# Cloud Platform Development (MHI326408/MHI325614)

Coursework Report

**Gregor MacPherson**

**S1509595**

Software Development for Business

*I declare that all work submitted for this coursework is the work of myself Gregor MacPherson alone unless stated otherwise.*

## Table of Contents

[Table of Contents](#_hdffy6li6pi7) 2

[Description of problem and approach to solve it 5](#_lh6vbq1tr8at) 3

[Description of how to implement the solution on another cloud platform 5](#_r7jzn8ul4tpi) 4

[Description of how to secure the application 10](#_56hygt91uysq) 5

[Application testing 10](#_5kmbuz7zi33i) 6

## Description of problem and approach to solve it 5

An application to perform image recognition was built using Amazon Web Services. AWS Rekognition, along with an AWS Lambda function, was used to obtain identification labels on images supplied by the user from a local python script.The implementation also makes use of the S3, SQS and Dynamo services to achieve a solution.

A Python Script was written to allow the user to upload and download images to Amazon S3. This was achieved by using the boto3 sdk for python. Boto3 contains a number of faciilies to interface with Amazon Web Services, including the ability to upload and download files to an S# bucket. A text based console was developed to allow the user to specify whether they want to upload or download to s3. They can then specify the file they wish to upload / download.

On upload to s3, the file is given a name in the format img###.ext. This was to aid in development and testing, as it is easier to identify when uploads have succeeded.



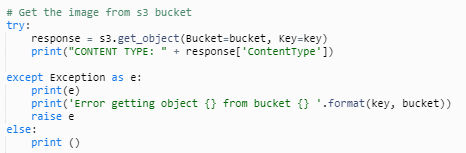
After being uploaded with the boto3 .upload\_file action, the image will appear in the s3 bucket image-greg-macpherson and trigger an SNS response. A topic was created that allowed SNS to publish a message on upload of an image

SNS will then send a message to the SQS queue, image-queue.

SQS will, upon receiving the message from s3/SNS, attempt to trigger a Lambda function. If this is unsuccessful then it will be retried 4 times before being sent to a Dead Letter Queue. This is again to aid in debugging as it makes it easy to see when the lambda trigger has failed.

The Lambda function gregs-image-processor will receive the message from SQS in a JSON format which will then be parsed by the function code, again written in python and implementing the boto3 sdk.

Using the image name taken fromt he JSON message, the lambda then accesses s3 to return the file as an object.



Using this, the function will make a call to rekognition which will identify the image and return labels.



Finally these labels will be added to a dynamo table using the put\_item command.

## Description of how to implement the solution on another cloud platform 5

Microsoft Azure could be used as an alternative platform for developing the application

Azure is a capable cloud platform solution that matches AWS in almost. Most AWS technologies have a comparable Azure counterpart.

Azure portal is the equivalent of the AWS Management console and provides a similar service

S3 for storage could be replaced with Azure Blob storage.

SQS would be Azure Queue Storage OR Azure Service Bus which both offer the ability to trigger Functions. Queue storage seems to be more aimed towards large workloads so for the scale of this application Service bus may be more appropriate. See([Compare Azure Storage queues and Service Bus queues](https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted))

The lambda function could be implemented using Azure Functions. The equivalent to DynamoDB would be Cosmo DB

## Description of how to secure the application

The application can be secure by management of permissions inside the IAM console.

An IAM service role is a role that can be used to delegate permissions to an AWS service.

It is best practice to use roles to set permissions rather than to individual users, it makes to easier to track the permissions available.In general, rule of least privileges should be followed for all permissions set.

S3 buckets are secured by default, but allowing access should be done with care so as not to expose data publicly. Instead of allowing access to all users, and all functions inside the bucket, the better solution would be to allow access to only upload and download functions and only from certain users. Redshift audit logging can be enabled for an s3 bucket to allow monitoring of suspicious activity.

Access keys can be used with S3 (and most other AWS services) through the use of KMS, a service for key management. This allows the data in s3 to be protected with encryption, further increasing its security.

SQS can be configured to only accept messages from the specific s3 bucket we wish to upload to. This can be specified by using the Amazon Resource Name (ARN) associated with the bucket. This is a unique identifier that can be used to ensure that only messages from the bucket are receieved.

The Lambda’s execution role allows it access to other AWS services. The specific ARN of the service is specified along with the operations it is allowed to perform.

## Application testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Test** | **Expected result** | **Result** |  |
| Python Script | Correct file name | Success message | As expected |  |
|  | Incorrect file name | Fail with message | As expected |  |
| s3 | Upload Image to s3 from python script | File will appear in S3 | As expected |  |
|  | Download images from s3 with python script | Successful download | As expected |  |
|  | Upload with incorrect permissions | Fail | As expected |  |
|  | Message from SNS on image upload | Message received should be contain corrrect information | As expected |  |
| SQS | Image upload | should trigger message in queue & trigger lambda function | As expected |  |
|  | Upload with Incorrect SQS permissions | Failure to receive messages | As expected | No message appears in queue |
|  | Incorrect message format | Sent to dead letter queue | As expected |  |
| Lambda | Image upload | Lambda function triggered | As expected |  |
|  | Image upload | Labels added to DynamoDB table | As expected |  |
|  |  |  |  |  |
|  |  |  |  |  |