THE IMPLEMENTATION OF A CLOUD/WEB-BASED SOLUTION

AISHA BI

2021

Table of Contents

Introduction

This section introduces the aim of the project and the objectives used to achieve this.

The aim of this project is to design and develop an application to process real-time data and analytics of a data stream. To achieve this aim, several objectives have been created as seen below.

Objective 1: Retrieve real-time stream data from Twitter. This data must be streamed into the application.

Objective 2: Display a dynamic list of 10,30,50 top trending Hashtags in either the last 10,30 or 60 minutes.

Objective 3: Display a dynamic list of 10,50,100 top trending Tweets in either the last 10,30 or 60 minutes.

Objective 4: Use a word cloud model to represent the hashtag or tweet data.

Objective 5: Demonstrate additional insights on the data insight

Objective 6: Create a command line interface showing options for the data type, time and amount to demonstrate the features of this application.

The importance of this work would be to increase insights into twitter's data. This application is cloud-based and thus results in reduced infrastructure costs.

Background

To implement the objectives for this project, many tools and concepts were used. This section describes the concepts and tools used such as Cloud Computing, Amazon Web Services, Twitter Application Programming Interface (API) and the programming language, Python .

One of the concepts used was Cloud Computing.

Cloud computing:
Cloud computing describes the process of having readily available computer system resources such as storage, servers and software over the internet. Cloud computing offers three main services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a

Service (SaaS). Rountree, D. et al. (2014) explains that IaaS provides users with computing resources such as data storage, processing power or a data centre. PaaS is responsible for providing users with storage infrastructure and development solutions such as web servers, software development tools and database systems. This type of cloud computing allows the user to develop and manage the required application for the business without the need to build and maintain the infrastructure. SaaS provides the user with application-level licensed software resources which provides office software, database or messaging software. Baun, C. et al. (2011) explains that the main benefit for using cloud applications for the user it to eliminate the host and physical and maintenance cost of the computing devices. The cloud services offers self-service through a web interface in which the user can control the usage level of the products needed for the business. The user also only pays for the service or application they use, reducing costs substantially.

The cloud computing service used to implement the objectives for this project was Amazon web services. AWS provides users with on-demand cloud applications on a pay-as-you go payment model. This service is implemented using server farms which are located over many countries.

<u>Twitter API -</u>Twitter is a social networking service in which users post messages limited to 280 characters called ‘tweets’. According to Twitter’s Q3 2020 report, 187 million people use the platform daily. This statistic shows the scope of the data twitter holds. To access the tweets and hashtags from this web service, a twitter API was used. An API can be classified as a software intermediary between two application to facilitate communicate. API’s are used to interact with a software without the need for the developer to make the entire code publicly visible. The purpose of an API is to increase efficiency, by increasing accessibility, the data can be accessed and used for different channels thus allowing for the software to be shared easily and integrated with other applications and technologies.

<u>Python was used to create the command line interface.</u>
Python was chosen as the programming language for this project.For this project, the Python programming language is used. Anders, M (2011) argues that the object-oriented programming language, Python is a favourable server-side scripting language. Benefits of this language include easy readability due to treating whitespace as meaningful. This is especially useful for beginners using this language as it makes the language easier to learn and use, as is the case for this project. Another advantage to using Python is that the language is distributed with ‘well-maintained libraries called modules’ (Anders, M , 2011) that provide access to .csv files and parsing XML. Pythons’ standard libraries enable the user to execute complex functionalities easily in applications.

Detailed Background

This section delves deeper into the tools used for this project. Tweepy was used to collect the tweets. The AWS services used for this project include, Lambda for tweet analysis and S3 Buckets for tweet storage.

AWS LAMBDA

AWS Lambda is a good example of the abstract serverless computing model. AWS Lambda allows users to run functions in the cloud using AWS's serverless technology. (Baun, C. et al, 2011) This service manages the computing resources required to run the user's code, thus reducing administrative costs and increasing efficiency. The code is executed in response to events in AWS services. These events could range from a HTTP request from the Amazon API Gateway or file changes in S3 buckets. Lambda manages the administrative duties such as managing the capacity and logging the functions.(Amazon, 2021)

