2021 Cloud IoT Services Assignment #3

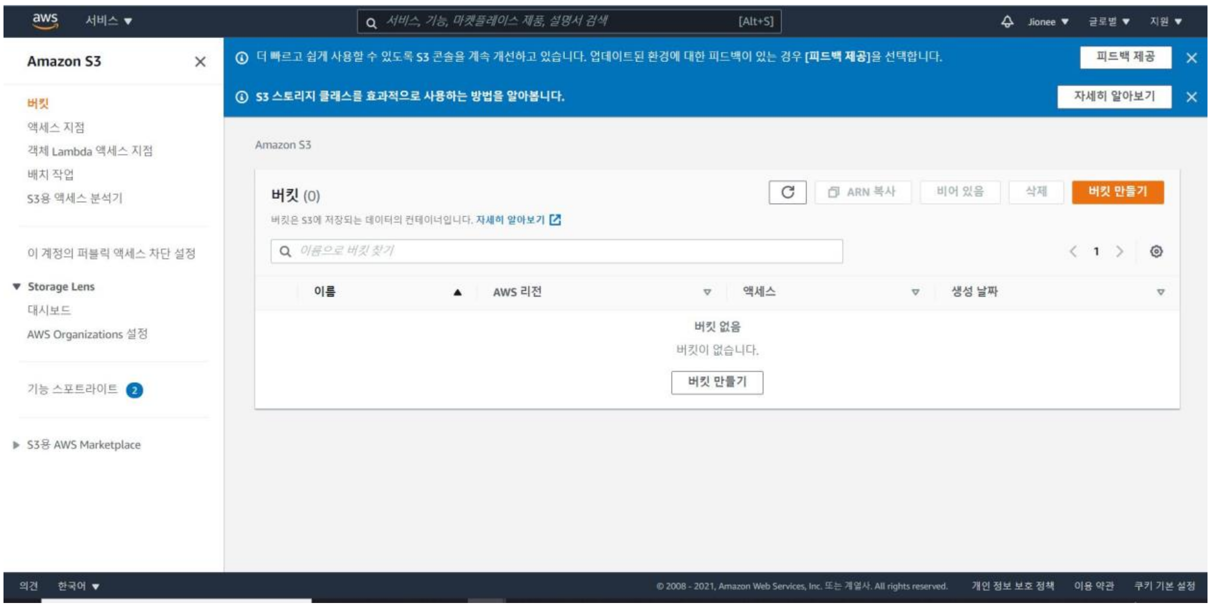
[이름 : 유지원 / 학번 : 201812627]

Task #1 : Create Applications for AWS S3 Service

1-1) Create AWS S3 Bucket and Object by AWS Web Console

- Create a S3 Bucket

> S3 > 버킷 만들기 클릭 > 이름 작성, Region 설정 후 S3 Bucket 을 생성한다.



더 빠르고 쉽게 사용할 수 있도록 S3 콘솔을 계속 개선하고 있습니다. 업데이트된 환경에 대한 피드백이 있는 경우 **[피드백 제공]**을 선택합니다.

S3 스토리지 클래스를 효과적으로 사용하는 방법을 알아봅니다.

버킷 (0)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

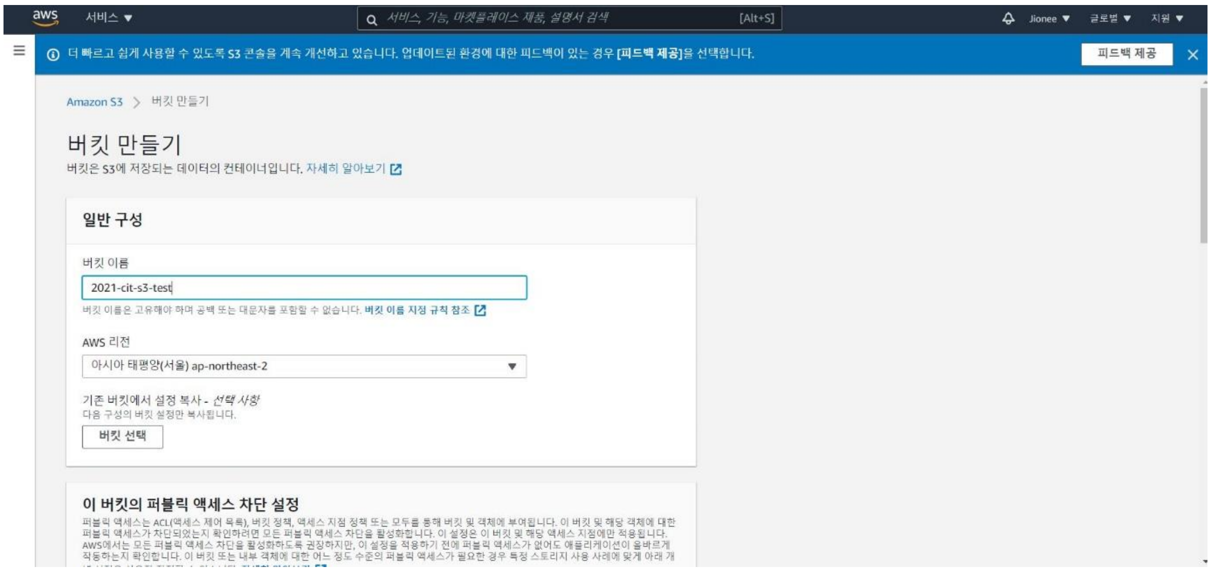
이름	AWS 리전	액세스	생성 날짜

버킷 없음

버킷이 없습니다.

버킷 만들기

© 2008 - 2021, Amazon Web Services, Inc. 또는 계열사. All rights reserved.



더 빠르고 쉽게 사용할 수 있도록 S3 콘솔을 계속 개선하고 있습니다. 업데이트된 환경에 대한 피드백이 있는 경우 **[피드백 제공]**을 선택합니다.

버킷 만들기

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

일반 구성

버킷 이름

2021-cit-s3-test

버킷 이름은 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. **버킷 이름 지정 규칙 참조**

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - *선택 사항*

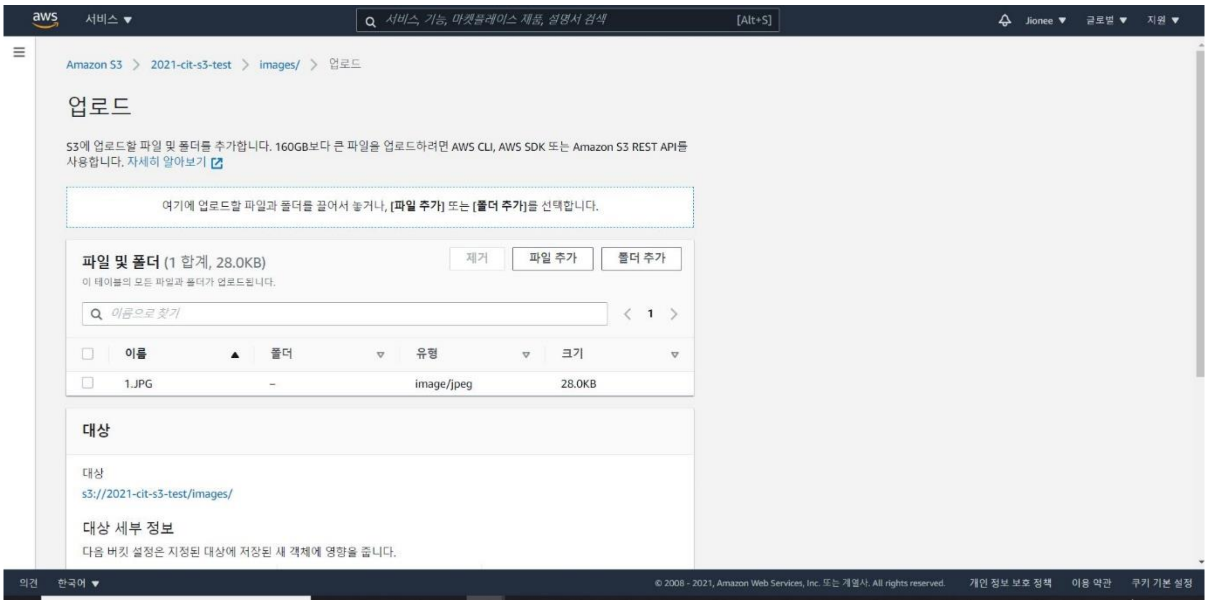
다음 구성의 버킷 설정만 복사됩니다.

