

AWS Lambda Hands On Lab

[AWS Console 로그인](#)

[IAM 사용자 생성](#)

[압축된 이미지가 저장될 S3 Bucket 생성하기](#)

[Github Repository fork 및 코드 파악하기](#)

[코드 봄보기](#)

[CD Pipeline 보기](#)

[Github Actions 실행하기](#)

[Lambda Console로 가서 배포된 Lambda 확인하기](#)

[Lambda에 Amazon API Gateway 연동하기](#)

[테스트 해보기](#)

[리소스 정리하기 \(CloudFormation으로 바꿔야댐!\)](#)

AWS Console 로그인

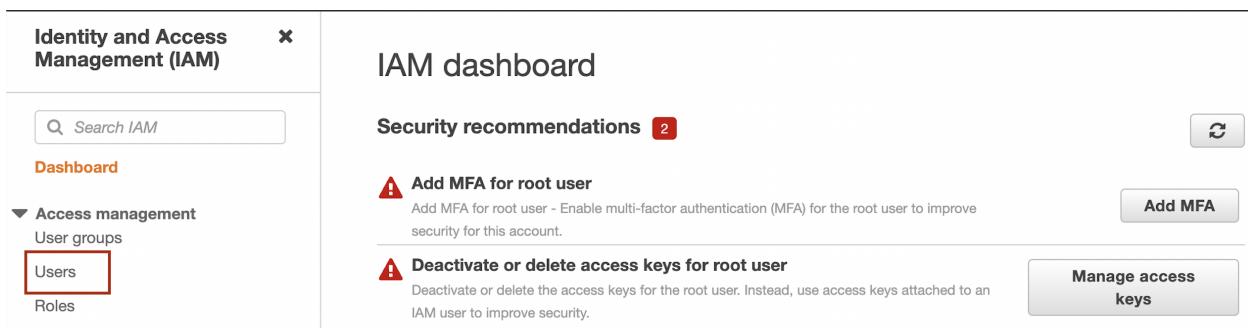
- [AWS Console](#)에 로그인 합니다.

IAM 사용자 생성

| 이 IAM 사용자는 이후 Lambda 코드를 AWS에 배포할 때 사용합니다.

1. Console에 IAM 검색 후 IAM Console로 이동합니다.

이후 좌측에 있는 [Users](#)를 클릭합니다.



2. 우측에 있는 [Add users](#)를 클릭합니다.

Users (12) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Find users by username or access key

Add users

3.

- Username: `lambda-handson-닉네임` (ex. `lambda-handson-roy`)
- Access key - Programmatic access 체크
- 이후 `Next: Permissions` 클릭

4.

- `Attach existing policies directly` 선택
- `AdministratorAccess` 체크

▼ Set permissions

Add user to group

Copy permissions from existing user

Attach existing policies directly

Create policy

Filter policies ▾ Search Showing 763 results

Policy name	Type	Used as
<input checked="" type="checkbox"/> AdministratorAccess	Job function	Permissions policy (4)
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	None
<input type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	None
<input type="checkbox"/> AlexaForBusinessDeviceSetup	AWS managed	None
<input type="checkbox"/> AlexaForBusinessFullAccess	AWS managed	None
<input type="checkbox"/> AlexaForBusinessGatewayExecution	AWS managed	None
<input type="checkbox"/> AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None

5. `Next: Tags` 클릭 후 `Next: Review` 를 클릭하고, 마지막으로 `Create user` 를 클릭합니다.
그러면 아래의 화면이 나옵니다.

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://598334522273.signin.aws.amazon.com/console>

[Download .csv](#)

	User	Access key ID	Secret access key
▶	lambda-handson-roy	AKIAYWT4EIOQXMKL4F6S	***** Show



해당 화면에 나오는 **Access key ID**, **Secret access key** 는 이 화면을 벗어나면 다시 찾을 수 없습니다.
이후에 둘 다 사용되므로 모두 메모장 등의 다른 곳에 기록해 두어야 합니다.

압축된 이미지가 저장될 S3 Bucket 생성하기

S3는 Bucket 단위로 사용합니다. Bucket은 말 그대로 데이터들을 담을 저장소라고 생각하시면 됩니다.

1. Console 검색창에 S3를 검색하고, S3 console로 이동합니다.
2. 우측 상단에 있는 **Create bucket** 을 클릭합니다.

Amazon S3 > Buckets

▶ **Account snapshot** [View Storage Lens dashboard](#)
 Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (14) [Info](#) [C](#) [Copy ARN](#) [Empty](#) [Delete](#) **[Create bucket](#)**

Buckets are containers for data stored in S3. [Learn more](#)

3. **General Configuration** :
 - Bucket name: **lambda-handson-닉네임** 지정 (ex. **lambda-handson-roy**)

- 그 외의 나머지 값들은 그대로 둡니다.

4. **Object Ownership** : 기본 값 그대로 높습니다.
5. **Block Public Access settings for this bucket** 에서 **Block all public access** 를 체크 해제 합니다. 해제하면 아래에 경고ダイ얼로그가 나오는데, 해당ダイ얼로그도 체크 해줍니다.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through any access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through new public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through any public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

6. **Bucket Versioning**, **Tags**, **Default encryption** 등은 기본 값을 그대로 유지하고 가장 하단의 **Create bucket** 버튼을 클릭해 Bucket을 생성합니다.
7. Bucket 목록 중 방금 만든 bucket을 클릭하고, **Permissions** 탭을 클릭합니다.
8. 아래로 스크롤 해 **Bucket Policy** 를 찾고, **Edit** 버튼을 눌러 아래의 내용을 입력해 줍니다.

```
{  
    "Version": "2012-10-17",  
    "Id": "Policy1605861118540",  
    "Statement": [  
        {  
            "Sid": "Stmt1605861113574",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::lambda-handson-닉네임/*"  
        }  
    ]  
}
```

"Resource" 부분의 값에는 본인이 생성한 bucket 이름을 넣어줍니다.

