Polly Transcribe Cloud Based Application Using Amazon Web Services
CIS4010 Cloud Computing Team Project Report

Miriam Snow, Darren Chan, Alex Ciuba
School of Computer Science, University of Guelph, Guelph, Ontario, Canada
{*msnow01, dchan04, aciuba*}*@uoguelph.ca*

Keywords: cloud, computing, amazon, polly, transcribe, text, speech

Abstract: In this report we investigate the challenges and benefits of developing in the cloud. We primarily focus our efforts on one of the major cloud providers, Amazon Web Services (AWS). To gather firsthand experience, we come up with a project that relies heavily on AWS for its core functionality. We reflect on the development process of this sample project to provide insight to those interested in pursuing similar projects. As part of the report we include instructional materials for some key APIs utilized in our work, Amazon Polly, Amazon Transcribe, Amazon S3.

1 INTRODUCTION

Our task was to choose a project that explores the benefits and challenges of developing applications in the cloud. To get some idea of what project we wanted to pursue, we began by looking through some of the services offered by Amazon Web Services (AWS) and Azure. During our initial research into offerings of both providers, one service that peaked our interest was Amazon Polly. It is a text-to-speech solution that is capable of generating speech in a variety of languages and different accents. Going off of the same theme, we noticed that Amazon also offered a transcription service.

1.1 Project Statement

Build a transcription and text-to-speech application that is user friendly and designed to assist university students with accessibility concerns who require content in alternate formats.

2 OBJECTIVES

Our original goal was to develop an android application that allows users to upload their notes as text or audio, and have them converted to an alternative format (text to speech, or speech to text). The user base that we wanted to cater to was university students who may have accessibility requirements that prevent them from effectively taking notes in a certain medium. We also envisioned that the application would appeal to those who find that they benefit from studying their notes in multiple forms. Aside from the main deliverable, we wanted to use this project as an opportunity to get familiar with some of the many APIs offered by AWS and Azure.

3 CLOUD ASPECTS

Amazon Polly is a cloud based machine learning service that turns text into lifelike speech. This service allows you to create applications that talk and build new categories of speech-enabled products. This service offers dozens of lifelike voices across many languages, giving you the tools to build speech-enabled applications that work in many different countries. Amazon Polly offers many benefits such as natural sounding voices, store and redistribute speech, real-time streaming, and customize and control speech output.

Amazon Transcribe is a cloud based machine learning service that offers speech-to-text capability to your applications. This service allows you to analyze audio files stored in S3, where it will perform speech-to-text conversion and return a text file of the audio file. Common uses of AWS Transcribe include transcription of customer service calls and generating subtitles for video content. This service is continuously keeping pace with the evolution of language,

Figure 1: Retrieve file contents.

```
textstring = ""
for line in textFileObj:
    textstring = textstring + " " + line
```

Figure 2: Initial setup.

```
session = Session(profile_name="MSnow")
polly = session.client("polly")
```

simplifying the development of your application.

Amazon S3 is amazon's cloud storage that is used in conjunction with Polly and Transcribe. S3 allows you to store and retrieve the files used in Polly and Transcribe at a fast and inexpensive way.

3.1 Tutorial Topic 1: Amazon Polly

In this tutorial, you will be introduced to Python SDK for Amazon Polly. By the end, you should have enough knowledge to be able to complete text-to-speech functionality and create an mp3 file.

The prerequisites for this tutorial are:

- Install and set up AWS CLI
- Store a text file (.txt) in your local directory (downloaded from an S3 bucket)
- Import boto3

Complete the following steps:

1. In order to simplify the next steps, get the file from your local directory and create a single string of text. (figure 1)
2. Define the session profile name and polly client variables to use in the next step of text-to-speech conversion. (figure 2)
3. Call the polly method using the string of text that you created in step 1 as the first parameter. Here, you can also specify the output format and VoiceId which represents the accent that is chosen. A list of Voice IDs can be found in Amazon Polly's documentation. If the response creates an error, the program will exit. (figure 3)
4. The next step is to stream the audio and use the response from step 3 to create an mp3 audio file. You can specify the name of the mp3 file by changing the output variable. In this case, a file named speech.mp3 will be downloaded to your local directory which will reflect the contents of the text file used in step 1. In addition, you can upload this file from your local directory to any S3 bucket. (figure 4)