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지점 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지점에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개

- Create a S3 Object in the bucket by uploading a file from your notebook through AWS Console

> 업로드 > 파일 추가



서비스
서비스, 기능, 마켓플레이스 제품, 설명서 검색 [Alt+S]
Jionee
글로벌
지원

Amazon S3 > 2021-cit-s3-test > images/ > 업로드

업로드

S3에 업로드할 파일 및 폴더를 추가합니다. 160GB보다 큰 파일을 업로드하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다. 자세히 알아보기

여기에 업로드할 파일과 폴더를 끌어서 놓거나, **[파일 추가]** 또는 **[폴더 추가]**를 선택합니다.

파일 및 폴더 (1 합계, 28.0KB)

이 테이블의 모든 파일과 폴더가 업로드됩니다.

이름	폴더	유형	크기
1.JPG	-	image/jpeg	28.0KB

대상

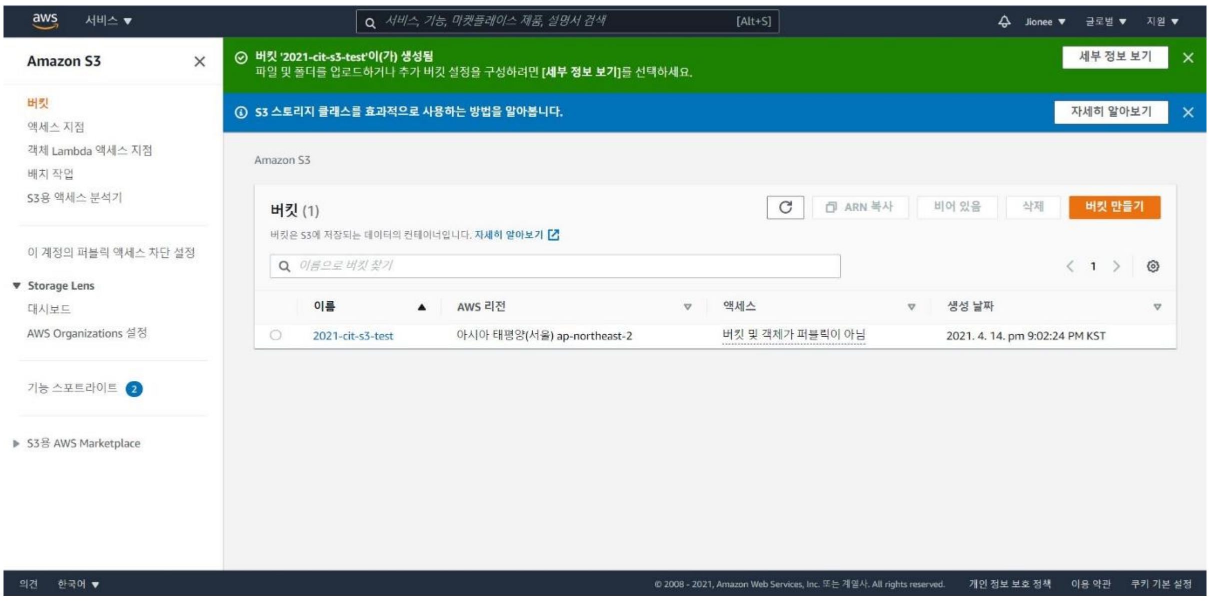
대상

s3://2021-cit-s3-test/images/

대상 세부 정보

다음 버킷 설정은 지정된 대상에 저장된 새 객체에 영향을 줍니다.

의견
한국어
© 2008 - 2021, Amazon Web Services, Inc. 또는 계열사. All rights reserved.
개인 정보 보호 정책
이용 약관
쿠키 기본 설정



버킷 '2021-cit-s3-test'이(가) 생성됨

파일 및 폴더를 업로드하거나 추가 버킷 설정을 구성하려면 **[세부 정보 보기]**를 선택하세요.

S3 스토리지 클래스를 효과적으로 사용하는 방법을 알아봅니다.

버킷 (1)

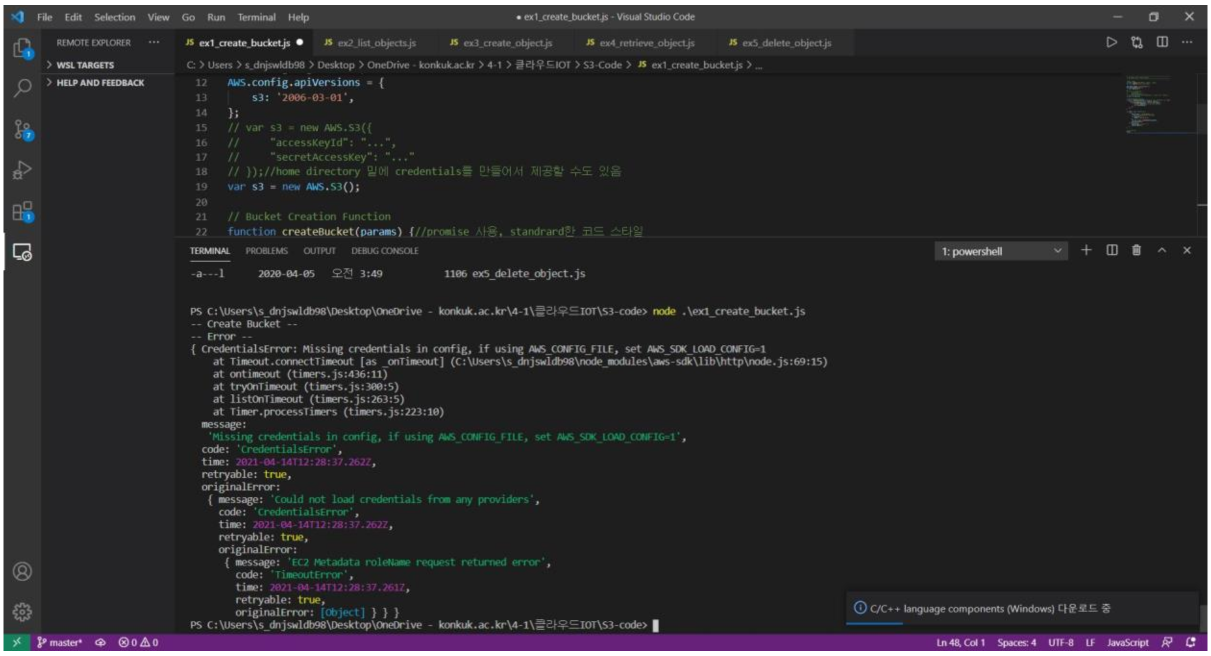
버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

이름	AWS 리전	액세스	생성 날짜
2021-cit-s3-test	아시아 태평양(서울) ap-northeast-2	버킷 및 객체가 퍼블릭이 아님	2021. 4. 14. pm 9:02:24 PM KST