맨 뒤에 /* 꼭 붙여야 해요!

붙여넣기 후 Save changes 를 클릭해 적용합니다.

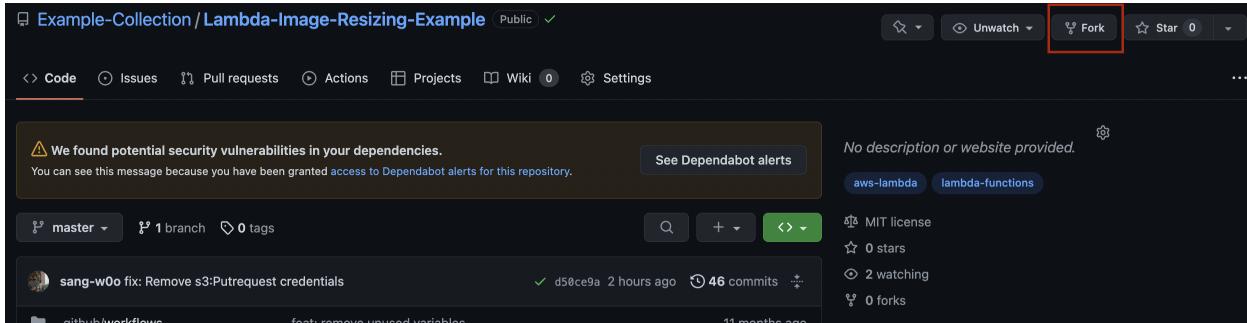
9. 다시 Permissions 탭으로 가서 하단에 Cross-origin resource sharing(CORS)에서 Edit 을 클릭해 아래 내용을 입력해 줍니다.

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

10. 여기까지 압축된 이미지가 저장될 Bucket의 생성 및 설정을 모두 완료했어요!

Github Repository fork 및 코드 파악하기

- 이 Github Repository 를 자신의 저장소로 fork 합니다.



코드 봄보기

이번 세션에서 람다가 수행하는 코드는 그렇게 중요하지 않습니다.
단지 클라이언트로부터 `multipart/form-data` 형식으로 사진을 받고,
width와 height의 query parameter를 받아서 Lambda function이 사진
을 조건에 맞게 압축한 후 S3로 업로드한다는 것만 알면 됩니다.

- 모든 코드는 `src/` 폴더 하위에 있습니다. 총 3개의 파일이 있는데요, 각각의 역할은 아래와 같습니다.
 - `index.ts` : `process.env`에서 `AWS_S3_BUCKET` 을 읽어와 어떤 bucket에 사진을 업로드할지 정하고, bucket 이름을 반환해줍니다.
 - `compressor.service.ts` : `sharp` 라이브러리를 사용해 이미지를 압축하는 `compress()` 함수가 구현되어 있습니다.
 - `handler.ts` : API Gateway가 받은 요청을 Lambda에게 전달하면, 우선 이 파일 내의 `upload()` 함수가 동작합니다. 이후 request에서 사진과 width, height를 `parseRequest()` 가 가져오고, `uploadToS3()` 에서 압축을 수행하는 `compress()` 를 호출한 후 S3에 업로드 합니다.

```
const upload: Handler = async (event: APIGatewayProxyEvent) => {
  const requestInfo = await parseRequest(event);
  const response = await uploadToS3(requestInfo);
  return {
    statusCode: 200,
    body: JSON.stringify(response),
    headers: {
      "Access-Control-Allow-Headers": "*",
      "Access-Control-Allow-Origin": "*",
      "Access-Control-Allow-Methods": "HEAD,OPTIONS,POST,GET",
    },
  };
};
```

CD Pipeline 보기

CD(Continuous Delivery)는 지속적 전달, 배포를 의미하는 단어이며, 이 애플리케이션의 경우 램다 코드가 수정되면 이를 AWS에 지속적으로 배포해 수정된 Lambda 함수를 클라이언트가 사용할 수 있게 합니다. 그리고 **CD에서 수행되는 일련의 작업들의 묶음을 Pipeline**이라 합니다.

- 이 프로젝트의 CD에 사용되는 파일은 두 가지가 있습니다. 우선 최상위 경로에 있는 `serverless.yml` 파일과 `.github/workflows/deploy.yml` 파일이 있습니다.

```
# serverless.yml

service: ImageResizing
plugins:
  - serverless-plugin-typescript

provider:
  lambdaHashingVersion: "20201221"
  name: aws
  runtime: nodejs14.x
  stage: DEVELOPMENT
  region: ap-northeast-2
```

```

environment:
  TZ: "Asia/Seoul"
  AWS_S3_BUCKET: ${env:AWS_S3_BUCKET}

functions:
  resize:
    handler: src/image-resize/handler.upload

