AWS Rekognition Image Matching Project

Steve Fiske

October 6, 2020

AWS Rekognition Project Usage Instructions

1. if not already setup, create AWS account
2. if not already setup, create C:\Users\USER\_NAME\.aws\credentials for Windows or ~/.aws/credentials for Linux with AWS account credentials:

aws\_access\_key\_id = <AWS access key id>

aws\_secret\_access\_key = <AWS secret access key>

1. if not already setup, create C:\Users\USER\_NAME\.aws\config for Windows or ~/.aws/config for Linux with AWS account config with desired region:

[default]

region=us-west-1

output=json

1. create AWS S3 bucket
2. unzip project onto hard drive
3. upload images from public/example\_imgs to AWS S3 bucket
4. update config.js with AWS region, AWS S3 bucket name and desired collection name
5. open MS-DOS command window
6. install Node.js which automatically installs Node Package Manager (npm) using instructions at <https://nodejs.org>
7. make sure nodejs is in path (e.g. - for Windows from MS-DOS command window: set PATH=%PATH%;C:\Program Files\nodejs;)
8. cd to project directory
9. npm install
10. node setup.js (only needs to be executed once ever unless config values are changed)
11. node index.ejs
12. open Chrome or Firefox web browser
13. clear cache
14. open <http://localhost:5555>
15. click ‘Match Images’ button
16. examine results noting that images with same prefix should match with Score: 100 - e.g. each S324\* image should match the other S324\* image with Score: 100; each S309\* image should match the other S309\* image with Score 100.

Implementation Process

Started researching by reading AWS Rekognition documentation on Internet. Created AWS account and S3 bucket on AWS website. Verified with current project System Admin that .aws credential and config files used for current project could be renamed and new credential and config files could be created without causing any problems - just need to rename original files back to original names to use for current project. Created S3 bucket and uploaded example images. Looked at Node.js AWS Rekognition examples. Downloaded and experimented with multiple Node.js AWS Rekognition examples. Created starting project from <https://github.com/aws-samples/aws-nodejs-sample>. Developed code to create collection of images from S3 bucket using Internet examples as a starting point. Reviewed available AWS Rekognition function documentation to determine that searchFacesByImage was the desired choice for image matching. Used AWS S3 listObjects to get list of images in applicable bucket/collection. Looped through images to call AWS Rekognition searchFacesByImage for each image. Formatted image match results into JSON.

Subsequently attempted to find solution with better performance. Used AWS Rekognition listFaces function to get list of FaceId values for each image in bucket/image and then called AWS SearchFaces for each image to be matched which became final solution. Developed minimum client front end needed to display and verify results of server image matching. Formatted image match JSON data returned from server into HTML table.

Estimated time spent is 32 hours (AWS account/S3 bucket/image upload: 2 hours; AWS Rekognition research: 6 hours; development: 14 hours; debug/test: 8 hours; documentation: 2 hours)

Issues

When using JavaScript to invoke either AWS Rekognition searchFaces or searchFacesByImage to get image matching results, the calls return immediately before AWS Rekognition image matching results are available. A large number of techniques were unsuccessfully attempted to wait for searchFaces or searchFacesByImage completion. That is why the workaround is made in index.ejs to call matchRekog.getImageCompList after delaying long enough for AWS Rekognition calls to complete. This allows AWS Rekognition searchFaces to complete and have the corresponding JSON result built and provided to build web page results. If code to call AWS Rekognition services would have been implemented with Java instead of Node.js and JavaScript, it may have been possible to use AmazonRekognitionAsync searchFacesAsync to complete image matching before returning to the caller eliminating the need for index.ejs to call matchRekog.getImageCompList.

Project currently works for Chrome and Firefox browsers but not Edge or IE 11 which have a problem loading jQuery. Investigation and debug attempts have not resolved this problem so far.

Resources Used

Some of the websites referenced to get information for implementation include:

<https://github.com/dwarcher/amazon-rekognition-example>

https://medium.com/@davidwallin/amazon-rekognition-api-example-23dc8d215a45

<https://www.youtube.com/watch?v=9Ka1fcn74Hg> (Rekognition JavaScript/Node.js video)

<https://docs.aws.amazon.com/rekognition/latest/dg/API_CompareFaces.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/faces-comparefaces.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/setup-awscli-sdk.html>

<https://aws.amazon.com/tools/>

<https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/getting-started-nodejs.html>

<https://aws.amazon.com/developers/getting-started/nodejs/>

<https://aws.amazon.com/sdk-for-node-js/>

<https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/index.html>

<https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/images.html>

<https://s3.console.aws.amazon.com/s3/buckets/sdf-rekog-example-images/?region=us-west-1&tab=overview>

<https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-region.html>

<https://docs.aws.amazon.com/general/latest/gr/rande.html>

<https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-credentials.html>

<https://stackoverflow.com/questions/31039948/configuring-region-in-node-js-aws-sdk>

<https://aws.amazon.com/visualstudiocode/>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_CreateCollection.html>

<https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/Rekognition.html#indexFaces-property>

<https://codingfundas.com/node-js-aws-sdk-how-to-list-all-the-keys-of-a-large-s3-bucket/index.html>

<https://www.nixgyd.com/read-data-from-aws-s3-in-node-js-aws-sdk/363>

<https://docs.aws.amazon.com/AmazonS3/latest/API/API_GetObject.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_IndexFaces.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_Image.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_S3Object.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/images-information.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/add-faces-to-collection-procedure.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_SearchFaces.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_SearchFacesByImage.html>

<https://docs.aws.amazon.com/rekognition/latest/dg/API_DetectFaces.html>

<https://aws.amazon.com/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc>

<https://us-west-1.console.aws.amazon.com/wellarchitected/home?region=us-west-1#/welcome>

<https://docs.aws.amazon.com/wellarchitected/latest/userguide/intro.html>

<https://aws.amazon.com/well-architected-tool/faqs/>

<https://d1.awsstatic.com/whitepapers/architecture/wellarchitected-Analytics-Lens.pdf?did=wp_card&trk=wp_card>

<https://d1.awsstatic.com/whitepapers/architecture/AWS-Serverless-Applications-Lens.pdf?did=wp_card&trk=wp_card>