Figure 3: Retrieve a response.

```
try:
    response = polly.synthesize_speech(Text=textstring, OutputFormat="mp3", VoiceId="Joanna")
except (BotoCoreError, ClientError) as error:
    print(error)
    sys.exit(-1)
```

Figure 4: Convert the response to an audio file.

```
if "AudioStream" in response:
    with closing(response["AudioStream"]) as stream:
        output = "speech.mp3"
        try:
            with open(output, "wb") as file:
                file.write(stream.read())
        except IOError as error:
            print(error)
            sys.exit(-1)
else:
    print("Could not stream audio")
    sys.exit(-1)
```

Figure 5: Initial setup.

```
region = 'ca-central-1'
s3 = boto3.client('s3', region_name=region)
transcribe = boto3.client('transcribe')

job_name = "<Name of the transcription job"
job_uri = "<URI of the audio file>"
bucketName = "<S3 bucket name>"
```

3.2 Tutorial Topic 2: Amazon Transcribe

In this tutorial, you will be introduced to Python SDK for Amazon Transcribe. By the end, you should have enough knowledge to be able to start and retrieve transcription jobs as well as extract the transcribed text from the output json file.

The prerequisites for this tutorial are:

- Install and set up AWS CLI
- Store an audio file (.WAV or .MP4) in your S3 bucket with proper permissions

Figure 6: Start a transcription job.

```
transcribe.start_transcription_job(
    TranscriptionJobName=job_name,
    Media={'MediaFileUri': job_uri},
    MediaFormat='mp3',
    OutputBucketName=bucketName,
    LanguageCode='en-US'
)
```

Figure 7: Retrieve all completed transcription jobs.

```
response = transcribe.list_transcription_jobs(
    Status='COMPLETED'
)
```

Figure 8: Extract the transcribed text

```
for jobs in response['TranscriptionJobSummaries']:
    jobName = jobs['TranscriptionJobName']
    jobName.encode('utf-8');
    print('Transcription job name: ' + jobName)
    transcriptionFile = jobName + ".json"
    result = s3.get_object(Bucket=bucketName, Key=transcriptionFile)
    data = json.load(result['Body'])
    transcriptText = data['results']['transcripts']
    print("Text: " + transcriptText[0]['transcript'])
```

- Import boto3

Complete the following steps:

1. In order to simplify the next steps, initialize three variables (job_name, job_uri, and bucketName). Job_name will be the name of the transcription job that we will use to retrieve it. (figure 5)
2. To start a transcription job, we will call the "start_transcription_job" function from boto3 client and fill in the required parameters. In this example, we use an mp3 file. Simply change MediaFormat to 'wav' if you use a .wav file. This will start the transcription job using the audio file in the s3 bucket indicated in "bucketName". (figure 6)
3. To retrieve a list of all transcription jobs that have been completed, simply call the "list_transcription_jobs" from boto3.client with the Status set to 'COMPLETED'. (figure 7)
4. This part of the tutorial will be quite bothersome because the output produced by the transcription service buries the transcripted text under many layers. To begin, use the 'response' variable in step 3 and create a loop using response['TranscriptionJobSummaries']. Next you need to extract the job name located in 'TranscriptionJobName'. Afterwards, we will concatenate '.json.' onto the transcription job name in order to locate the json file (the output produced by AWS Transcribe) located in your S3 bucket. Simply call the 'get_object' function to retrieve the Json file and load the data using 'json.load' into a variable. The transcripted text will be located in 'results' → 'transcripts' → '0' → 'transcript'. (figure 8)

3.3 Tutorial Topic 3: Amazon S3

In this tutorial, you will be introduced to Python SDK for Amazon S3. By the end, you should have enough knowledge to be able to upload, download and delete files from S3 buckets.