```

- 간단히 보면,
 - 서비스 이름은 `ImageResizing` 입니다.
 - 해당 배포가 진행될 곳(`provider.name`)은 aws 입니다.
 - 람다 코드가 실행될 runtime 환경은 `nodejs14.x` 입니다.
 - `functions.resize.handler` : `src/image-resize/handler.ts`에 있는 `upload()` 가 이 Lambda가 호출되었을 때 실행될 handler입니다.

```

# .github/workflows/deploy.yml

name: Deploy Lambda

on:
  push:
    branches: [master]

jobs:
  Deploy:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node-version: [14.x]
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Install Packages Node ${{ matrix.node-version }}
        uses: actions/setup-node@v1
        with:
          node-version: ${{ matrix.node-version }}

      - run: yarn install --frozen-lockfile

      - name: Serverless Deploy
        uses: serverless/github-action@v2.1.0

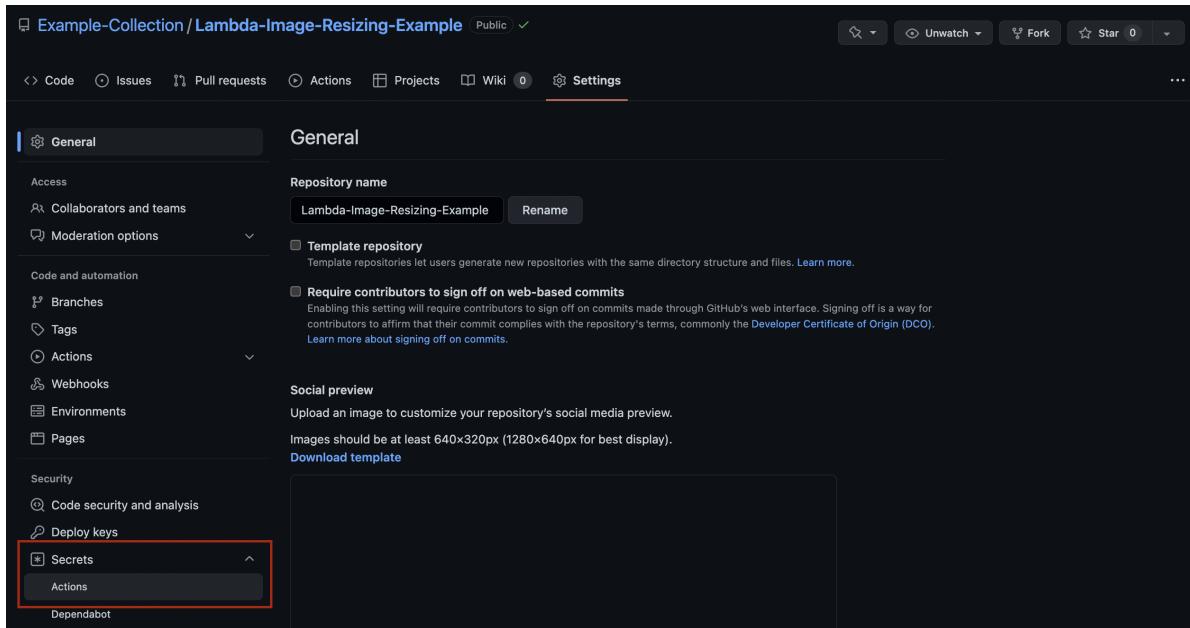
```

```
env:  
  AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}  
  AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}  
  AWS_S3_BUCKET: ${{ secrets.AWS_S3_BUCKET }}  
with:  
  args: deploy
```

- 마찬가지로 간단히 보면,
 - 우선 환경은 `node 14.x`에서 수행되며,
 - `yarn install`로 `package.json`에 기술된 필요한 외부 라이브러리들을 모두 다운로드 합니다.
 - 이 람다 함수를 Serverless가 배포할 때 어떤 AWS 계정에, 그리고 배포할 수 있는 권한이 있는지를 이전에 만들었던 IAM User의 인증 정보를 전달해 알려줍니다.

Github Actions 실행하기

- 위에서 `deploy.yml` 파일을 봤을 때 `env` 부분의 `AWS_ACCESS_KEY_ID`를 보면 `${{ secrets.AWS_ACCESS_KEY_ID }}`로 값이 지정되어 있습니다. 이렇게 하는 이유는 IAM 사용자의 정보가 유출되면, 악성 사용자들에 의해 어뷰징이 발생하고, 이는 엄청난 과금으로 이어질 수 있기 때문에 보안 상의 이유로 코드에 직접 넣지 않는 것입니다.
- 이제 이를 지정하기 위해 Github에 fork한 곳으로 이동해서, `Settings`를 클릭해 설정 페이지로 이동합니다.
 - 이후 왼쪽 하단에 `Secrets`를 클릭하고, `Actions`에 들어갑니다.



- 다음으로 **New repository secret** 을 클릭합니다.

Actions secrets

New repository secret

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for [Scale document down](#).

