Daniel Russell

S1707149

Cloud Platform Development

“I declare that all work submitted for this coursework is the work of Daniel Russell alone unless stated otherwise.”

Cloud Platform Development CW1

Daniel Russell

Contents

[Problem and Solution 2](#_Toc67874887)

[Description of problem and approach. 2](#_Toc67874888)

[Description of how the application can be cost optimised. 2](#_Toc67874889)

[Non-Code Implementation 2](#_Toc67874890)

[Setup of Lambda Role and Permissions 2](#_Toc67874891)

[Lambda Creation 5](#_Toc67874892)

[SQS Trigger 6](#_Toc67874893)

[Security features and application testing 6](#_Toc67874894)

[Description of security features. 6](#_Toc67874895)

[Application Testing. 7](#_Toc67874896)

# Problem and Solution

## Description of problem and approach.

The problem that must be solved from this coursework is to develop a solution that utilizes Amazon’s Web Services (AWS) to upload and store five audio files into an S3 Bucket. The Audio files must then be transcribed via AWS Transcribe to create a transcription file from the audio. The transcription file must then be sent to AWS Comprehend to detect the sentiment from the said text present in the transcription file. The audio file name and concluded sentiment must then be stored in DynamoDB and any Negative sentiment result should receive an SMS text message.

The approach chosen to approach this problem is to use Python to create a main script to set-up the AWS environment before uploading the audio files with an accompanied SQS message for each file.

Upon receiving the SQS Message a developed Lambda function (also written in Python) should trigger for that corresponding audio file to process it through Transcribe and Comprehend before storing the extracted details into DynamoDB and send an SMS text message via SNS if the sentiment is negative.

## Description of how the application can be cost optimised.

There are a few ways the application Can be cost optimised.

One way is to minimise the number of requests sent to Amazon Web Services when using resources such as avoiding unnecessary repetitions of CRUD operations when they could be managed in one request or store the required data local in a variable or a file to be used in a more efficient manner.

Another way is to create a budget which would monitor the incurred costs or total usage from utilizing Amazon services and allow for the implementation of restrictions on the future use of services to maintain a set cost limit or a usage limit and send a notification when the said limit is close to being exceeded.

# Non-Code Implementation

## Setup of Lambda Role and Permissions

|  |
| --- |
| Role Creation |
|  |
| Before any implementation is taken place, a role is defined for the Lambda function by the name of ‘LambdaRole-S1707149’. |

|  |
| --- |
| New Policy created via JSON |
|  |
| Using the JSON file created to give permissions to the Lambda, the data is pasted into the JSON textbox and attached to the newly created policy ‘Role-S1707149’. The Policy is then attached to ‘LambdaRole-S1707149’. |

|  |
| --- |
| LambdaRole Preview |
|  |
| In addition to the created Policy Role-S1707149. AwsLambdaBasicExecutionRole is added for minimal CloudWatch logs create and put permissions (As exemplified below). |
|  |

## Lambda Creation

|  |
| --- |
| Lambda Creation |
|  |
| Preview of Lambda |
|  |
| A lambda function called Lmabda-S1707149 is created using the previously mentions LambdaRole containing all the necessary permission required for the function to operate. |

## SQS Trigger

|  |
| --- |
| Create Trigger from SQS |
|  |
| During debugging the que ‘que-s1707149’ is created to configure the SQS que to trigger Lambda-S1707149 once a message is received. The que was then deleted for testing. |

|  |
| --- |
| Preview of Trigger from SQS |
|  |
| SQS ‘que-s1707149’ show Lambda-S1707149 as a Lambda trigger. |

# Security features and application testing

## Description of security features.

The application can be secured in many ways on AWS.

AWS services and access can be restricted according to IAM users or IAM roles using custom made or pre-defined policies granting permissions to any given service or user to operate on the cloud.

AWS also provides several services that can be enabled to monitor activity and applications for suspected malicious actions such as Amazon Guard Duty which uses machines learning to detect malicious activity or Amazon Inspector which is used to identify vulnerable areas on the cloud.

## Application Testing.

|  |
| --- |
| Main Is executed |
|  |
| The main python file is executed and begins by setting up the cloud environment. S3 Bucket is created if it does not already exist and same for SQS and the corresponding stack responsible the creation of DynamoDB.  After creating the necessary resources, the audio files are uploaded with a 15sec pause sending an SQS message post each upload before iterating to the next audio file. |

|  |
| --- |
| S3 Bucket evidence |
|  |
| The S3 bucket is created, and the audio files are uploaded one after the other with a 15sec gap and a 30sec gap for every SQS message. |

|  |
| --- |
| Transcription jobs evidence |
|  |
| When the lambda function is trigger after 30seconds of the SQS receiving the message Transcription jobs are created if they do not already exist. |

|  |
| --- |
| Evidence of DynamoDB Add |
|  |
| At the final stages of execution, the Lambda function extracts the transcription file url from the completed transcriptions to feed into Comprehend for sentiment analysis. The resulting response is them filtered for the sentiment scoring and is stored with the file name in table-S1707149.  A final SMS function is called for Audio1 and Audio5 for being negative. |

|  |
| --- |
| Evidence of SMS |
|  |
| Seeing as comprehend detects Audio1 and Audio2 as negative. SNS.publish is triggered from the Lambda function and sends a text notification to the receiving phone number set in the Lambda function. |