의견
한국어
© 2008 - 2021, Amazon Web Services, Inc. 또는 계열사. All rights reserved.
개인 정보 보호 정책
이용 약관
쿠키 기본 설정

1-2) Create a Node.JS Application with AWS SDK to create a S3 object in the bucket
- Credentials

* credential 을 설정하지 않고 node.js 파일 실행 시 credential missing error 발생 >>

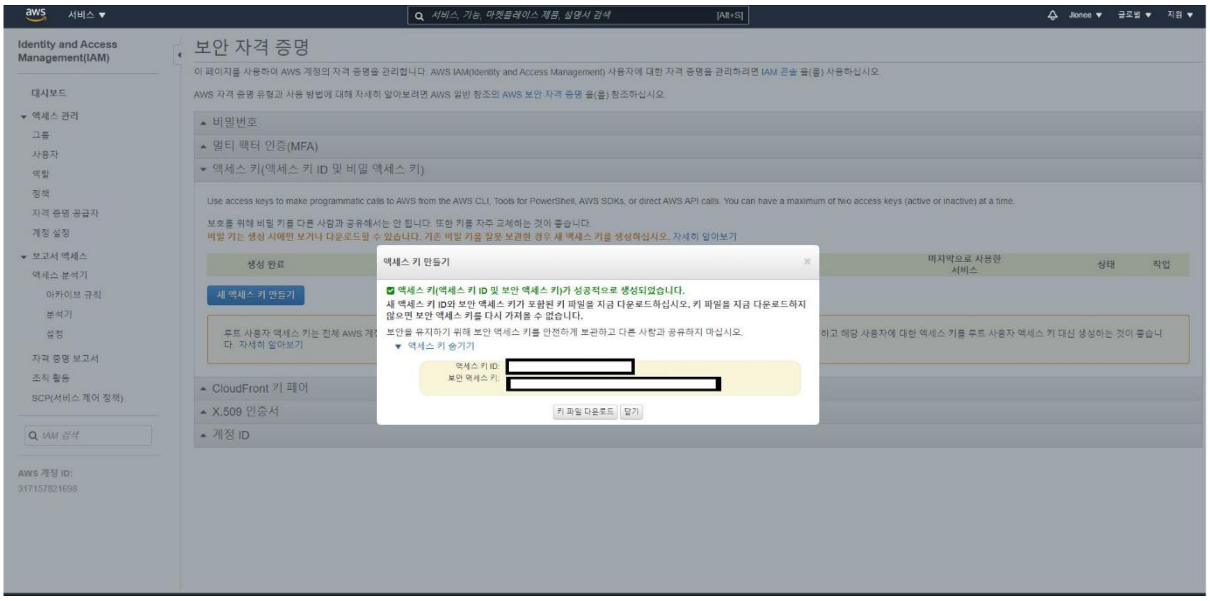


```
AWS.config.apiVersions = {
    s3: '2006-03-01',
};
// var s3 = new AWS.S3({
//     "accessKeyId": "...",
//     "secretAccessKey": "..."
// });//home directory 밑에 credentials를 만들어서 제공할 수도 있음
var s3 = new AWS.S3();

// Bucket Creation Function
function createBucket(params) {//promise 사용, standrard한 코드 스타일
```

```
-a---l       2020-04-05   오전 3:49          1106 ex5_delete_object.js


PS C:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-code> node .\ex1_create_bucket.js
-- Create Bucket --
-- Error --
{ CredentialsError: Missing credentials in config, if using AWS_CONFIG_FILE, set AWS_SDK_LOAD_CONFIG=1
    at Timeout.connectTimeout [as _onTimeout] (C:\Users\s_dnjswldb98\node_modules\aws-sdk\lib\http\node.js:69:15)
    at ontimeout (timers.js:436:11)
    at tryOnTimeout (timers.js:300:5)
    at listOnTimeout (timers.js:263:5)
    at Timer.processTimers (timers.js:223:10)
  message:
   'Missing credentials in config, if using AWS_CONFIG_FILE, set AWS_SDK_LOAD_CONFIG=1',
  code: 'CredentialsError',
  time: 2021-04-14T12:28:37.262Z,
  retryable: true,
  originalError:
   { message: 'Could not load credentials from any providers',
     code: 'CredentialsError',
     time: 2021-04-14T12:28:37.262Z,
     retryable: true,
     originalError:
      { message: 'EC2 Metadata roleName request returned error',
        code: 'TimeoutError',
        time: 2021-04-14T12:28:37.261Z,
        retryable: true,
        originalError: [Object] } } }
PS C:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-code> 
```

File Edit Selection View Go Run Terminal Help
• ex1_create_bucket.js - Visual Studio Code
REMOTE EXPLORER
WSL TARGETS
HELP AND FEEDBACK
ex1_create_bucket.js
ex2_list_objects.js
ex3_create_object.js
ex4_retrieve_object.js
ex5_delete_object.js
C: > Users > s_dnjswldb98 > Desktop > OneDrive - konkuk.ac.kr > 4-1 > 클라우드IOT > S3-Code > ex1_create_bucket.js > ...
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
1: powershell
C/C++ language components (Windows) 다운로드 중
master* ⊗0 ⚠0
Ln 48, Col 1 Spaces: 4 UTF-8 LF JavaScript

>> Credential 을 발급받고, Configuration 설정을 해주었다.



AWS 계정 ID:
317157821698

보안 자격 증명

이 페이지를 사용하여 AWS 계정의 자격 증명을 관리합니다. AWS IAM(Identity and Access Management) 사용자에 대한 자격 증명을 관리하려면 IAM 콘솔 을(를) 사용하십시오.

AWS 자격 증명 유형과 사용 방법에 대해 자세히 알아보려면 AWS 일반 참조의 AWS 보안 자격 증명 을(를) 참조하십시오.

▴ 비밀번호
▴ 멀티 팩터 인증(MFA)
▾ 액세스 키(액세스 키 ID 및 비밀 액세스 키)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

보호를 위해 비밀 키를 다른 사람과 공유해서는 안 됩니다. 또한 키를 자주 교체하는 것이 좋습니다.
비밀 키는 생성 시에만 보거나 다운로드할 수 있습니다. 기존 비밀 키를 잘못 보관한 경우 새 액세스 키를 생성하십시오. 자세히 알아보기

생성 완료	마지막으로 사용한 서비스	상태	작업

새 액세스 키 만들기

루트 사용자 액세스 키는 전체 AWS 계
하고 해당 사용자에 대한 액세스 키를 루트 사용자 액세스 키 대신 생성하는 것이 좋습니다. 자세히 알아보기

▴ CloudFront 키 페어
▴ X.509 인증서
▴ 계정 ID

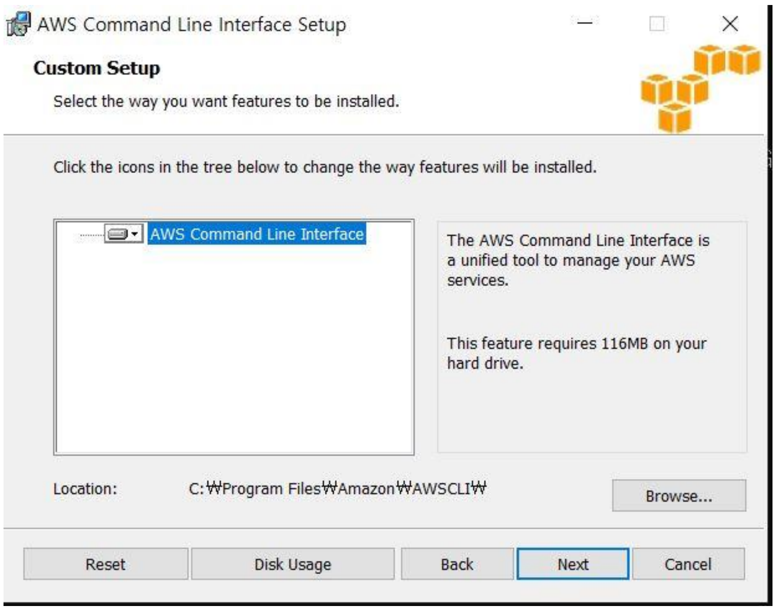
액세스 키 만들기

✓ 액세스 키(액세스 키 ID 및 보안 액세스 키)가 성공적으로 생성되었습니다.
새 액세스 키 ID와 보안 액세스 키가 포함된 키 파일을 지금 다운로드하십시오. 키 파일을 지금 다운로드하지 않으면 보안 액세스 키를 다시 가져올 수 없습니다.
보안을 유지하기 위해 보안 액세스 키를 안전하게 보관하고 다른 사람과 공유하지 마십시오.
▾ 액세스 키 숨기기