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

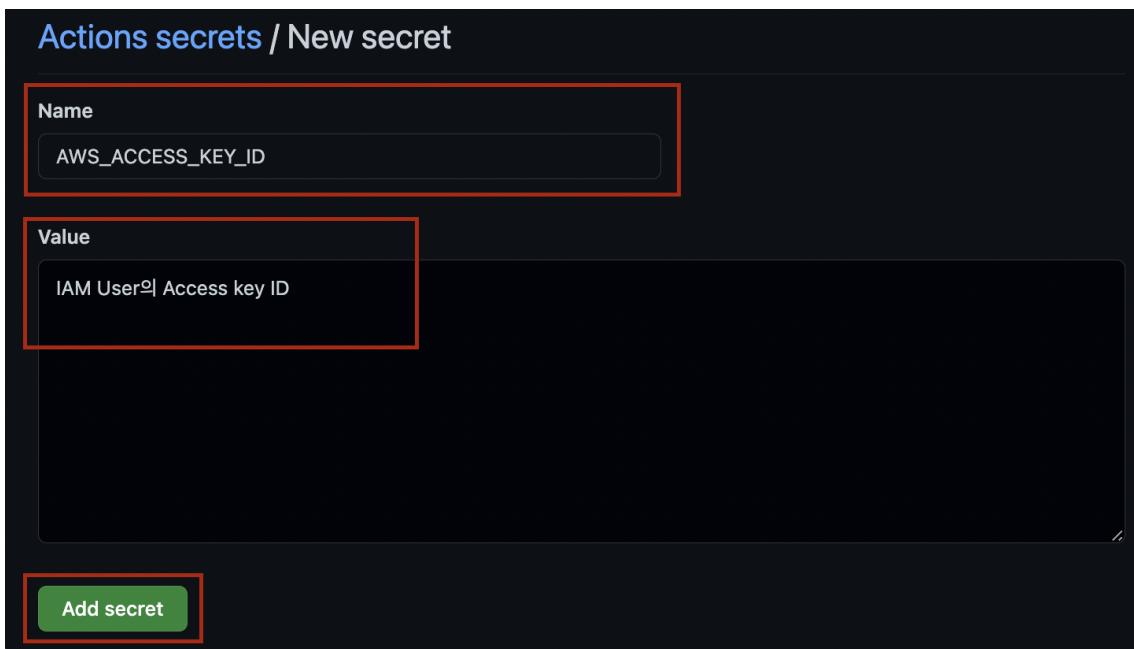
Environment secrets

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

[Manage your environments and add environment secrets](#)

- 이제 총 3개의 repository secret을 생성해 줍니다.
 - AWS_ACCESS_KEY_ID** : IAM User의 Access key ID
 - AWS_SECRET_ACCESS_KEY** : IAM User의 Secret access key
 - AWS_S3_BUCKET** : 처음에 생성한 S3 Bucket의 이름



- 모든 Repository Secret이 만들어지면, 아래와 같이 총 3개의 secret이 보입니다. 그리고 이제 Github Action script에서 이를 참조할 수 있습니다.

Actions secrets

[New repository secret](#)

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

Environment secrets

There are no secrets for this repository's environments.

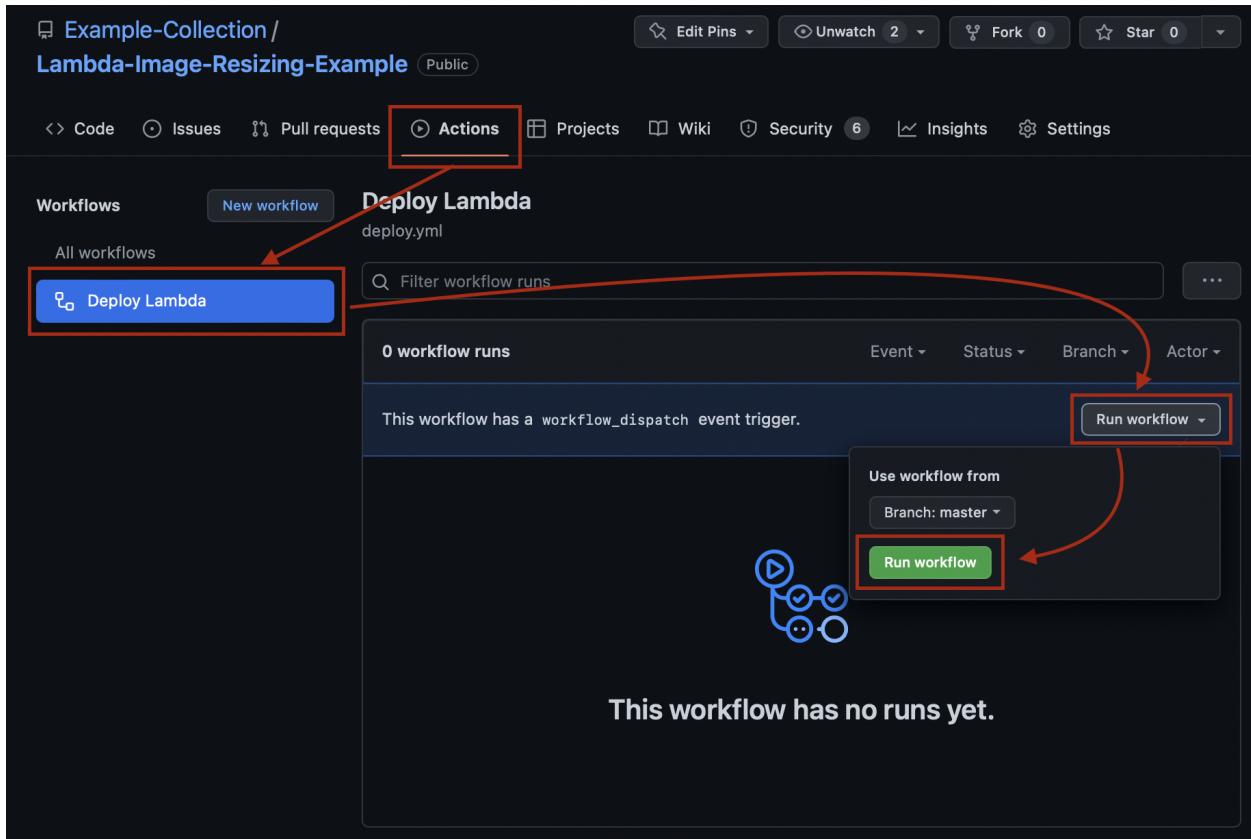
Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

