

1. Lambda 함수 코드 설명



AWS Lambda에서 assignment1이라는 새로운 함수를 만들어주었다. 시나리오를 간략히 설명하자면 환경변수를 이용하여 오른쪽으로 90도 회전, 위아래로 뒤집기, 크기 조정 세 가지 옵션 중 하나를 고르고, 버킷에 들어온 이미지에 반영하여 output 버킷에 결과물을 저장한다. 함수 코드와 그에 대한 설명은 다음과 같다.

```

1 // dependencies
2 var async = require('async');
3 var AWS = require('aws-sdk');
4 var gm = require('gm').subclass({ imageMagick: true }); // Enable ImageMagick integration.;
5 var util = require('util');
6
7 // get reference to S3 client
8 var s3 = new AWS.S3();
9
10 var transformFunc = process.env.TRANSFORM_FUNC;
11

```

이 부분은 실습 시간에 사용했던 예제 코드와 동일하다. 먼저 require 메서드를 이용하여 필요한 외부 모듈을 가져온다. 여기에서 가져오는 모듈 중 async, gm은 함수의 계층에 있다 (async, gm은 node_modules안에, imageMagick은 image-magick안에 존재) async는 비동기적으로 실행되는 javascript의 흐름 제어를 위한 것으로, 이번 과제에서는 그 중에서도 waterfall 메서드를 사용한다. aws-sdk는 aws에서 사용하는 개발 도구 집합이고, gm, imageMagick은 이미지 처리를 위한 모듈이다. util은 node.js의 보조적인 기능 중 유용한 기능을 모아 놓은 모듈로, 작성한 함수에서 util.inspect()를 이용하기 때문에 가져왔다.

var s3 = new AWS.S3()는 s3 버킷을 사용하기 위해 s3 service object를 만드는 부분이고, var transformFunc = process.env.TRANSFORM_FUNC은 함수의 환경변수를 받아오는 부분이다. 작성한 함수가 이미지 뒤집기, 회전, 사이즈 조정을 할 수 있는데 그 중에서도 어떤 작업을 할지를 환경 변수로 받는다.

```

exports.handler = function(event, context, callback) {
  // Read options from the event.
  console.log("Reading options from event:\n", util.inspect(event, {depth: 5}));
  var srcBucket = event.Records[0].s3.bucket.name;
  // Object key may have spaces or unicode non-ASCII characters.
  var srcKey = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, " "));
  var dstBucket = srcBucket + "-output";
  var dstKey   = "output-" + srcKey;

  // Sanity check: validate that source and destination are different buckets.
  if (srcBucket == dstBucket) {
    callback("Source and destination buckets are the same.");
    return;
  }

  // Infer the image type.
  console.log("Finding out the image type");
  var typeMatch = srcKey.match(/\.([\^.]*)$/);
  if (!typeMatch) {
    callback("Could not determine the image type.");
    return;
  }
  var imageType = typeMatch[1];
  console.log("To keep it simple, we will accept only png or jpeg image types");
  if (imageType != "jpeg" && imageType != "png") {
    callback('Unsupported image type: ${imageType}');
    return;
  }
}

```

handler는 이벤트 발생 시 실행되는 부분이다. console.log로 event에 대한 옵션을 출력한다. 이 때 depth:5는 event 객체를 포매팅할 때 inspect가 5번 재귀한다는 말이다. 그 뒤 srcBucket에 이벤트가 발생한 버킷의 이름을 받아오고, srcKey에는 버킷 내 해당 파일명을 받아오는데 이 때 +기호는 공백으로 변환해서 받는다. 그리고 결과물을 저장할 bucket과 결과물의 파일명을 output을 붙여 지정해준다. 이벤트 발생 버킷과 결과물 저장 버킷이 같을 경우 함수가 무한히 실행될 수 있으므로 다른지 검사해준다. 그 후에는 이미지 타입을 검사하고 지원하는 확장자가 아닐 경우 callback을 호출한다. match는 괄호 안의 정규식과 일치하는 문자열이 있으면 반환해주는데, 여기에서는 확장자가 반환된다. 확장자가 없거나, 지원하는 종류(jpeg, png)가 아닐 경우 리턴한다.

```

44  async.waterfall([
45    function download(next) {
46      // Download the image from S3 into a buffer.
47      s3.getObject({
48        Bucket: srcBucket,
49        Key: srcKey
50      },
51      next);
52    },
53    function transform(response, next) {