액세스 키 ID: [redacted]
보안 액세스 키: [redacted]

키 파일 다운로드 | 닫기

>> 윈도우에서 aws 명령어를 사용하기 위해 AWS Command Line interface 설치


AWS Command Line Interface Setup

Custom Setup

Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

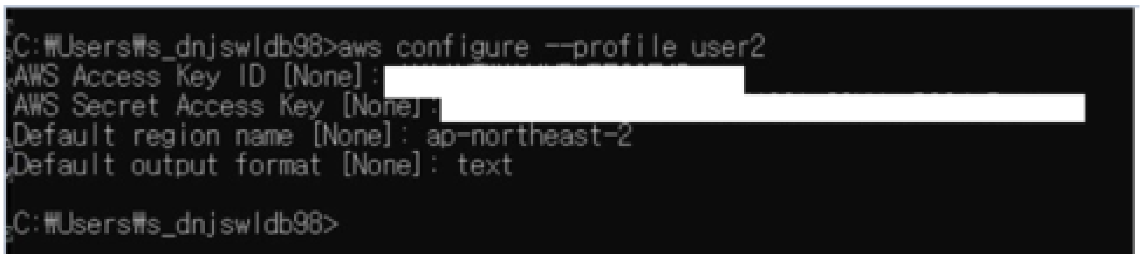
AWS Command Line Interface

The AWS Command Line Interface is a unified tool to manage your AWS services.

This feature requires 116MB on your hard drive.

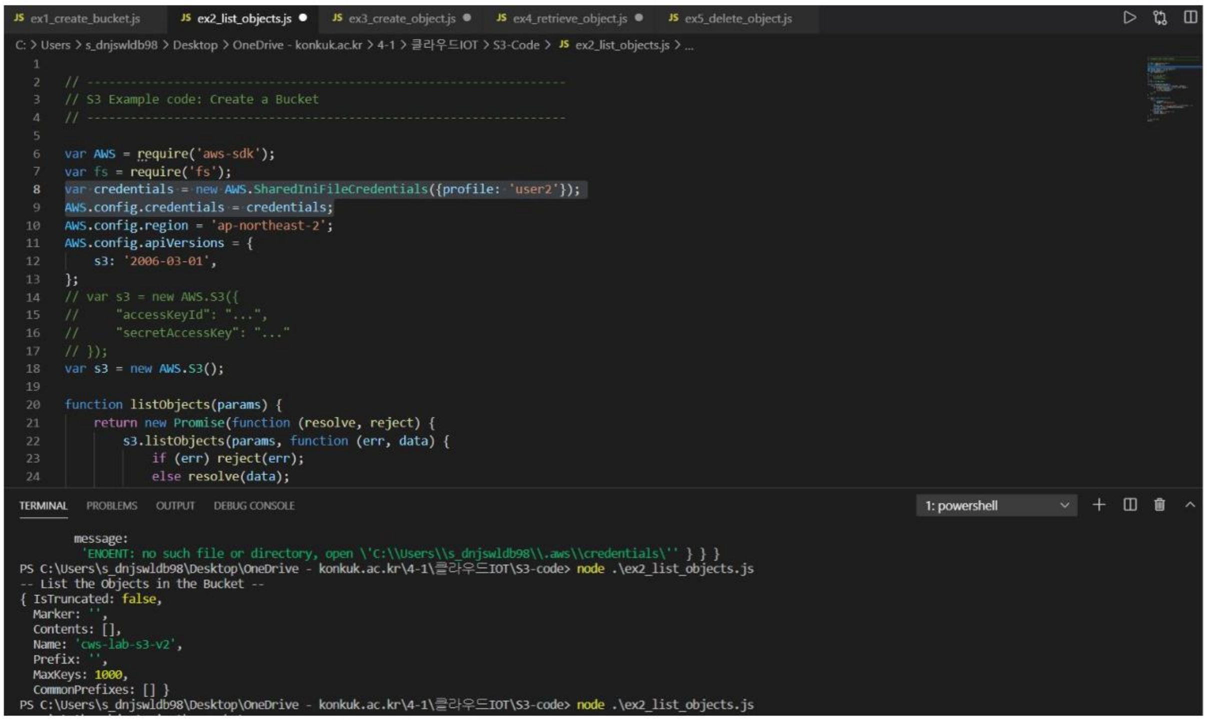
Location: C:₩Program Files₩Amazon₩AWSCLI₩

>> aws configure –profile user2 를 이용해서 credential 설정


```
C:₩Users₩s_dnjswldb98>aws configure --profile user2
AWS Access Key ID [None]: [redacted]
AWS Secret Access Key [None]: [redacted]
Default region name [None]: ap-northeast-2
Default output format [None]: text

C:₩Users₩s_dnjswldb98>
```

>> 코드에서 credential 사용

```
Var credentials = new AWS.SharedIniFileCredentials({profile: ‘user2’});
AWS.config.credentials = credentials;
```



```

// ---------------------------------------------------------------
// S3 Example code: Create a Bucket
// ---------------------------------------------------------------

var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};
// var s3 = new AWS.S3({
//     "accessKeyId": "...",
//     "secretAccessKey": "..."
// });
var s3 = new AWS.S3();

function listObjects(params) {
    return new Promise(function (resolve, reject) {
        s3.listObjects(params, function (err, data) {
            if (err) reject(err);
            else resolve(data);
```

```
        message:
         'ENOENT: no such file or directory, open \'C:\\Users\\s_dnjswldb98\\.aws\\credentials\'' } } }
PS C:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-code> node .\ex2_list_objects.js
-- List the Objects in the Bucket --
{ IsTruncated: false,
  Marker: '',
  Contents: [],
  Name: 'cws-lab-s3-v2',
  Prefix: '',
  MaxKeys: 1000,
  CommonPrefixes: [] }
PS C:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-code> node .\ex2_list_objects.js
```

- Create a S3 object in the bucket by uploading a file from notebook

#1. create_bucket.js

```
//버킷 만들기
var AWS = require('aws-sdk'); //object 만들어짐
var fs = require('fs'); //file 시스템
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};
var s3 = new AWS.S3();

// Bucket Creation Function
function createBucket(params) {//promise 사용, standrard한 코드 스타일
    return new Promise(function (resolve, reject) {
        s3.createBucket(params, function (err, data) {
            if (err) reject(err); // an error occurred
            else resolve(data);
        });
    });
}

var test = async function () {
    try {
        console.log("Region: ", AWS.config);
        console.log("env, ", process.env.AWS_CONFIG)
        console.log('-- Create Bucket --');
        // Bucket Creation Request Parameters
        var cb_params = {
            Bucket: "cws-lab-s3-v2", //만들 버킷 이름
        };
        var res1 = await createBucket(cb_params);
        console.log(res1);
    } catch (err) {
        console.log('-- Error --');
        console.log(err);
    }
}

// run the test
test();
```