[Manage your environments and add environment secrets](#)

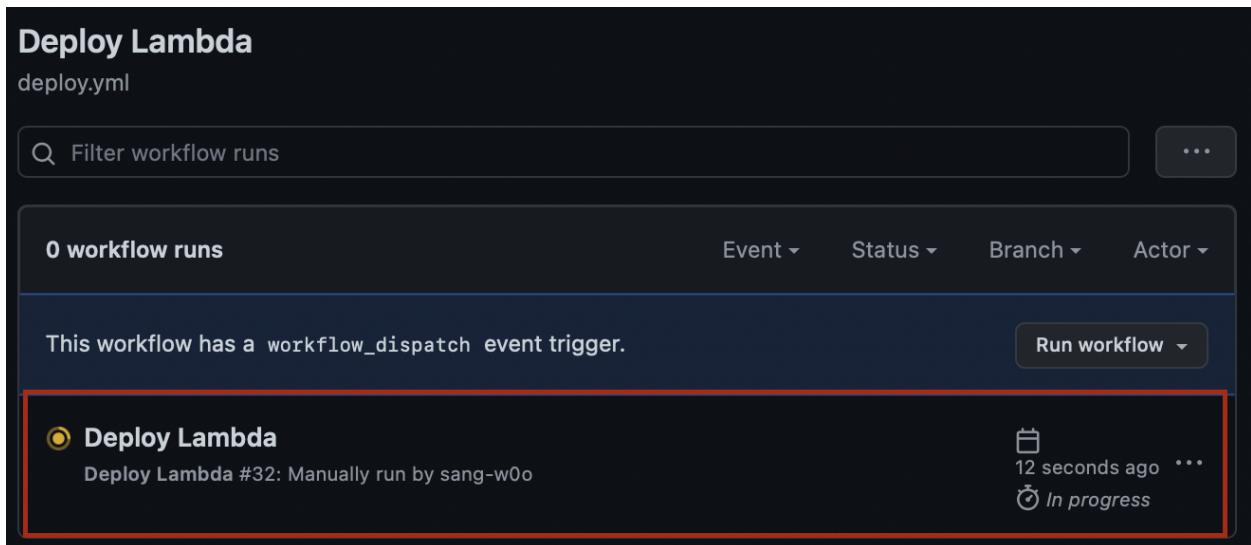
Repository secrets

AWS_ACCESS_KEY_ID	Updated 4 hours ago	Update	Remove
AWS_S3_BUCKET	Updated 4 hours ago	Update	Remove
AWS_SECRET_ACCESS_KEY	Updated 4 hours ago	Update	Remove

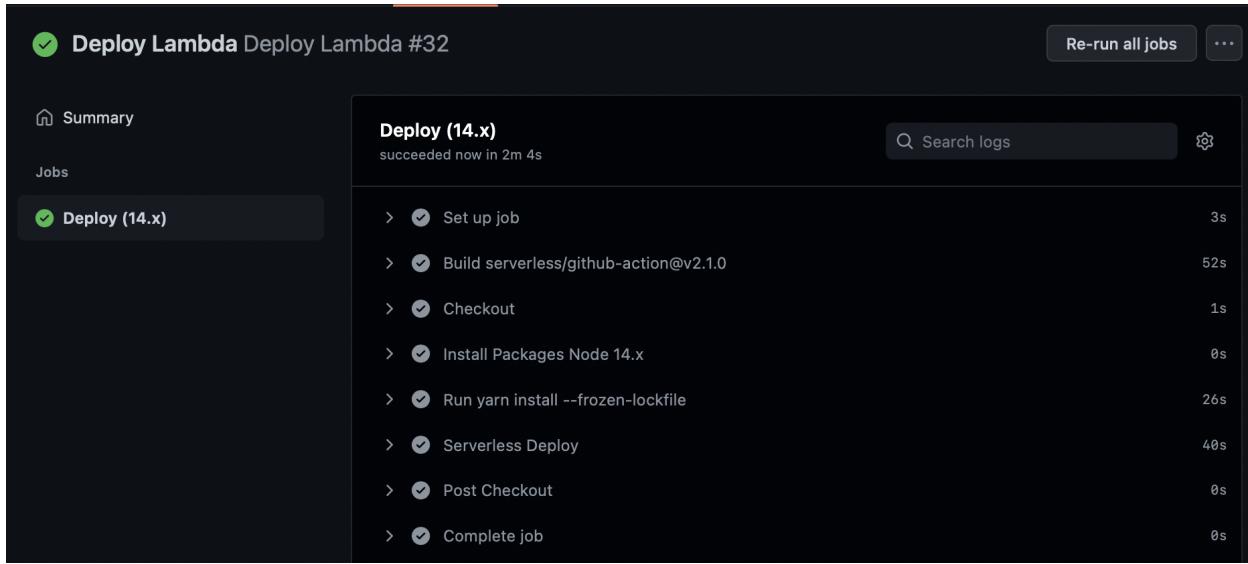
- 이제 드디어 배포를 수행할 단계가 되었습니다. 아래와 같이 **Actions** 탭으로 이동 후 좌측에 **Deploy Lambda**를 클릭한 후 **Run workflow**를 클릭해 배포를 합니다.