```

앞에서도 말했듯이 async.waterfall은 함수를 순차적으로 실행하기 위한 것이다. 매개변수로 들어온 메소드들을 차례로 실행하는데, 앞의 메소드가 실행이 끝나면 그 결과값을 넘겨주고 다음 메소드가 실행되는 방식이다. 여기서는 S3에서 이미지를 가져오는 download 함수가 제일 먼저 실행되고 있다.

```

53     function transform(response, next) {
54         console.log("Here we have three option - flip, rotate and resize");
55         console.log("Currently we have got the option of "+ transformFunc+".");
56         switch(transformFunc) {
57             case "flip":
58                 console.log("flip");
59                 gm(response.Body).flip()
60                     .toBuffer(imageType, function(err, buffer) {
61                     if (err) {
62                         next(err);
63                     } else {
64                         next(null, response.ContentType, buffer);
65                     }
66                 });
67                 break;

```

그 다음은 이미지를 처리하는 함수이다. console.log으로 어떤 옵션이 있고 그 중 어떤 것을 선택했는지 (환경변수) 보여준다. 수업시간에 다룬 예시와는 다르게 과제로 구현한 함수에서는 flip, rotate, resize의 세 가지 옵션 중 하나를 선택하도록 했다. switch-case문을 이용하여 선택한 옵션에 따라 다르게 실행한다. 먼저 flip을 선택한 경우 이미지를 위아래로 뒤집어준다. 앞 메소드(download)에서 받아온 이미지(response.Body)를 gm 모듈의 flip메소드를 이용해서 뒤집어주고, toBuffer를 이용해서 결과물을 버퍼로 출력한다. 이 때 에러가 발생했을 경우 다음 함수에 에러를 넘기고 그렇지 않을 경우 contentType과 버퍼를 넘겨준다.

```

68     case "rotate":
69         console.log("roate");
70         gm(response.Body).rotate('black', 90)
71             .toBuffer(imageType, function(err, buffer) {
72             if (err) {
73                 next(err);
74             } else {
75                 next(null, response.ContentType, buffer);
76             }
77             });
78         break;

```

회전을 선택한 경우 실행되는 부분이다. flip 대신 rotate (오른쪽으로 90도 회전, 회전하고 남는 공간이 생길 경우 검은색을 채움)이 들어간 것을 제외하면 위의 flip과 동일하다.

```

79     case "resize":
80         console.log("resize");
81         gm(response.Body).resize(240, 240)
82             .toBuffer(imageType, function(err, buffer) {
83             if (err) {
84                 next(err);
85             } else {
86                 next(null, response.ContentType, buffer);
87             }
88             });
89         break;
90     default:
91         console.log("None of the three options were selected");
92         callback("None of the three options were selected");
93         return;
94     }
95 },

```

resize를 선택한 경우 실행되는 부분이다. 가로 세로 길이를 240으로 조정하는 부분을 제외하면 위에서 나온 flip, rotate와 동일하다. 그 밑은 flip, rotate, resize 중에서 선택하지 않은 경우 경고메

시지를 띄우고 리턴하는 예외처리 부분이다.

```

96     function upload(contentType, data, next) {
97         // Stream the transformed image to a different S3 bucket.
98         s3.putObject({
99             Bucket: dstBucket,
100             Key: dstKey,
101             Body: data,
102             ContentType: contentType
103         },
104         next);
105     }
106     ], function (err) {
107         if (err) {
108             console.error(
109                 'Unable to resize ' + srcBucket + '/' + srcKey +
110                 ' and upload to ' + dstBucket + '/' + dstKey +
111                 ' due to an error: ' + err
112             );
113         } else {
114             console.log(
115                 'Successfully resized ' + srcBucket + '/' + srcKey +
116                 ' and uploaded to ' + dstBucket + '/' + dstKey
117             );
118         }
119         callback(null, "message");
120     }
121 }
122 );
123 };
```

이미지 처리가 끝난 뒤 실행되는 부분이다. s3.putObject를 이용하여 output bucket에 결과물을 업로드 한다. 에러가 발생한 경우 console.error로 에러가 발생했음을 알리고, 그렇지 않은 경우 성공적으로 저장했다는 로그메시지를 남긴다.