#2. list_object.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var s3 = new AWS.S3();

function listObjects(params) {
    return new Promise(function (resolve, reject) {
        s3.listObjects(params, function (err, data) {
            if (err) reject(err);
            else resolve(data);
        });
    });
}
```

```
var test = async function () {
    try {
        var cb_params = {
            Bucket: "cws-lab-s3-v2",
        };
        console.log('-- List the Objects in the Bucket --');
        var res4 = await listObjects(cb_params);
        console.log(res4);
    } catch (err) {
        console.log('-- Error --');
        console.log(err);
    }
}

// run the test
test();
```

#3. create_object.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var s3 = new AWS.S3();

function createObject(params) {
    return new Promise(function (resolve, reject) {
        s3.upload(params, function (err, data) { //local 에서 S3object 로 업로드 하는 function
            if (err) reject(err);
            else resolve(data);
        })
    });
}

var test = async function () {
    try {
        // 1st Object
        const co_params1 = {
            Bucket: "cws-lab-s3-v2",
            Key: 'testImg.jpg', //만들 Object ID
            Body: fs.createReadStream("./testImg.jpg") //현재 디렉토리 여기에
 있는 애를 //크니까 stream 으로 읽어서 위 ID 로 object 생성
        };
        var res2 = await createObject(co_params1);
        console.log(res2);

    } catch (err) {
        console.log('-- Error --');
        console.log(err);
    }
}

// run the test
test();
```

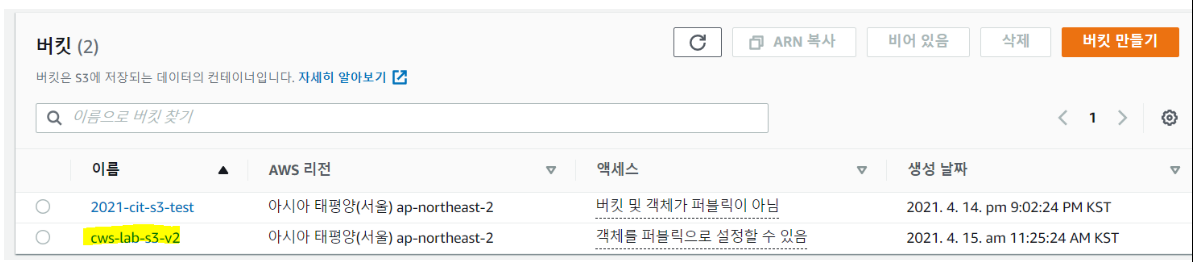
#4. retrieve_object.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var s3 = new AWS.S3();

function retrieveObject(params) {
    return new Promise(function (resolve, reject) {
        s3.getObject(params, function (err, data) {
            if (err) reject(err);
            else resolve(data);
        });
    });
}

var test = async function () {
    try {
        console.log('-- Retrieve an Object from the Bucket --');
        // Object Retrieval Request Parameters
        const ro_params = {
            Bucket: "cws-lab-s3-v2",
            Key: 'testImg.jpg', //이 object
        };
        var data = await retrieveObject(ro_params);
        // write the obejct to a local file
        fs.writeFile('./data.jpg', data.Body, (e, d) => {
            if (e) console.log(e);
            else console.log('Image is read and written to data.jpg !'); //s
3 에 있는 파일을 읽어서 local 의 data.jpg 에 저장하기
        });
    } catch (err) {
        console.log('-- Error --');
        console.log(err);
    }
}

// run the test
test();
```

#5. delete_object.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var s3 = new AWS.S3();

// Object Deletion
function deleteObject(params) {
    return new Promise(function(resolve, reject) {
        s3.deleteObject(params, function (err, data) {
            if(err) reject(err);
            else resolve(data);
        });
    });
}

var test = async function () {
    try {
        console.log('-- Delete an Object from the Bucket --');
        const do_params = {
            Bucket: "cws-lab-s3-v2",
            Key: 'testImg.jpg',
        };
        var res6 = await deleteObject(do_params);
        console.log(res6);
    } catch (err) {
        console.log('-- Error --');
        console.log(err);
    }
}

// run the test
test();
```

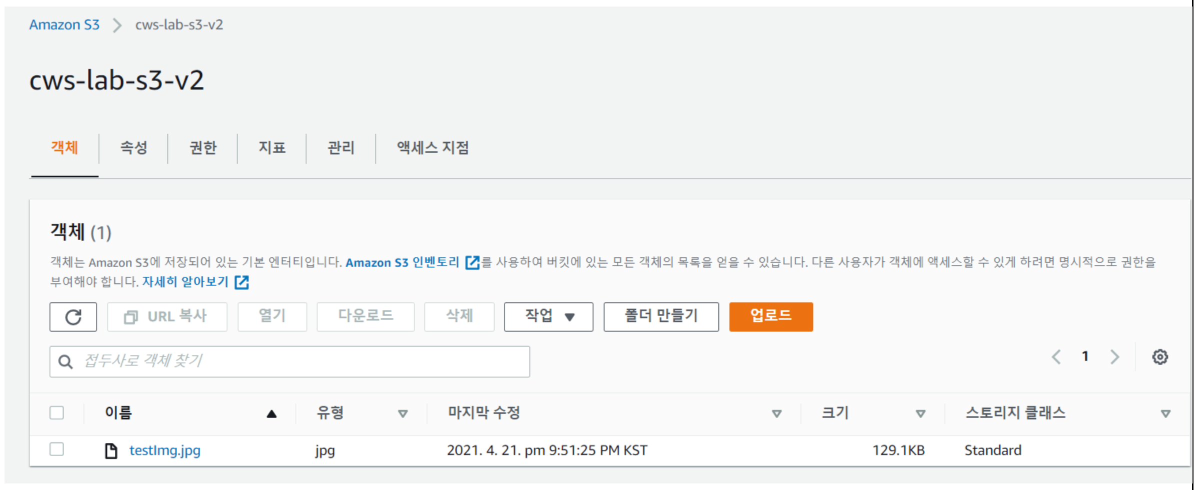
>> **생성 결과**



버킷 (2)

버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기

이름	AWS 리전	액세스	생성 날짜
2021-cit-s3-test	아시아 태평양(서울) ap-northeast-2	버킷 및 객체가 퍼블릭이 아님	2021. 4. 14. pm 9:02:24 PM KST
cws-lab-s3-v2	아시아 태평양(서울) ap-northeast-2	객체를 퍼블릭으로 설정할 수 있음	2021. 4. 15. am 11:25:24 AM KST



Amazon S3 > cws-lab-s3-v2

cws-lab-s3-v2

객체 (1)

객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 인벤토리를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. 자세히 알아보기

이름	유형	마지막 수정	크기	스토리지 클래스
testImg.jpg	jpg	2021. 4. 21. pm 9:51:25 PM KST	129.1KB	Standard