- 클릭 후 다시 Actions 탭을 클릭하면, 하나의 배포가 진행됨을 확인할 수 있습니다.



- 위를 클릭해 배포 과정을 보고, 배포가 완료되는 모습을 확인합니다.



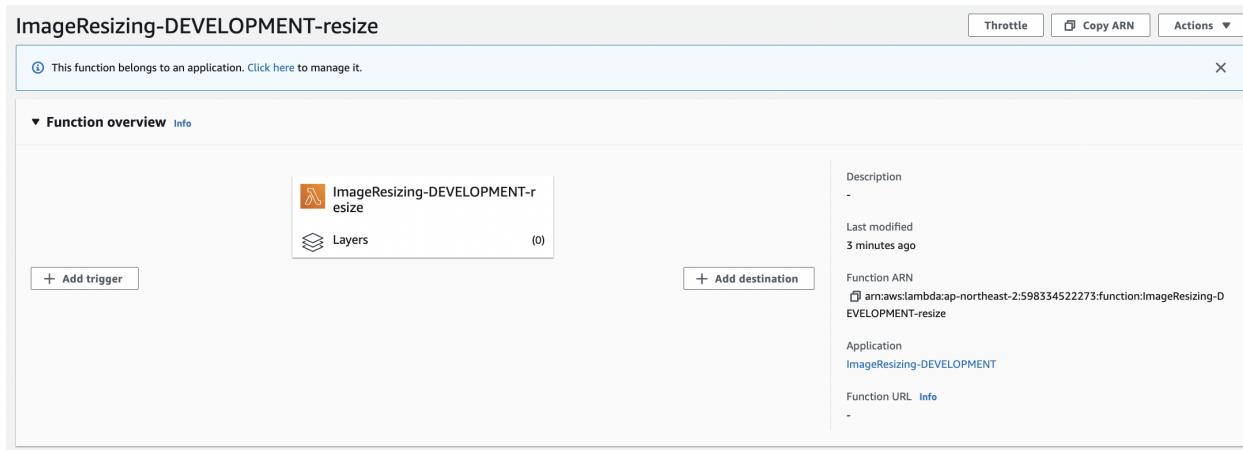
Lambda Console로 가서 배포된 Lambda 확인하기

- 이제 실제 AWS로 가서, 정상적으로 배포가 완료되었는지 확인해 봅니다.
- 다시 AWS Console로 이동해 먼저 Lambda를 검색해 Lambda Console로 이동합니다.
그러면 아래와 같이 하나의 function이 생성된 모습을 볼 수 있습니다.

The screenshot shows the AWS Lambda Functions list. The title indicates there are 7 functions. A search bar at the top right shows 'Matches: 1'. Below the search bar are filter options: 'Image' (selected), 'X', and 'Clear filters'. The main list displays one function entry:

<input type="checkbox"/>	Function name
<input type="checkbox"/>	ImageResizing-DEVELOPMENT-resize

- 해당 함수를 클릭해 상세 정보 페이지로 이동합니다. 그러면 아래처럼 화면에 나타납니다.



- 하지만 바로 아래에 **Code** 를 보면, 코드의 크기가 너무 커서 불러올 수 없다고 합니다. 그렇다면 실제 코드는 모두 어디에 저장될까요? 바로 Serverless가 배포를 하면서 Lambda 코드를 담기 위해 만든 S3 Bucket에 저장됩니다.
- (선택) S3 Console로 이동하면 아래와 같이 random string이 붙은 알 수 없는 bucket이 있습니다. 바로 이 곳에 람다가 수행하는 데 필요한 모든 코드가 저장됩니다.

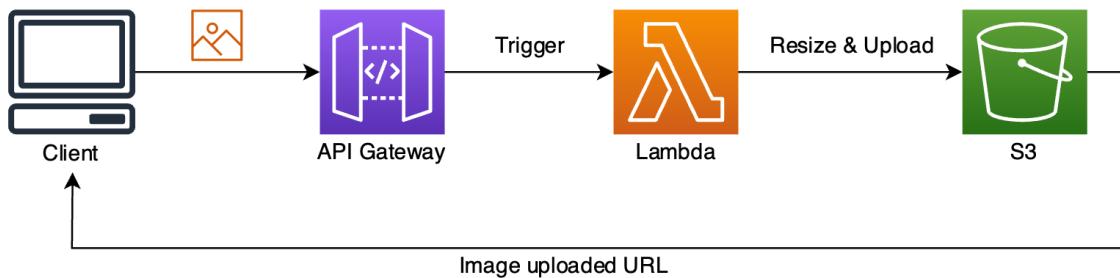
Buckets (14) Info				
C Copy ARN Empty Delete Create bucket				
<input type="text" value="imagere"/> X 1 match < 1 > ②				
Name	AWS Region	Access	Creation date	
imageresizing-developmen-serverlessdeploymentbuck-1jmwomr6mcymt	Asia Pacific (Seoul) ap-northeast-2	Objects can be public	June 23, 2022, 16:08:21 (UTC+09:00)	

Lambda에 Amazon API Gateway 연동하기

- 지금까지 한 내용을 정리하면, 아래와 같습니다.
 - IAM User 생성 및 압축된 이미지가 저장될 S3 Bucket 생성
 - Lambda를 위한 코드 작성
 - CD Pipeline 구성
 - Lambda를 AWS에 배포
- 다시 Lambda가 수행되는 과정과, 아래의 문구를 떠올려 봅시다.

