I recently forked a project from another developer (see CREDIT below) and rebuilt a mini serverless app to support dynamically resizing images with S3 and API Gateway.

The project is available here: https://github.com/litwicki/lambda-resize-image

lambda-resize-image

An AWS Lambda Function to resize images automatically with API Gateway and S3 for imagemagick tasks. When an image is called on AWS Api Gateway, this package will resize it and send it to the S3.

Requirements

- Node.js – AWS Lambda supports versions of **10.20.1** or above (Recommended: **12.X**).

Binary API Gateway Settings

You must have AWS CLI installed to execute a command from your console:

```
aws apigateway update-integration-response --rest-api-id <API_ID> --resource-id <RESOURCE_ID> --http-method GET --status-code 200 --patch-operations '[{"op" : "replace", "path" : "/contentHandling", "value" : "CONVERT_TO_BINARY"}]'
```

In API GW -> Settings -> Binary Media Types and add:

```
*/*
```

Description

The combination of API Gateway and Lambda is very powerful. It allows you to build some complex functionalities without maintaining any virtual machines yourself. Lambda can be

hooked up to many other (AWS) Services including S3. That's why I decided to build an AWS Lambda Function to resize images automatically with API Gateway and S3 for imagemagick tasks. When an image is called on AWS S3 bucket (via API Gateway), this package will resize it and send it to the S3 before redirecting you to the new path of the image (aws bucket url or CDN).

Features

- Use Serverless Framework
- Use Serverless Webpack
- Use Serverless Offline
- The image conversion endpoint by API Gateway or cloudfront URL.

Installation

First, add Serverless globally:

```
npm install -g serverless
```

Then, clone the repository into your local environment:

```
git clone https://github.com/litwicki/lambda-resize-image
cd lambda-resize-image
npm install
```

Setup your environment configuration

```
cp .env.example env.yml
```

AWS credentials

To run local development you also might need to configure you aws credentials, or you can

set them to what I've shown below.

Deploy to Amazon AWS

You can also check if you have everything installed in the correct way:

```
$ serverless
```

To deploy from your environment to Amazon AWS, you must:

`$ serverless deploy --stage dev` or `$ serverless deploy --stage prod` for production configurations.

Environment variables

- BUCKET – AWS S3 bucket. (required)

Configure Serverless

You need to configure serverless with your credentials so you can deploy your stages.

You will need your `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` regardless of which method you prefer.

Local development

First you'll need to install imagemagick locally, depending on which OS you use:

MAC OS

```
$ brew install imagemagick
$ brew install ghostscript
```

LINUX

```
$ sudo apt-get install imagemagick
```

WINDOWS

You'll need to install the binary from here and follow the directions.

Install Serverless & Packages

```
$ npm i -g serverless && npm i
```

Note that you will need to be into the root repository. The last command (4.) will spin up an serverless-offline version of an API Gateway, that will simulate the real one. Once it is running, you can see all the requests on your command line.

Example request

http://localhost:3000<YOUR_KEYNAME_TO_IMAGE>?width=<WIDTH>&height=<HEIGHT>

Credit

This project is a fork of apoca/lambda-resize-image, and all credit is due to apoca.