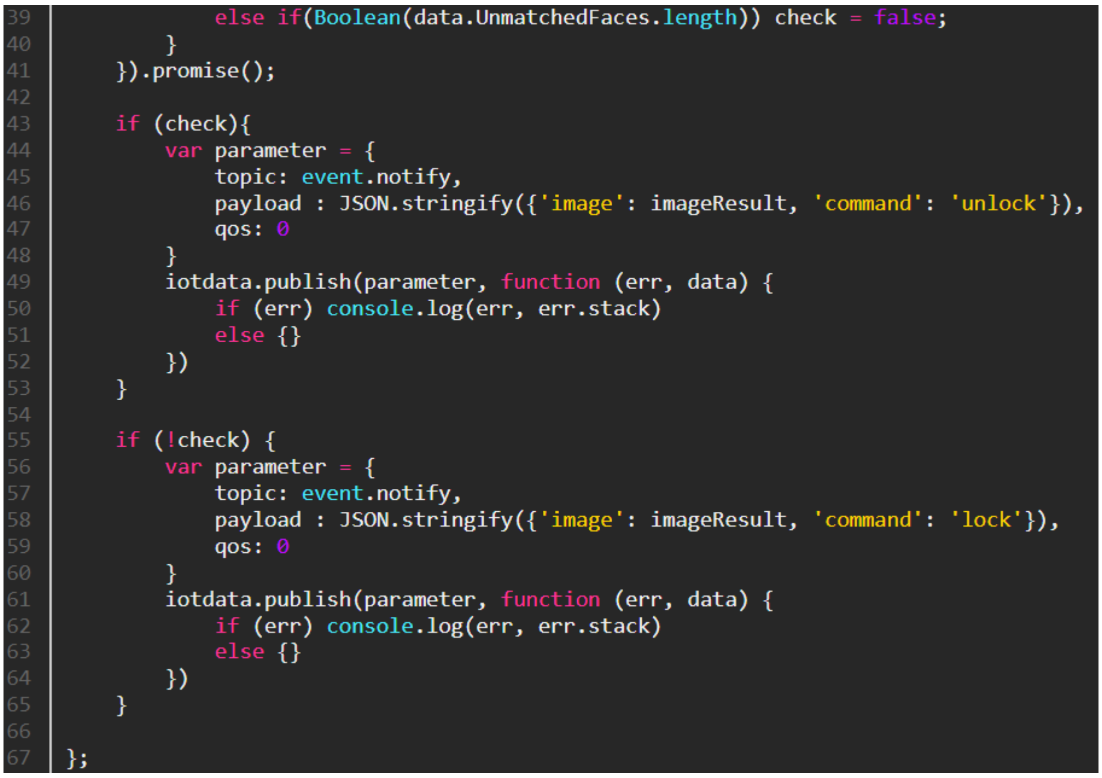
2-4-2.awsiot-doorLock1.js

```

// Door Lock Device Example
// awsiot-doorLock.js

var awsIot = require('aws-iot-device-sdk');

var doorLock = awsIot.device({
  keyPath: "./credentials/lock/ba7962bcae-private.pem.key",
  certPath: "./credentials/lock/ba7962bcae-certificate.pem.crt",
  caPath: "./credentials/lock/AmazonRootCA1.pem",
  clientId: "doorLock1",
  host: "a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com"
});

// Device is an instance returned by mqtt.Client(), see mqtt.js for full docum
entation.
doorLock.on('connect', function () {
  console.log('Door Lock connected');
  doorLock.subscribe('faceRecog/notify/door1', function () {
    console.log('subscribing 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, ': unlock door1')
      else console.log(noti.image, ': unauthenticated person')
    }
  })
});
```

2-4-3. index.js (in Lambda)