어떤 무언가가 특정 작업을 하면 람다가 수행된다.

- 그리고 다시 이번에 만들 애플리케이션의 아키텍처 그림을 봅시다.



- 지금까지 한 내용으로만 봤을 때,
 - 어떤 무언가**: Client
 - 특정 작업**: API Gateway에 사진과 width, height를 담은 요청
 - 람다가 수행**: 이미지를 압축한 후 S3에 저장
- 즉, 클라이언트가 람다 함수를 **trigger** 할 수 있는 주체가 없습니다. 이를 위해 API Gateway를 Lambda와 연동시켜 봅시다.
- 다시 해당 Lambda의 화면으로 가서, **Add trigger** 버튼을 클릭합니다.

The screenshot shows the AWS Lambda function configuration page for 'ImageResizing-DEVELOPMENT-resize'. At the top, there are buttons for 'Throttle', 'Copy ARN', and 'Actions'. Below that, a message says 'This function belongs to an application. Click here to manage it.' and a search bar. Under 'Function overview', there's a 'Function name' field with 'ImageResizing-DEVELOPMENT-resize' and a 'Layers' section with '(0)'. On the right, there are sections for 'Description', 'Last modified', 'Function ARN', 'Application', and 'Function URL'. At the bottom left, a red box highlights the '+ Add trigger' button.

- 이후 첫 번째로 drop-down 에 나오는 `Amazon API Gateway` 를 클릭합니다.
이 drop-down에 있는 항목들만 봐도, 얼마나 많은 서비스가 Lambda와 연동될 수 있는지 체감할 수 있습니다.
- 아래와 같이 설정해줍니다.
 - Create a new API
 - `REST API` 선택
 - Security: `OPEN`
 - Additional Settings - Deployment Stage: `development`
 - Binary media types: `multipart/form-data`

Trigger configuration



API Gateway

api application-services aws serverless

▼

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports two types of RESTful APIs: HTTP APIs and REST APIs. [Learn more](#)

API

Create a new API or attach an existing one.

[Create an API](#)

API type

HTTP API

Create an HTTP API.

REST API

Create a REST API.

Security

Configure the security mechanism for your API endpoint.

[Open](#)

▼ Additional settings

API name

Choose a name for your API. API names don't need to be unique.

ImageResizing-DEVELOPMENT-resize-API

Deployment stage

The name of your API's deployment stage.

development

Cross-origin resource sharing (CORS)

CORS is required to call your API from a webpage that isn't hosted on the same domain. To enable CORS for a REST API, set the Access-Control-Allow-Origin header in the response object that you return from your function code.

Enable metrics and error logging

Record latency and error metrics in Amazon CloudWatch, and log errors in Amazon CloudWatch Logs. Standard CloudWatch and CloudWatch Logs pricing applies.

Binary media types

Specify response types that API Gateway should treat as binary data instead of text. To treat all responses as binary data, use */*. If you enter one or more specific types, you must also set the Content-Type header in the response object that you return from your function code.

multipart/form-data

[Remove](#)

[Add](#)

- 추가가 완료되면, 다시 해당 Lambda 함수를 봤을 때 API Gateway와 연동된 모습을 볼 수 있습니다.

`Configuration` 탭에 가면 `Triggers` 에 API Gateway가 있는데, 하나의 API Endpoint가 보입니다.

이 API Endpoint가 클라이언트가 사진과 width, height를 담아 요청을 보내는 주소입니다.

The screenshot shows the AWS Lambda function configuration page for 'ImageResizing-DEVELOPMENT-resize'. The 'Configuration' tab is selected. In the 'Destinations' section, there is one API Gateway trigger named 'ImageResizing-DEVELOPMENT-resize'. In the 'Triggers' section, there is one trigger named 'ImageResizing-DEVELOPMENT-resize' with the ARN: 'arn:aws:execute-api:ap-northeast-2:598334522273:nbqot5aipf�行:/imageResizing-DEVELOPMENT-resize'.

테스트 해보기

- 이제 구축한 Lambda와 API Gateway가 정상적으로 동작하는지 확인해볼 차례입니다.
- 이 링크에 접속해 아래의 웹사이트가 잘 뜨는지 확인합니다.

Image Uploader



API Gateway endpoint:

Before Uploading..

Name	Size
Not selected.	Not selected.

width in number height in number

After Uploading!

URL
Not uploaded
Uploaded Size
Not uploaded.

Supress rate: N/A

- 이제 API Gateway endpoint를 입력하고, **Select** 버튼을 클릭해 사진을 선택하고 width, height를 입력(선택) 후 **Upload** 를 클릭합니다.
그러면 S3에 업로드된 이미지의 URL과 압축된 후의 크기, 그리고 압축률을 확인할 수 있습니다.

Image Uploader



API Gateway endpoint: <https://nbqot5aipf.execute-api.us-east-1.amazonaws.com>

[Select](#) [Upload](#)

Before Uploading..

Name	Size
Community Builder Wa	809.81KB

width in number height in number

After Uploading!

URL
https://lambda-hands-on-ssu-roy.s3.ap-northeast-2.amazonaws.com
Uploaded Size
197.32KB

Supress rate: 75.63%

- 실제로 해당 이미지가 압축되었는지 확인하려면 압축된 이미지의 URL을 접속해 다운로드 해도 되고, 아래와 같이 S3에 가서 확인할 수 있습니다.

