**Tools used:**

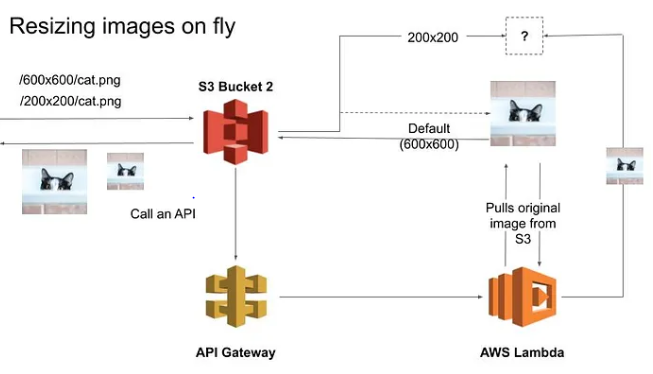
Amazon web services (lambda and API gateway)

Sharp library.

**Language:**

JavaScript

**Architecture:**



Function:

* Compress the newly loaded images.
* Compress the existing images(bigger challenge)
* Resize the images which caters the screen requirement.

Steps:

* Create Two S3 buckets with one image in one bucket and other bucket empty. Upload Image to Resize.
* Create Bucket Policy and Update as below

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:PutLogEvents",

"logs:CreateLogGroup",

"logs:CreateLogStream"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": ["s3:GetObject"],

"Resource": "arn:aws:s3:::BUCKET\_NAME/\*"

},

{

"Effect": "Allow",

"Action": ["s3:PutObject"],

"Resource": "arn:aws:s3:::DEST\_BUCKET/\*"

}

]

}

* Now Create a Lambda function from scratch using any runtime for example in python we should install boto3 module.
* We will create a Blank Node.js Lambda function and add index.html in Code space in Lambda

import {

S3Client,

GetObjectCommand,

PutObjectCommand,

} from "@aws-sdk/client-s3";

import sharp from "sharp";

const S3 = new S3Client();

const DEST\_BUCKET = process.env.DEST\_BUCKET;

const THUMBNAIL\_WIDTH = 200; // px

const SUPPORTED\_FORMATS = {

jpg: true,

jpeg: true,

png: true,

};

export const handler = async (event, context) => {

const { eventTime, s3 } = event.Records[0];

const srcBucket = s3.bucket.name;

// Object key may have spaces or unicode non-ASCII characters

const srcKey = decodeURIComponent(s3.object.key.replace(/\+/g, " "));

const ext = srcKey.replace(/^.\*\./, "").toLowerCase();

console.log(`${eventTime} - ${srcBucket}/${srcKey}`);

if (!SUPPORTED\_FORMATS[ext]) {

console.log(`ERROR: Unsupported file type (${ext})`);

return;

}

// Get the image from the source bucket

try {

const { Body, ContentType } = await S3.send(

new GetObjectCommand({

Bucket: srcBucket,

Key: srcKey,

})

);

const image = await Body.transformToByteArray();

// resize image

const outputBuffer = await sharp(image).resize(THUMBNAIL\_WIDTH).toBuffer();

// store new image in the destination bucket

await S3.send(

new PutObjectCommand({

Bucket: DEST\_BUCKET,

Key: srcKey,

Body: outputBuffer,

ContentType,

})

);

const message = `Successfully resized ${srcBucket}/${srcKey} and uploaded to ${DEST\_BUCKET}/${srcKey}`;

console.log(message);

return {

statusCode: 200,

body: message,

};

} catch (error) {

console.log(error);

}

};

* Add S3 Trigger so that when image Uploaded Image automatically resizes
* Now add package.json and Create Necessary package zip Obtained by running “Npm run package” command
* Upload Package in Code source of Lambda and Rename the
* We should also add Environment variable by linking DEST\_BUCKET as key and Second Bucket as Value
* Navigate to Test tab and add S3 Template and we should check event.json and edit file name of Bucket, arn value,key(image)
* Click on Test and check Whether the image is resized or not It shows health Check 200 means Lambda function is working