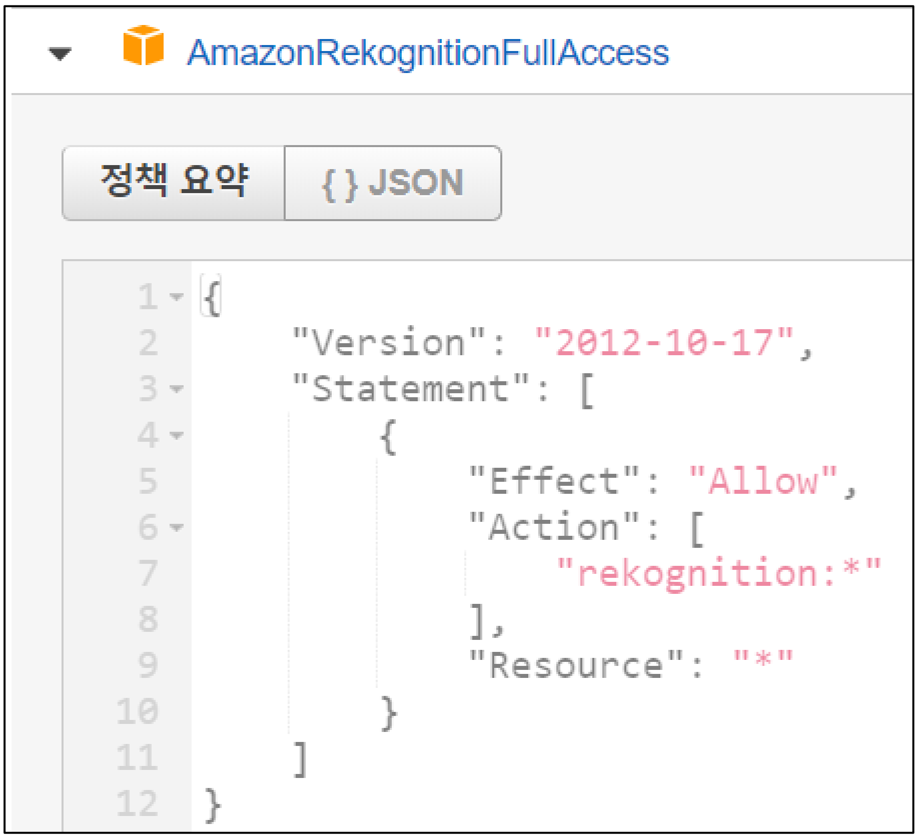
```
// Face Recognition System with Promise
// lambda-faceRecogSys.js

var AWS = require('aws-sdk');
var bucket = 'saveimagebucket';
var imageR = new AWS.Rekognition();
var iotdata = new AWS.IotData({endpoint:'a1wc5scouqf41e-ats.iot.ap-northeast-2.amazonaws.com'});
var check = false;

exports.handler = async function (event, context) {

    var testImageID = ['Answer/face3.jpg'];

    var compareParams = {
        "QualityFilter": "AUTO",
        "SimilarityThreshold": 90,

        "SourceImage": {
            "S3Object": {
                "Bucket": bucket,
                "Name": testImageID[0]
            }
        },

        "TargetImage": {
            "S3Object": {
                "Bucket": bucket,
                "Name": event.image.image
            }
        }
    };

    const imageResult = event.image.bucket + "/" + event.image.image;

    var result = await imageR.compareFaces(compareParams, function(err, data) {
        if (err) console.log(err, err.stack)
        else {
            if (Boolean(data.FaceMatches.length))  check = true;
```



```
            else if(Boolean(data.UnmatchedFaces.length)) check = false;
        }
    }).promise();

    if (check){
        var parameter = {
            topic: event.notify,
            payload : JSON.stringify({'image': imageResult, 'command': 'unlock'}),
            qos: 0
        }
        iotdata.publish(parameter, function (err, data) {
            if (err) console.log(err, err.stack)
            else {}
        })
    }

    if (!check) {
        var parameter = {
            topic: event.notify,
            payload : JSON.stringify({'image': imageResult, 'command': 'lock'}),
            qos: 0
        }
        iotdata.publish(parameter, function (err, data) {
            if (err) console.log(err, err.stack)
            else {}
        })
    }

};
```