2. 이벤트 트리거와 레이어 설정

버킷 (4) Info				
버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기				
	ARN 복사	비어 있음	삭제	버킷 만들기
<input type="text" value="이름으로 버킷 찾기"/>				
이름	AWS 리전	액세스	생성 날짜	
<input type="radio"/> cloud-computing-assignment1 ✓	미국 동부(버지니아 북부) us-east-1	객체를 퍼블릭으로 설정할 수 있음	2021. 10. 27. pm 12:01:10 PM KST	
<input type="radio"/> cloud-computing-assignment1-output ✓	미국 동부(버지니아 북부) us-east-1	객체를 퍼블릭으로 설정할 수 있음	2021. 10. 27. pm 12:01:43 PM KST	
<input type="radio"/> ke-s3-to-lambda	미국 동부(버지니아 북부) us-east-1	객체를 퍼블릭으로 설정할 수 있음	2021. 10. 26. pm 11:11:35 PM KST	
<input type="radio"/> ke-s3-to-lambda-output	미국 동부(버지니아 북부) us-east-1	객체를 퍼블릭으로 설정할 수 있음	2021. 10. 26. pm 11:12:04 PM KST	

우선 위와 같이 이미지를 업로드할 cloud-computing-assignment1 버킷과 lambda 함수 실행 후 나온 결과물이 저장될 cloud-computing-assignment-output 버킷을 만들었다.

추가 트리거

트리거 구성

버킷
이벤트 소스의 역할을 하는 S3 버킷을 선택하십시오. 버킷은 함수와 같은 리전에 있어야 합니다.
cloud-computing-assignment1

이벤트 유형
Lambda 함수를 트리거하려는 이벤트를 선택합니다. 필요에 따라 이벤트의 접두사 또는 접미사를 설정할 수 있습니다. 하지만 각 버킷에서 개별 이벤트는 접두사나 접미사가 겹쳐서 객체 키가 동일해질 수 있는 구성을 여러 개 가질 수 없습니다.
모든 객체 생성 이벤트

접두사 - 선택 사항
필요할 경우, 일치하는 문자로 시작하는 키를 사용하여 객체에 대해 일부를 제한하려는 단일 접두사를 입력합니다.
images/

접미사 - 선택 사항
필요할 경우, 일치하는 문자로 끝나는 키를 사용하여 객체에 대해 일부를 제한하려는 단일 접미사를 입력합니다.

트리거 (1)

트리거

S3: cloud-computing-assignment1
arn:aws:s3:::cloud-computing-assignment1
세부 정보

트리거는 cloud-computing-assignment1 버킷에 새로운 객체가 생성될 경우 작동되도록 설정해 놓았다.

계층 (2)

이름	버전	호환 런타임	호환 아키텍처	버전 ARN
image_magicks	2	nodejs14.x	-	arn:aws:lambda:us-east-1:940089762753:layer:image_magicks:2
node_modules	2	nodejs14.x	-	arn:aws:lambda:us-east-1:940089762753:layer:node_modules:2

런타임 설정

런타임
Node.js 14.x

핸들러
index.handler

아키텍처
x86_64

계층

병합 주문	이름	Layer 버전	호환 런타임	호환 아키텍처
1	node_modules	2	nodejs14.x	-
2	image_magicks	2	nodejs14.x	-

lambda함수가 하는 일이 이미지 처리이기 때문에 레이어는 실습시간에 사용했던 것과 같은 zip 파일을 업로드하여 만들었다. 앞에서도 설명했듯이 node_modules에는 함수 흐름제어를 위한 async와 이미지 처리를 위한 gm이 있고, image_magicks에는 이미지 처리를 위한 image_magicks 라이브러리가 있다. 단, 새로 만든 함수는 런타임이 Node.js 12.x가 아닌 Node.js 14.x라서 기존 계층에 버전2를 만들어서 사용하였다.