Task #2 : Create an AWS Lambda function to create a S3 object

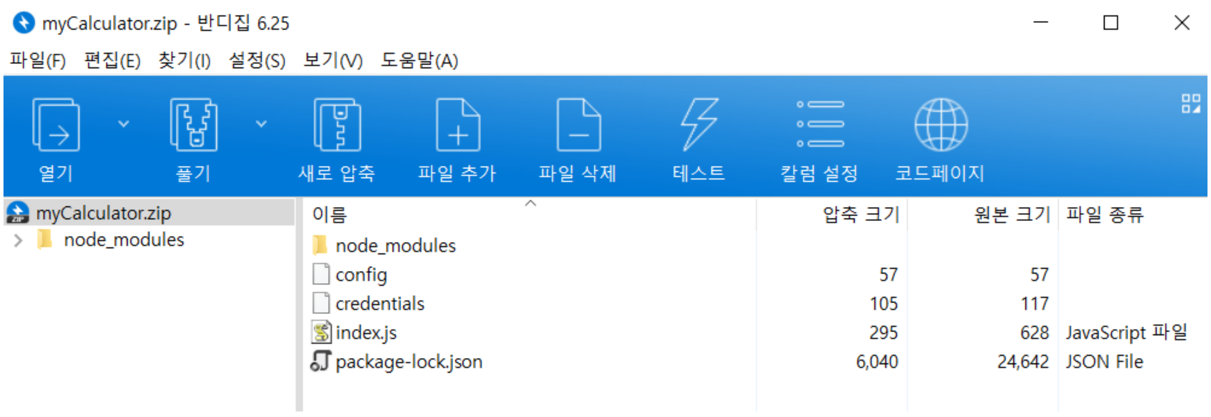
2-1) index.js 작성해서 압축 후 S3 에 올리기

- 앞서 작성하였던 index.js 와 node_modules, credentials, 그리고 package-lock.json 까지 압축한 후 createDeploymentPackage.js 를 실행시켜 S3 스토리지에 압축 파일을 저장한다.

createDeploymentPackage.js

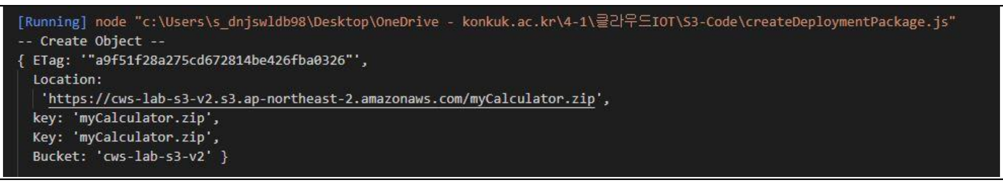
```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var s3 = new AWS.S3();

var createDeploymentPackage = async function (dp_params){
    function createBucket(bucket_name){
        var cb_params = {
            Bucket: bucket_name,
        };
        return new Promise(function(resolve,reject){
            s3.createBucket(cb_params,function(err,data){
                if(err) reject(err); //an error occurred
                else resolve(data);
            });
        });
    }

    function createObject(bucket_name,zipfile){
        const co_params = {
            Bucket: bucket_name,
            Key:zipfile,
            Body:fs.createReadStream("./"+zipfile),
        };
        return new Promise(function(resolve,reject){
            s3.upload(co_params,function(err,data){
                if(err) reject(err); //an error occurred
                else resolve(data);
            })
        })
    }

    try{
        if(dp_params.newBucketFlag){
            console.log('-- Create Bucket --');
            var res = await createBucket(dp_params.bucket);
            console.log(res)
        }
        console.log('-- Create Object --');
        var res = await createObject(dp_params.bucket,dp_params.zipfile);
        console.log(res);
    }catch(err) {console.log(err)}
}

var dp_params = {
    bucket :'cws-lab-s3-v2',
    newBucketFlag : false,
    zipfile : 'myCalculator.zip'
};

createDeploymentPackage(dp_params)
```

- myCalculator.zip



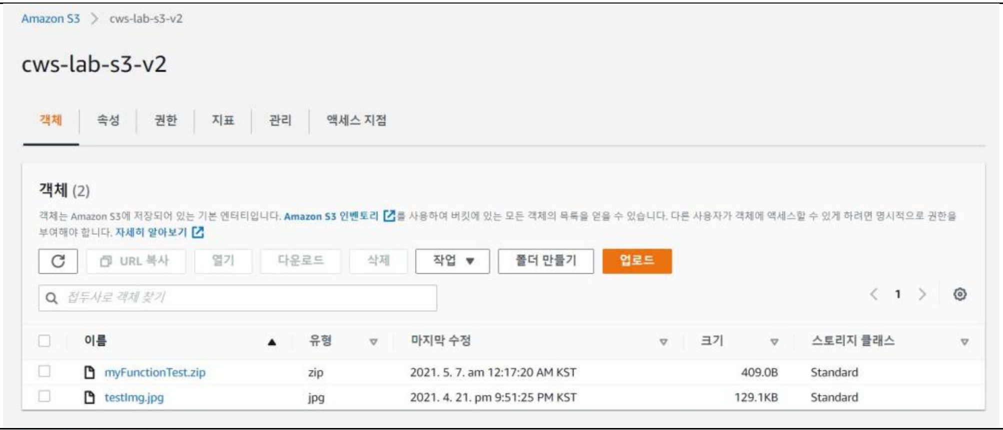
이름	압축 크기	원본 크기	파일 종류
node_modules			
config	57	57	
credentials	105	117	
index.js	295	628	JavaScript 파일
package-lock.json	6,040	24,642	JSON File

결과



```
[Running] node "c:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-Code\createDeploymentPackage.js"
-- Create Object --
{ ETag: '"a9f51f28a275cd672814be426fba0326"',
  Location:
   'https://cws-lab-s3-v2.s3.ap-northeast-2.amazonaws.com/myCalculator.zip',
  key: 'myCalculator.zip',
  Key: 'myCalculator.zip',
  Bucket: 'cws-lab-s3-v2' }
```

실행 전



Amazon S3 > cws-lab-s3-v2

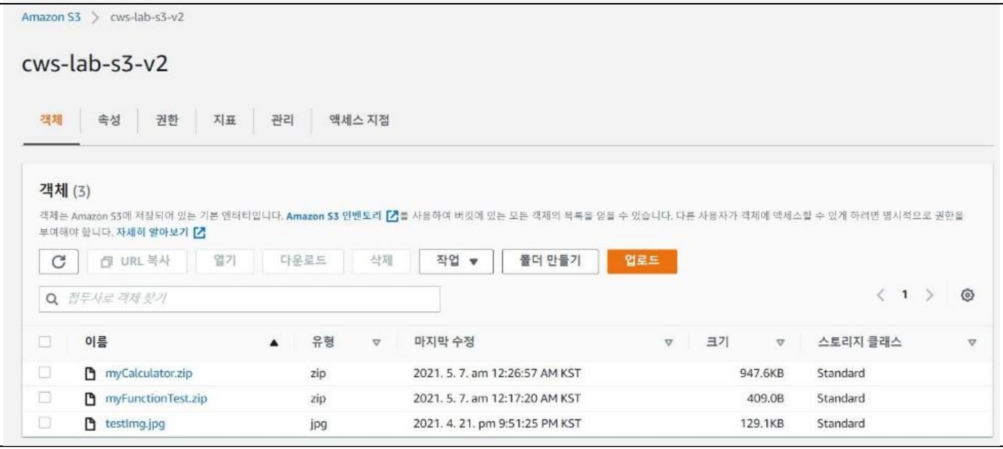
cws-lab-s3-v2

객체 (2)

객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 인벤토리 를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. 자세히 알아보기

이름	유형	마지막 수정	크기	스토리지 클래스
myFunctionTest.zip	zip	2021. 5. 7. am 12:17:20 AM KST	409.0B	Standard
testImg.jpg	jpg	2021. 4. 21. pm 9:51:25 PM KST	129.1KB	Standard

실행 후



Amazon S3 > cws-lab-s3-v2

cws-lab-s3-v2

객체 (3)

객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. Amazon S3 인벤토리 를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. 자세히 알아보기

이름	유형	마지막 수정	크기	스토리지 클래스
myCalculator.zip	zip	2021. 5. 7. am 12:26:57 AM KST	947.6KB	Standard
myFunctionTest.zip	zip	2021. 5. 7. am 12:17:20 AM KST	409.0B	Standard
testImg.jpg	jpg	2021. 4. 21. pm 9:51:25 PM KST	129.1KB	Standard

실행 후 S3 스토리지 내부에 myCalculator.zip 파일이 생성되었음을 볼 수 있다.

