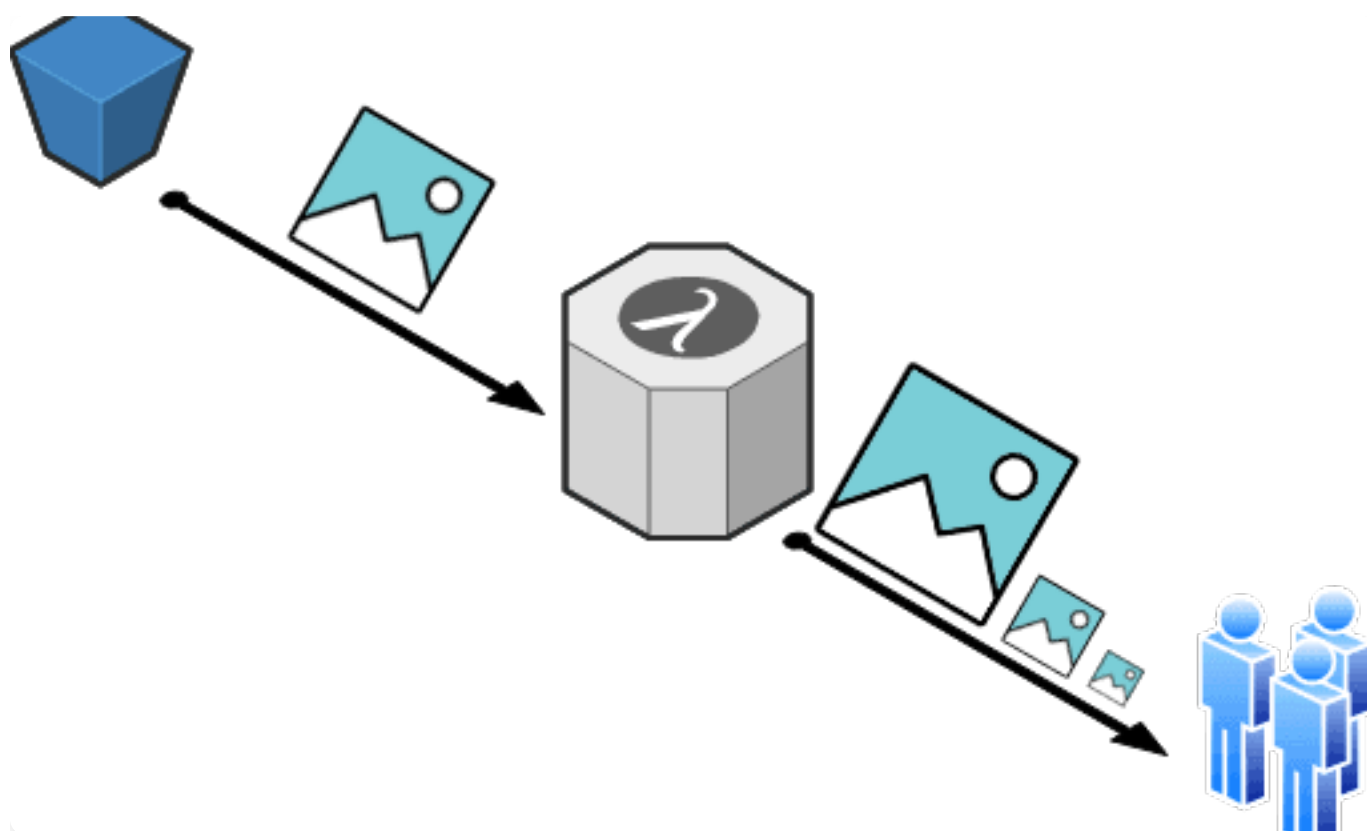


## Image resizing on the fly with AWS Lambda, API Gateway, and S3 Storage



DevOps

Back-end



Mohammed SAFI

2019-08-20 | 5 min read



In this article, we are going to talk about creating different image sizes after requesting the original one with size parameters (width/height) from AWS S3 storage.