3. 이벤트 발생 시 나타난 결과물 확인



The screenshot shows the AWS Cloud Computing Assignment 1 console. The '객체' (Objects) tab is selected, displaying a table of objects. The table has columns for '이름' (Name), '유형' (Type), '마지막 수정' (Last Modified), '크기' (Size), and '스토리지 클래스' (Storage Class). One object is listed: 'apeach1.png' (png, 62.6KB, Standard). Below the table, the image is displayed on a black background.

이름	유형	마지막 수정	크기	스토리지 클래스
apeach1.png	png	2021. 10. 27. pm 6:13:19 PM KST	62.6KB	Standard

환경 변수 (1) 편집

아래의 환경 변수는 기본 Lambda 서비스 키를 이용해 저장 중 암호화됩니다.

키	값
TRANSFORM_FUNC	rotate

위와 같은 이미지를 지정한 버킷에 업로드하고 환경변수를 rotate로 설정했을 때, 결과는 다음과 같다.

cloud-computing-assignment1-output Info

객체 | 속성 | 권한 | 지표 | 관리 | 액세스 지점

객체 (1)

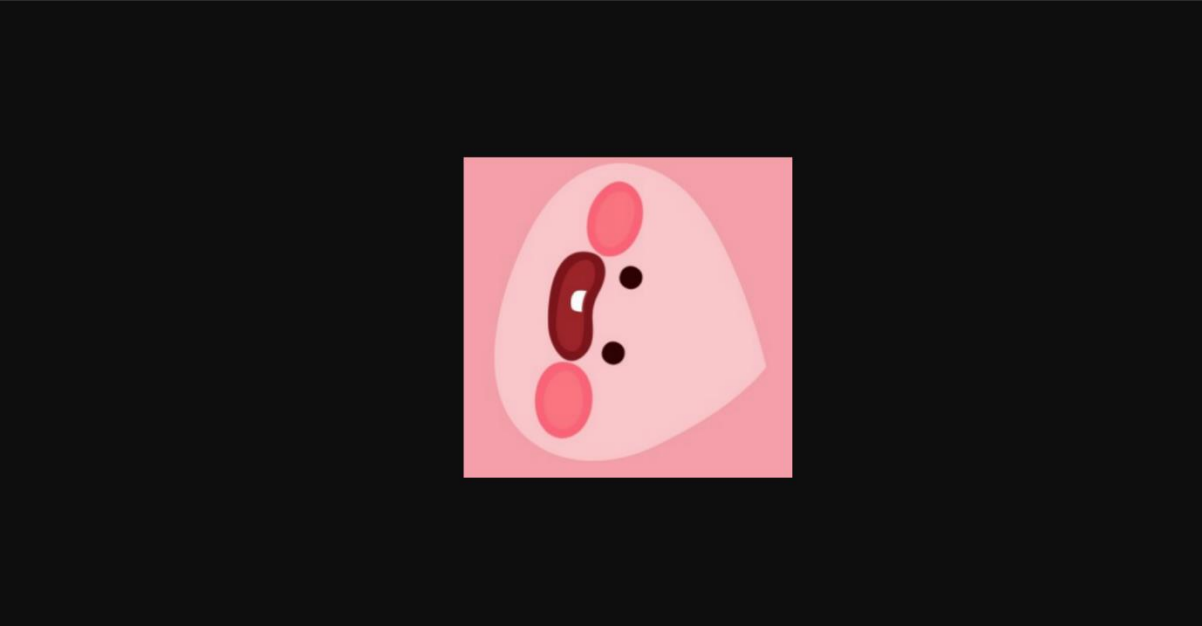
객체는 Amazon S3에 저장되어 있는 기본 엔터티입니다. [Amazon S3 인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. [자세히 알아보기](#)

🔄
S3 URI 복사
URL 복사
다운로드
열기
삭제
작업 ▼

폴더 만들기
업로드

🔍 접두사로 객체 찾기

<input type="checkbox"/>	이름 ▲	유형 ▼	마지막 수정 ▼	크기 ▼	스토리지 클래스 ▼
<input type="checkbox"/>	output-apeach1.png	png	2021. 10. 27. pm 6:13:22 PM KST	48.1KB	Standard