Figure 9: Download a file from a bucket.

```
s3 = boto3.client('s3')
s3.download_file(bucketName, fileName, fileName)
```

Figure 10: Upload a file to a bucket.

```
s3 = boto3.client('s3')
with open(myFile, "rb") as f:
    s3.upload_fileobj(f, bucketName, myFile)
```

The prerequisites for this tutorial are:

- Install and set up AWS CLI
- Have an existing S3 bucket containing files with proper permissions
- Import boto3

Complete the following steps:

1. To download a file from a bucket to your local directory, simply define a boto3 client and then call the download_file method using the bucket name and filename as parameters. The first filename parameter is the name of the file in the bucket that is to be downloaded. The second filename parameter is the name of the file once it is downloaded to your local directory. (figure 9)
2. To upload a file from your local directory to a bucket, simply define a boto3 client and then open the file in read-binary mode. The myFile parameter indicates the name of the file in your local directory. Then call the upload_fileobj method indicating the local file, bucket name and name of the file once it is uploaded to your bucket. (figure 10)
3. To delete a file from your S3 bucket, simply define a boto3 resource and then define an object using the bucket name and name of the file to be deleted. Then, call the delete method on the object and your file will be deleted from the bucket. (figure 11)

4 MAJOR ACCOMPLISHMENTS AND CONTRIBUTIONS

The major accomplishment from our work is completing what was expected based on our project

Figure 11: Delete a file from a bucket.

```
s3 = boto3.resource('s3')
s3.Object(bucketName, theFile).delete()
```

statement. We delivered a fully functional speech-to-text and text-to-speech application which showcased Amazon's EC2, S3, Polly, and Transcribe in action. The main purpose for this project was to learn about different cloud computing resources and to help students who want or need new methods of studying.

While developing this project, we discovered some usability concerns that we did not think about with the original project proposal. We realized that in addition to storing the files in the cloud it would be helpful to have other features like uploading, downloading, and deleting files directly through the UI. This helped to improve usability for the user so they do not need to access Amazon's S3 portal whatsoever. Storing all files in the cloud allows users to refer back to past transcription and text-to-speech files at any point, and all of their files will be accessible to them, even if the app is stopped and restarted.

In addition, while this is an English language application, various accents and genders were available via Amazon Polly, so we thought it would be beneficial to offer some of these same options to the user as well when creating an audio file. Certain accents may be easier or more difficult to comprehend for different students. Therefore, we also added the "change accent" feature in the user interface, primarily to assist with accessibility.

Next, during the course of this project, not only were we able to deepen our understanding of Amazon's cloud services, specifically the Python SDK for Amazon Polly, Amazon Transcribe, and the AWS S3 storage management platform, but we learned more about natural language processing too. NLP is a hugely applicable field, and developing a deeper understanding of text-to-speech and speech-to-text was particularly educational.

Lastly, accessibility is something that is important to us, since it is a huge part of university life. All students process and create information at different levels, so we were excited to learn more about accessibility and contribute to helping a cause that affects everyone personally.

5 FUTURE WORK

If we decide to continue working on this project, the user interface would be our next focus. Currently, our application has a very simple UI displayed in the terminal. This was due to our decision made at the start to put a majority of our time towards developing the main functionalities. After developing a much needed user-friendly UI, the next step towards improvement is to include more language choices. The current state of our application only works for english which definitely puts a restraint on who will be able to use our application. We would like to remove this restraint by implementing a way for user's to choose their preferred language. Lastly, we want to include more file format options for user's to choose from. The only file-formats our application can work with are .mp3 and .txt files. We want to work towards increasing the compatibility with other file formats such as .pdf and .wav.

6 TEAM MEMBER CONTRIBUTIONS

6.1 Miriam Snow

- Code to implement AWS Polly, UI, changing accents, delete/upload/download files
- Screen record and present demo video
- Wrote presentation slides
- Screen record and edit together group video
- Upload group video to YouTube
- Major Accomplishments and Contributions (report)
- AWS Polly Tutorial (report)
- S3 Bucket Tutorial (report)
- References (report)

6.2 Darren Chan

- Code to implement AWS Transcribe
- Present slides 6-9
- AWS Transcribe Tutorial (report)
- Cloud Aspects (report)
- Future Work (report)

6.3 Alex Ciuba

- Present slides 2-5
- Introduction (report)
- Objectives (report)
- Abstract (report)
- Format report to latex

REFERENCES

Amazon Web Services, Inc. (n.d.). *What Is Amazon Transcribe?* Amazon Transcribe Developer Guide https://docs.aws.amazon.com/transcribe/latest/dg/what-is-transcribe.html

Amazon Web Services, Inc. (n.d.). *What Is Amazon Polly?* Amazon Polly Developer Guide https://docs.aws.amazon.com/polly/latest/dg/what-is.html

Amazon Web Services, Inc. (n.d.). *Getting Started with Amazon Simple Storage Service.* Amazon Simple Storage Service https://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html