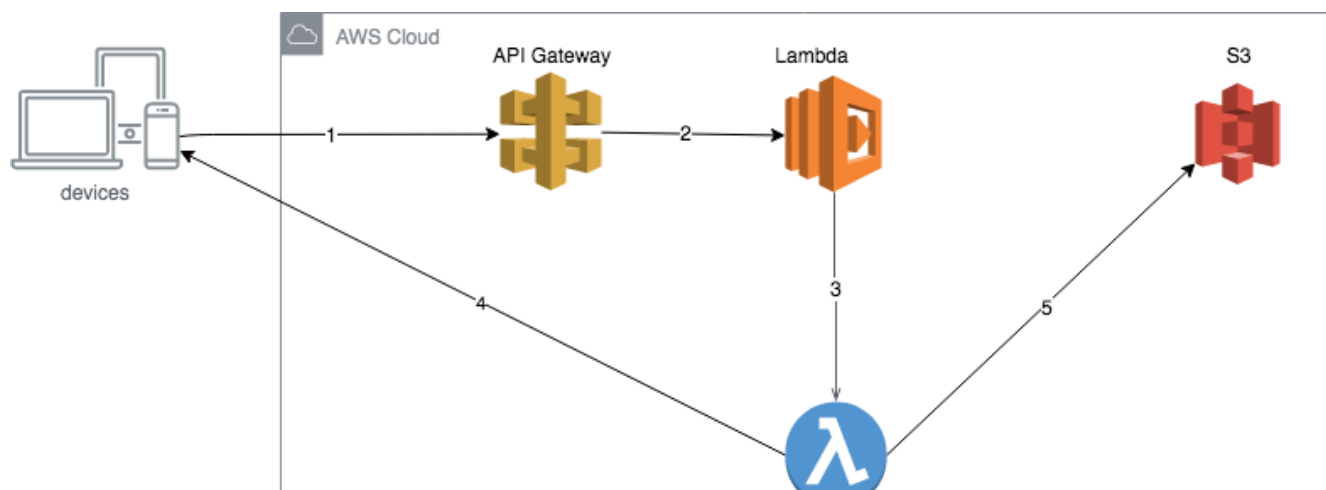
but the problem here is that they can slow down the download speed of the entire web page. Imagine that you have a blogs site, and most readers are viewing your site on their phones, in this case, they don't need the high-resolution pictures, this may slow downloading the whole content which is not good for your readers, and some of them are reading from their laptops where they have a better connection and a better screen, and so will appreciate the higher-quality images.

The main goal of this post is to show you how you can use only the AWS lambda function, which means no servers (EC2 instances) is needed, to create an image resizing task. When an image gets loaded from the s3 bucket through an API Gateway endpoint, the lambda function will be triggered which will resize the image (if the image with the specified size does not exist in the bucket) based on the specified size, and it will return the resized image URL.

## Table of contents

1. Architecture
2. Lambda Function Configuration
3. Upload Source Code
4. API Gateway Configuration
5. Testing

### Architecture



parameters(width/height) in order to receive the s3 URL of the resized image,

2. The API Gateway request will trigger a lambda function,
3. that will check if the image with the given size exists or not,
4. if it does this function will return the image s3 URL,
5. else it will take the original image from s3, resize it with the given size and return back the image s3 URL.

## Lambda Function Configuration

First of all, we assume that you already have an s3 bucket and API gateway instance created, so our images resizing lambda function will need at the minimum these configurations:

- Execution role: AWS S3 Full Access, and AWS Basic Execution Role

Roles > resize\_s3\_images-role-v4ygd336

### Summary

Delete role

Role ARN	<a href="#">[redacted]</a>
Role description	<a href="#">Edit</a>
Instance Profile ARNs	<a href="#">[redacted]</a>
Path	/service-role/
Creation time	2019-07-25 16:27 UTC+0100
Maximum CLI/API session duration	1 hour <a href="#">Edit</a>