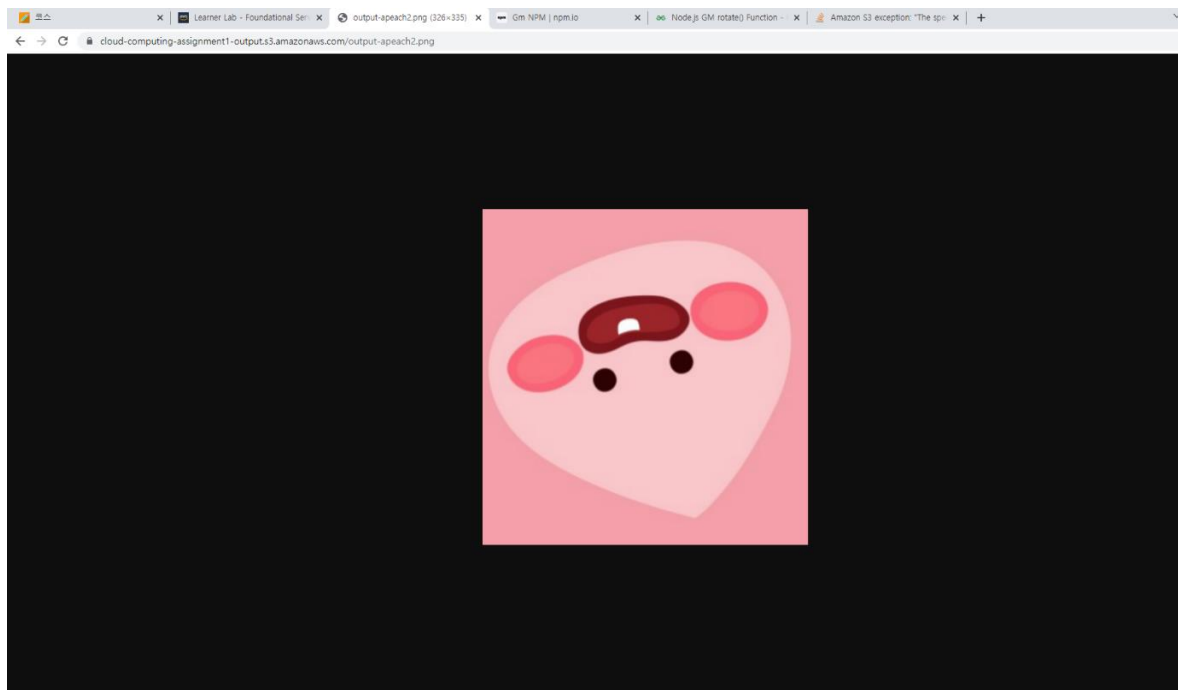


Output 버킷에 이미지가 오른쪽으로 90도 회전된 채로 저장된 것을 확인할 수 있다.

환경 변수 (1) 편집

아래의 환경 변수는 기본 Lambda 서비스 키를 이용해 저장 중 암호화됩니다.