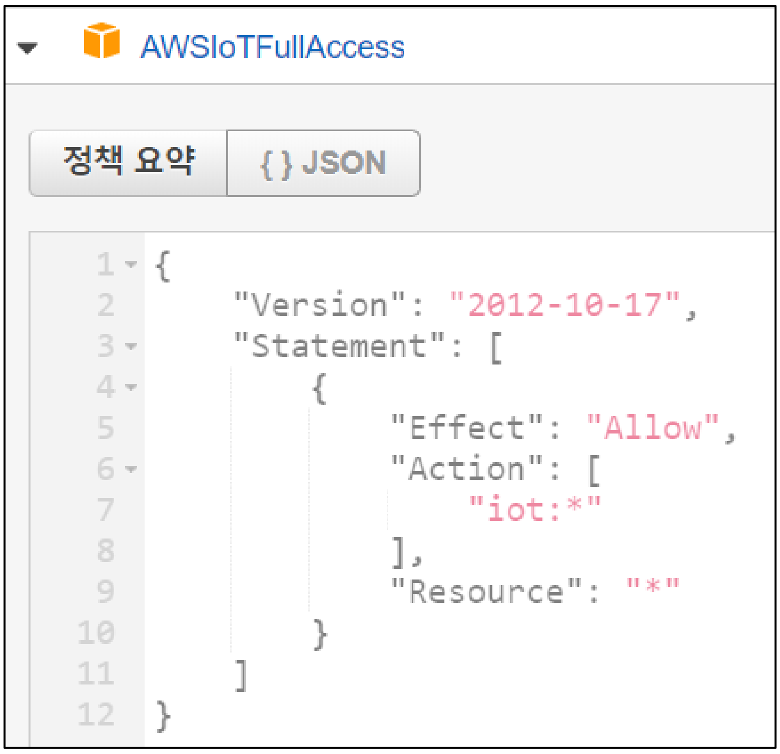
2-5. Permission JSON Documentation

2-5-1.AmazonRekognitionFullAccess



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        }
    ]
}
```

2-5-2. AWSIoTFullAccess



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:*"
            ],
            "Resource": "*"
        }
    ]
}
```

3. Why PaaS? Not SaaS?

3-1. SaaS(Software as a Service)

Software as a Service, also known as cloud application services, represents the most commonly utilized option for businesses in the cloud market. SaaS utilizes the internet to deliver applications, which are managed by a third-party vendor, to its users. A majority of SaaS applications run directly through your web browser, which means they do not require any downloads or installations on the client side.

Adavantages of SaaS

SaaS provides numerous advantages to employees and companies by greatly reducing the time and money spent on tedious tasks such as installing, managing, and upgrading software. This frees up plenty of time for technical staff to spend on more pressing matters and issues within the organization.

Characteristics of SaaS

- Managed from a central location
- Hosted on a remote server
- Accesible over the internet
- Users not responsible for hardware or software updates

3-2. PaaS(Platform as a Service)

Cloud platform services, also known as Platform as a Service (PaaS), provide cloud components to certain software while being used mainly for applications. PaaS delivers a framework for developers that they can build upon and use to create customized applications. All servers, storage, and networking can be managed by the enterprise or a third-party provider while the developers can maintain management of the applications.

Advantages of PaaS

- Simple, cost-effective development and deployment of apps

- Scalable
- High available
- Developers can customize apps without the headache of maintaining the software
- Significant reduction in the amount of coding needed
- Automaton of business policy
- Easy migration to the hybrid model

Characteristics of PaaS

- Builds on virtualization technology, so resources can easily be scaled up or down as your business changes
- Provides a variety of services to assist with the development, testing, and deployment of apps
- Accessible to numerous users via the same devlopment application
- Integrates web services and databases

3-3. Why Use PaaS

PaaS can streamline workflows when multiple developers are working on the same development project. If other vendors must be included, PaaS can provide great speed and flexibility to the entire process. PaaS is particularly beneficial if you need to create customized applications. This cloud service also can greatly reduce costs and it can simplify some challenges that come up if you are rapidly developing or deploying an app.

SaaS Limitations and Concerns

Data security Problem

Large volumes of data may have to be exchanged to the backend data centers of SaaS apps in order to perform the necessary software functionality. Transferring sensitive business information to public-cloud based SaaS service may result in compromised security and compliance in addition to significant cost for migrating large data workloads.