Permissions Trust relationships Tags Access Advisor Revoke sessions

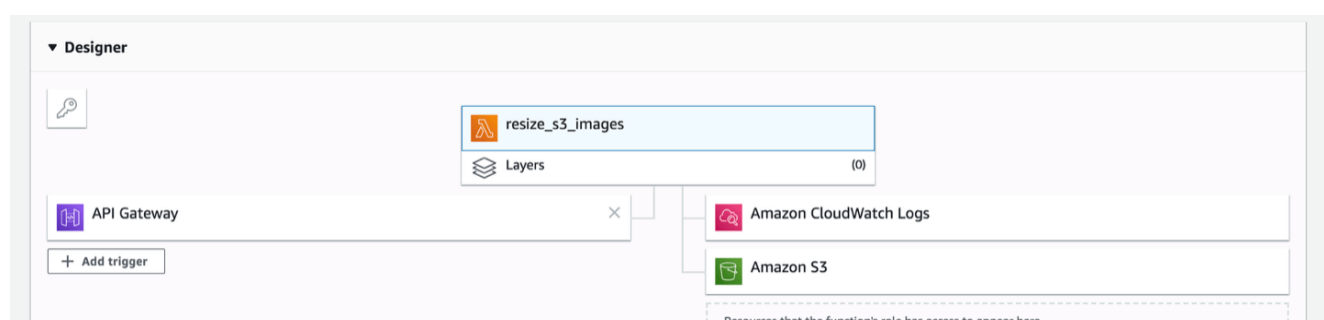
Permissions policies (2 policies applied)

Attach policies

[Add inline policy](#)

Policy name	Policy type	
AmazonS3FullAccess	AWS managed policy	<a href="#">✕</a>
AWSLambdaBasicExecutionRole-[redacted]	Managed policy	<a href="#">✕</a>

- Basic settings: Timeout 20 sec
- Add an API Gateway trigger to the lambda function with the existing API



We are going to use Python programming language to build our image resizing lambda function,

*“SOURCE CODE: [https://github.com/obytes/resize\\_s3\\_images](https://github.com/obytes/resize_s3_images)”*

Let us start by explaining the source code:

```
def lambda_handler(event, context):
    key = event['queryStringParameters'].get('key', None)
    size = event['queryStringParameters'].get('size', None)

    image_s3_url = resize_image(os.environ['BUCKET'], key, size)

    return {
        'statusCode': 301,
        'body': image_s3_url
    }
```

The `lambda_handler` will get called when our lambda function got triggered by a new incoming request for an image to be resized. It gets the `key` and `size` arguments from the request path, then it calls the `resize_image` function with the previous arguments and the s3 bucket name in order to get where our new resized image is.

Now let's get deeply under our `resize_image` function, first it gets the original from the given s3 bucket if exists:

```
s3 = boto3.resource('s3')
obj = s3.Object(
    bucket_name=bucket_name,
    key=key,
)
obj_body = obj.get()['Body'].read()
```

Second, it resizes the image using `PILLOW` package:

```
img = Image.open(BytesIO(obj_body))
img = img.resize((int(size_split[0]), int(size_split[1])), PIL.Image.ANTIALIAS)
buffer = BytesIO()
img.save(buffer, 'JPEG')
buffer.seek(0)
```

Finally, it uploads the resized image back to s3, and returns its URL:

```
resized_key="{size}_{key}".format(size=size, key=key)
obj = s3.Object(
    bucket_name=bucket_name,
    key=resized_key
)
obj.put(Body=buffer, ContentType='image/jpeg')

return "https://{bucket}.s3.amazonaws.com/{resized_key}".format(bucket=bucket,
```

If we try to create a deployment package and upload it to lambda like it is documented in aws [here](#), it is not going to work. As we can see, it depends on `PIL`, a library for manipulating images that is not on the Lambda environment nor on the standard Python Library. How can we fix this? Using Python Wheels! **Wheels** are the new standard of python distribution. It creates Wheels packages ready to run everywhere. Since Lambda runs on the Amazon Linux Distribution, we can try [this](#) version. After downloading it, extract on the same directory as the `resize_s3_images.py` file and zip it all together!

```
$ unzip Pillow-6.1.0-cp36-cp36m-manylinux1_x86_64.whl && rm Pillow-6.1.0-cp36-cp36m-manylinux1_x86_64.whl
$ zip -r resize_s3_images.zip .
```