키	값
TRANSFORM_FUNC	flip

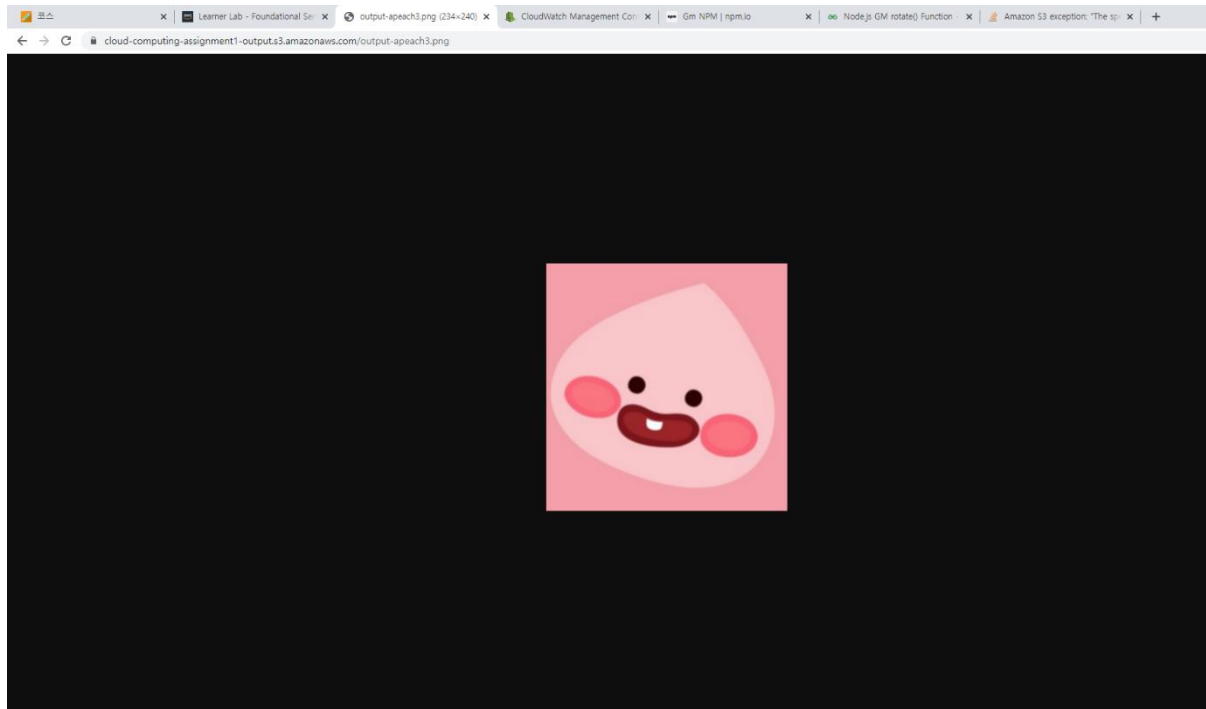


옵션을 flip으로 설정한 경우 이미지가 위아래로 뒤집혀 저장되어 있다.

환경 변수 (1) 편집

아래의 환경 변수는 기본 Lambda 서비스 키를 이용해 저장 중 암호화됩니다.

키	값
TRANSFORM_FUNC	resize



옵션을 `resize`로 설정한 경우 크기가 가로 세로 240에 맞게 저장된 것을 확인할 수 있다.




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](#)

[새로고침](#) [S3 URI 복사](#) [URL 복사](#) [다운로드](#) [열기](#) [삭제](#) [작업 ▼](#)

[폴더 만들기](#) [업로드](#)

Q 접두사로 객체 찾기

<input type="checkbox"/>	이름 ▲	유형 ▼	마지막 수정 ▼	크기 ▼	스토리지 클래스 ▼
<input type="checkbox"/>	 apeach1.png	png	2021. 10. 27. pm 6:13:19 PM KST	62.6KB	Standard
<input type="checkbox"/>	 apeach2.png	png	2021. 10. 27. pm 6:21:12 PM KST	62.6KB	Standard
<input type="checkbox"/>	 apeach3.png	png	2021. 10. 27. pm 6:23:49 PM KST	62.6KB	Standard

최종적으로 input bucket에 들어있는 파일들이다. 1,2,3 모두 파일 명만 다르고 동일한 이미지를 사용하였다.

객체

속성

권한

지표

관리

액세스 지점

객체 (3)

객체는 Amazon S3에 저장되어 있는 기본 엔티티입니다. [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](#)

🔄

S3 URI 복사

URL 복사

다운로드

열기

삭제




작업 ▼

폴더 만들기

업로드

🔍 점두사로 객체 찾기

< 1 > ⚙️

<input type="checkbox"/>	이름 ▲	유형 ▼	마지막 수정 ▼	크기 ▼	스토리지 클래스 ▼
<input type="checkbox"/>	 output-apeach1.png	png	2021. 10. 27. pm 6:13:22 PM KST	48.1KB	Standard
<input type="checkbox"/>	 output-apeach2.png	png	2021. 10. 27. pm 6:21:17 PM KST	47.6KB	Standard
<input type="checkbox"/>	 output-apeach3.png	png	2021. 10. 27. pm 6:23:53 PM KST	27.8KB	Standard

output 버킷에도 결과물들이 잘 남아있는 것을 확인할 수 있다.