2-2) Create an AWS Lambda function from the S3 object

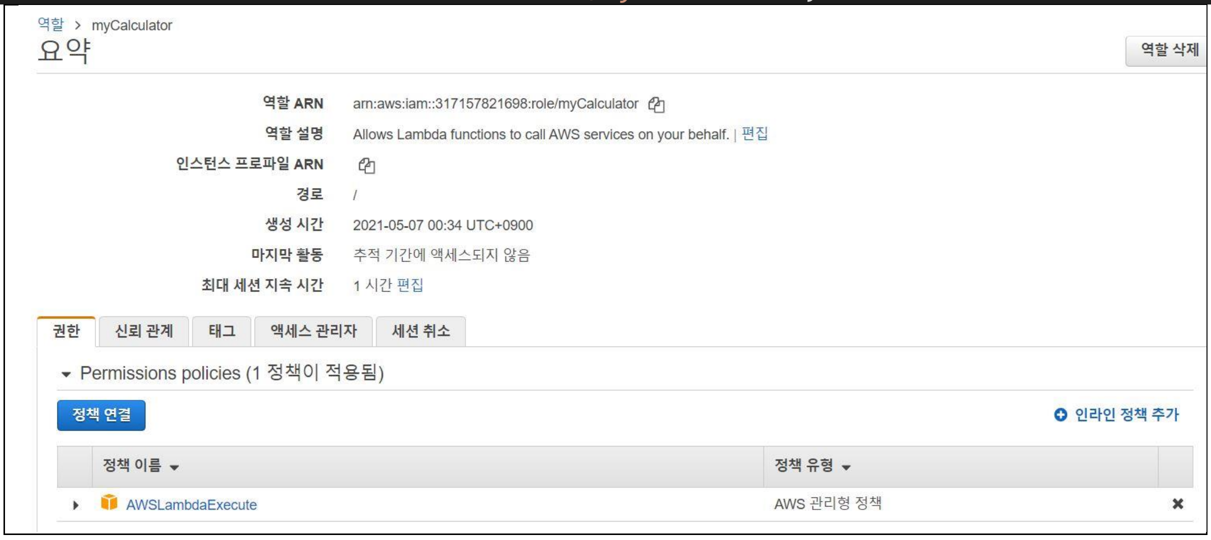
- LambdaExecution 을 위한 iam role 을 생성하여 createLambdaFunction.js 를 작성한 후 실행시킨다. 그 결과 s3 에 저장해 놓은 zip 파일을 이용한 lambda function 이 생성된다.

createLambdaFunction.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var lambda = new AWS.Lambda();

var params = {
    Code: {
        S3Bucket : 'cws-lab-s3-v2',
        S3Key : 'myCalculator.zip'
    },
    FunctionName : 'myCalculator',
    Handler : 'index.handler',
    Role : 'arn:aws:iam::317157821698:role/myCalculator',
    Runtime : 'nodejs14.x',//Node.js 14.x
    Description:""
};

lambda.createFunction(params,function(err,data){
    if(err) console.log(err);
    else console.log(data);
});
```

- iam role 생성

```
Role : 'arn:aws:iam::317157821698:role/myCalculator',
```



역할 > myCalculator

요약

역할 삭제

역할 ARN	arn:aws:iam::317157821698:role/myCalculator
역할 설명	Allows Lambda functions to call AWS services on your behalf. | 편집
인스턴스 프로파일 ARN	
경로	/
생성 시간	2021-05-07 00:34 UTC+0900
마지막 활동	추적 기간에 액세스되지 않음
최대 세션 지속 시간	1 시간 편집

권한 | 신뢰 관계 | 태그 | 액세스 관리자 | 세션 취소

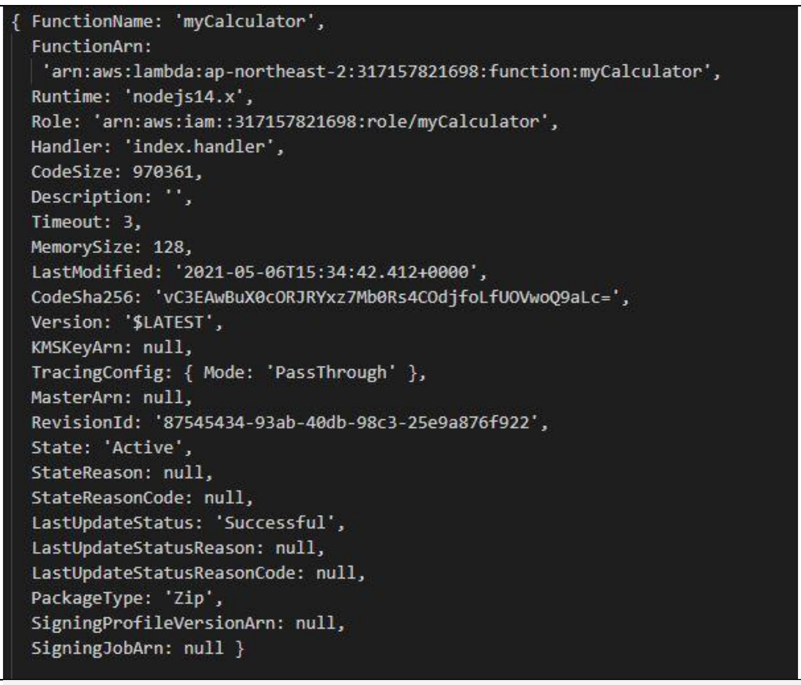
Permissions policies (1 정책이 적용됨)

정책 연결

인라인 정책 추가

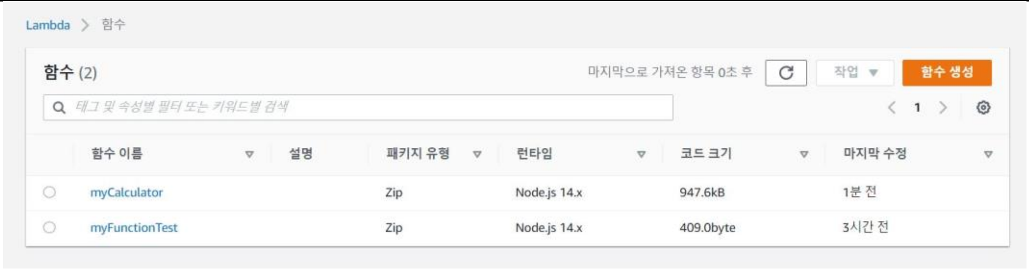
정책 이름	정책 유형
AWSLambdaExecute	AWS 관리형 정책

결과



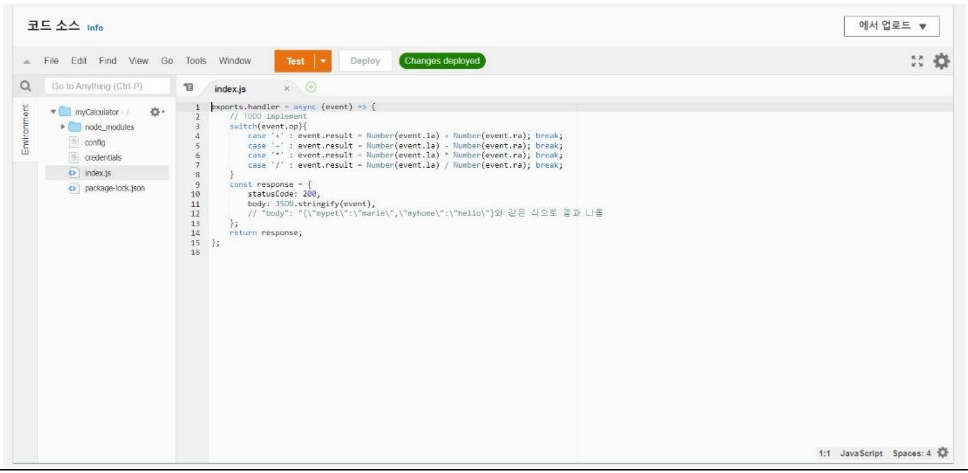
```
{ FunctionName: 'myCalculator',
  FunctionArn:
   'arn:aws:lambda:ap-northeast-2:317157821698:function:myCalculator',
  Runtime: 'nodejs14.x',
  Role: 'arn:aws:iam::317157821698:role/myCalculator',
  Handler: 'index.handler',
  CodeSize: 970361,
  Description: '',
  Timeout: 3,
  MemorySize: 128,
  LastModified: '2021-05-06T15:34:42.412+0000',
  CodeSha256: 'vC3EAwBuX0cORJRYxz7Mb0Rs4COdjfoLfUOVwoQ9aLc=',
  Version: '$LATEST',
  KMSKeyArn: null,
  TracingConfig: { Mode: 'PassThrough' },
  MasterArn: null,
  RevisionId: '87545434-93ab-40db-98c3-25e9a876f922',
  State: 'Active',
  StateReason: null,
  StateReasonCode: null,
  LastUpdateStatus: 'Successful',
  LastUpdateStatusReason: null,
  LastUpdateStatusReasonCode: null,
  PackageType: 'Zip',
  SigningProfileVersionArn: null,
  SigningJobArn: null }
```

실행 후



Lambda > 함수
함수 (2)
마지막으로 가져온 항목 0초 후
작업
함수 생성
태그 및 속성별 필터 또는 키워드별 검색
1

함수 이름	설명	패키지 유형	런타임	코드 크기	마지막 수정
myCalculator		Zip	Node.js 14.x	947.6kB	1분 전
myFunctionTest		Zip	Node.js 14.x	409.0byte	3시간 전