On the AWS Lambda dashboard upload the zipped file `resize_s3_images.zip`

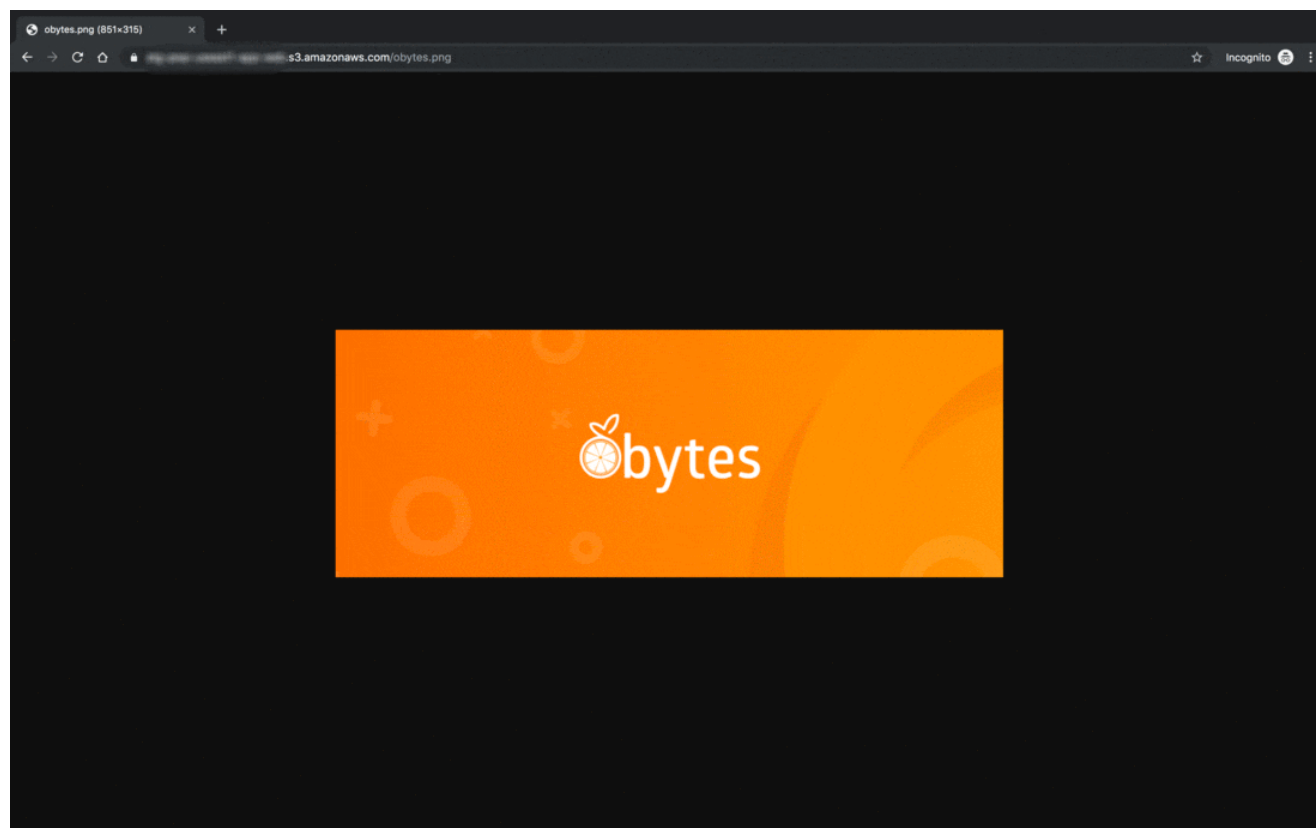
## API Gateway Configuration

Next, it is time to setup an api gateway endpoint:

- Root path `/`
- Resource in root path called `/resize`
- GET method in the previous resource that should be integrated with our lambda function we created before
- Enable CORS for the previous http GET method

You should deploy the API in order to make the whole changes live

Now we have built everything we need for our image resizing lambda function and let's test it using [cURL](#).



Share article



More articles



HIRE US



updating and delete

Back-end

5 min read

Youssef Naimi

2016-09-06

## Setup your Django app with Docker and Compose.

Docker provides an integrated technology suite that enables development and IT operations teams to build, ship

Back-end

DevOps

5 min read

Sign up to our newsletter !



Your email address

Subscribe





HIRE US



Our mission and ambition is to challenge the status quo, by doing things differently we nurture our love for craft and technology allowing us to create the unexpected.

[Linkedin](#) [Twitter](#)