The screenshot shows the AWS S3 console interface for the bucket 'lambda-handson-ssu-roy'. The 'Objects' tab is selected, displaying one item: 'Community Builder Twitter 1500px.png'. The object is publicly accessible and has a size of 26.9 KB. The storage class is Standard. The table header includes columns for Name, Type, Last modified, Size, and Storage class. A red box highlights the entire table row for the uploaded file.

Name	Type	Last modified	Size	Storage class
Community Builder Twitter 1500px.png	png	June 23, 2022, 20:01:09 (UTC+09:00)	26.9 KB	Standard

리소스 정리하기 (CloudFormation으로 바꿔야댐!)

이번 실습에서 사용된 AWS 서비스들은 실행 시에만 과금되지만, 까먹고 두면 의도치 않게 비용이 부과될 수 있습니다. 따라서 실습 후에는 꼭 리소스를 제거해주세요!

1. API Gateway 삭제하기
 - a. Lambda 함수를 trigger하도록 설정되어 있는 API Gateway를 삭제합니다.
 - b. AWS Console에 API Gateway를 검색한 후, 해당 API Gateway를 삭제합니다.
2. Lambda 삭제하기
 - a. 만들어진 Lambda 함수를 삭제합니다.
3. S3 Bucket 삭제하기
 - a. **lambda-handson-닉네임** : S3 Bucket을 삭제하려면 우선 해당 Bucket 내에 있는 리소스들을 먼저 제거해야 합니다.

삭제할 Bucket 을 클릭한 후 `Empty` 를 눌러 비워줍니다.

그 후 `Delete` 를 눌러 삭제합니다.

- b. `imageresizing-development-serverlessdeploy**` : `Empty` 를 눌러 비워주기만 합니다.

압축된 이미지가 저장될 bucket, 그리고 serverless가 Lambda 코드를 저장하기 위해 만든 bucket, 총 2개를 모두 삭제해야 합니다!

4. Serverless framework가 만든 리소스들 제거하기

- 이번 세션에서 CD Pipeline에서 Serverless framework가 아래의 리소스들을 자동으로 만들어줬어요.
 - Lambda
 - S3(Lambda 코드가 저장될 곳)
 - Lambda 코드를 실행하기 위한 IAM 권한
- 이 리소스들은 CD Pipeline에서 CloudFormation을 통해 만들어졌기에 한 번에 쉽게 제거할 수 있어요. Serverless framework은 내부적으로 AWS CloudFormation을 사용해 지정된 리소스들을 배포해요.
- CloudFormation은 `stack` 이라는 단위로 배포를 하는데, 따라서 이번 세션에 사용된 모든 리소스들을 제거하려면 해당 CloudFormation stack을 제거해주면 돼요.

1. AWS Console에 CloudFormation을 검색한 후, 해당되는 stack을 선택해 삭제합니다.

여기서는 아래처럼 `ImageResizing-DEVELOPMENT` 또는 이와 비슷한 형식의 이름을 선택합니다.

CloudFormation > Stacks

Stacks (5)				
<input type="text" value="Filter by stack name"/> View nested				
Stack name	Status	Created time	De...	
ImageResizing-DEVELOPMENT	UPDATE_COMPLETE	2022-06-23 16:08:16 UTC+0900	The	
CdkWorkshopPipelineStack	CREATE_COMPLETE	2022-06-23 00:01:28 UTC+0900	-	
CDKToolkit	CREATE_COMPLETE	2022-06-21 19:13:11 UTC+0900	Thi	
PlanitUserSignupSlackNotifier-dev	UPDATE_COMPLETE	2022-04-04 00:42:36 UTC+0900	The	
GiftishowBatchApiCaller-dev	UPDATE_COMPLETE	2022-03-13 23:21:46 UTC+0900	The	

- 위처럼 **ImageResizing-DEVELOPMENT** 또는 이와 비슷한 형식의 이름을 클릭합니다.

CloudFormation > Stacks > ImageResizing-DEVELOPMENT

ImageResizing-DEVELOPMENT

Stack ID	Description
arn:aws:cloudformation:ap-northeast-2:598334522273:stack/ImageResizing-DEVELOPMENT/4109d270-f2c3-11ec-bda5-06192289ee9e	The AWS CloudFormation template for this Serverless application
Status	Status reason
UPDATE_COMPLETE	-
Root stack	Parent stack
-	-
Created time	Deleted time
2022-06-23 16:08:16 UTC+0900	-
Updated time	Last drift check time
2022-06-23 17:12:22 UTC+0900	-
Drift status	IAM role
NOT_CHECKED	-
Termination protection	
Disabled	-

- 우측 상단의 **delete** 버튼을 클릭해 해당 CloudFormation stack을 제거합니다.

CloudFormation은 free-tier 계정에 대해 월 1,000 건의 CloudFormation 배포를 무료로 제공해요. 따라서 CD Pipeline을 1,000번 이상 수행하지 않는 한 과금되지는 않아요. 단, 위에서 Lambda 함수를 저장하기 위한 S3를 직접 삭제했는데, 이 상태에서 다시 배포를 하면 Serverless가 배포를 실패해요. 즉, 이후에 배포가 다시 되지 않을 수 있는 가능성을 위해 혹시 삭제해주는 거예요.