```
exports.handler = async (event) => {
    // TODO implement
    switch(event.op){
        case '+' : event.result = Number(event.la) + Number(event.ra); break;
        case '-' : event.result = Number(event.la) - Number(event.ra); break;
        case '*' : event.result = Number(event.la) * Number(event.ra); break;
        case '/' : event.result = Number(event.la) / Number(event.ra); break;
    }
    const response = {
        statusCode: 200,
        body: JSON.stringify(event),
        // "body": "{\"mypet\":\"marie\",\"myhome\":\"hello\"}와 같은 식으로 결과 나옴
    };
    return response;
};
```

실행 후 S3 에 저장해 놓았던 myCalculator.zip 로 lambda function 이 만들어 졌음을 볼 수 있다.

Task #3 : Create a Node.JS application to invoke the AWS Lambda function in Task#2

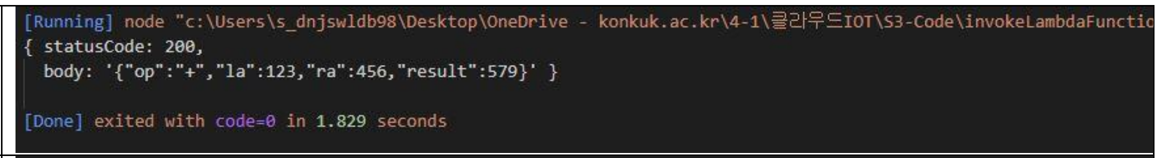
- lambda function 을 실행시키는 invokeLamdaFunction.js 를 작성 후 실행시킨다.

invokeLamdaFunction.js

```
var AWS = require('aws-sdk');
var fs = require('fs');
var credentials = new AWS.SharedIniFileCredentials({profile: 'user2'});
AWS.config.credentials = credentials;
AWS.config.region = 'ap-northeast-2';
AWS.config.apiVersions = {
    s3: '2006-03-01',
};

var lambda = new AWS.Lambda();

const exp = {"op":"+", "la":123, "ra":456};
var params = {
    FunctionName : "myCalculator",
    InvocationType : "RequestResponse",
    Payload : JSON.stringify(exp)
};

lambda.invoke(params,function(err,data){
    if(err) console.log(err);
    else console.log(JSON.parse(data.Payload));
});
```

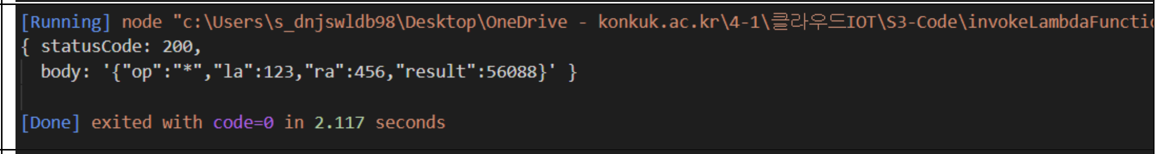
결과

+



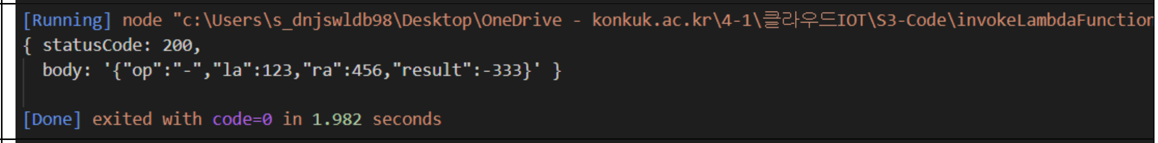
```
[Running] node "c:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-Code\invokeLambdaFunctio
{ statusCode: 200,
  body: '{"op":"+","la":123,"ra":456,"result":579}' }

[Done] exited with code=0 in 1.829 seconds
```

*****



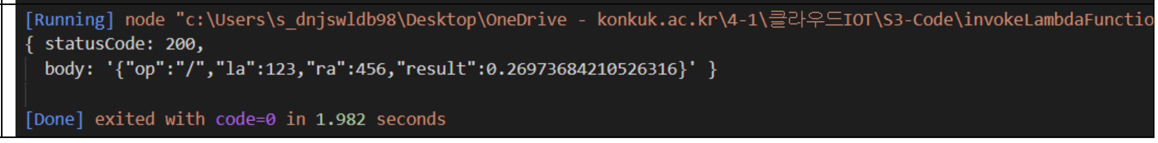
```
[Running] node "c:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-Code\invokeLambdaFuncti
{ statusCode: 200,
  body: '{"op":"*","la":123,"ra":456,"result":56088}' }

[Done] exited with code=0 in 2.117 seconds
```

-



```
[Running] node "c:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-Code\invokeLambdaFunction
{ statusCode: 200,
  body: '{"op":"-","la":123,"ra":456,"result":-333}' }

[Done] exited with code=0 in 1.982 seconds
```

/



```
[Running] node "c:\Users\s_dnjswldb98\Desktop\OneDrive - konkuk.ac.kr\4-1\클라우드IOT\S3-Code\invokeLambdaFunctio
{ statusCode: 200,
  body: '{"op":"/","la":123,"ra":456,"result":0.26973684210526316}' }

[Done] exited with code=0 in 1.982 seconds
```

실행 후 넣은 input(param)에 따라 결과가 다르게 나오는 것을 확인할 수 있다.