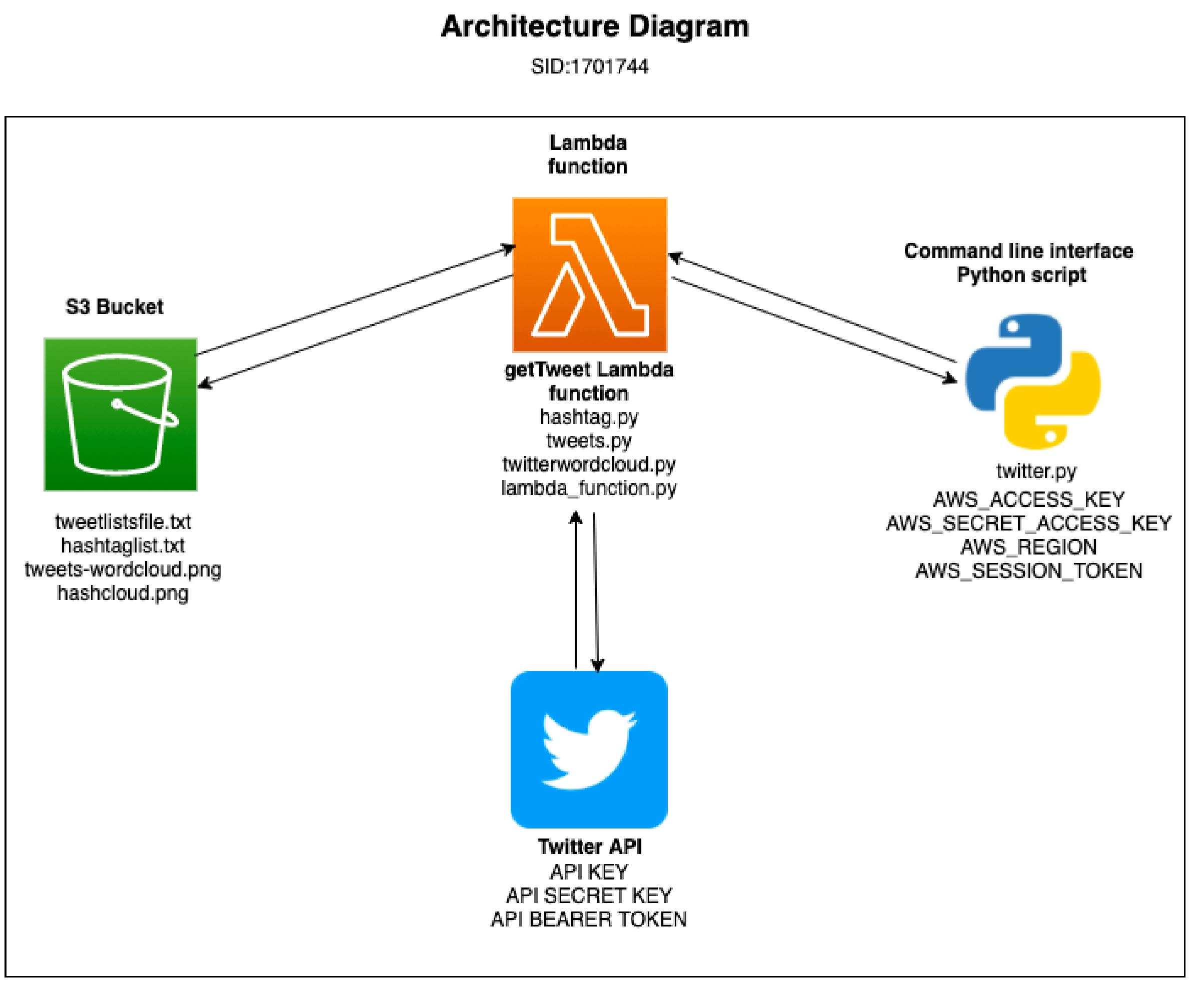
S3

AWS Lambda can be used alongside Amazon Simple Storage Service(S3). Amazon S3 provides a storage infrastructure which is capable of holding large sets of data. This uses an object storage architecture which focuses on three main properties: scalability, availability and low latency. The objects can be identified by a unique user-assigned key. The buckets are managed through a web console provided by Amazon. Amazon S3 provides an interface in which the user can store and retrieve the required data. This will be stored on the web and is accessible at any time. (Erdoğan, N, 2019) AWS S3 has buckets which are containers for storing the data. There is no limit to the number of objects that can be uploaded to the bucket. Each object can hold up to 5TB. Permissions can be assigned to the buckets to determine who can access the data. The Amazon S3 also provides logs for the user.

Tweepy

Twitter API is another core component for this cloud application. To use the Twitter API, a develop account must be created. Once the developer account has been approved by Twitter, credentials will be provided to authenticate the requests to the API. There are five keys provided: API Key which is a username, API Key Secret which acts as a password, Access Token which represents the twitter account, Access Token Secret which represents the Twitter account that owns the application and the Bearer Token which represents the application. (Twitter, 2021) For this project, Tweepy is used to access the Twitter API with Python. Tweepy is an open source Python package which includes classes and methods that represent the Twitter API endpoints. Twitter API uses OAuth authentication which require the authentication keys to connect tot he API instance. To use the Tweepy package, the package must be imported, the authentication credentials must be set and the API must be instantiated. The next step would be to create the API object. (Tweepy, 2021)

Architecture



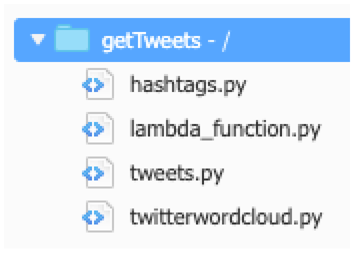
Architecture Diagram Description:

The Twitter developer account provides the authorisation credentials top access the Twitter API. The Lambda function has a layer with the tweepy installation which connects the Lambda function to the Twitter API to access the twitter data. The Lambda function streams the tweets from the Twitter API and analyses the tweets using the configurations and stores these sorted data into an S3 bucket. The Python command line interface triggers the lambda function and receives tweets, hashtag and wordcloud information from the S3 bucket using the Boto3 client. The AWS credentials must be used to connect the local python script to the Lambda function.

Solutions

Data Processing - Lambda function

A Lambda function was created to process the twitter data. The Twitter API credentials were stored in the configuration as environment variables. Tweepy and Word Cloud python packages were downloaded and uploaded to Lambda through Layers. The Lambda file structure split the different functionalities of the getTweets function into separate python files as seen below.



getTweets - /
hashtags.py
lambda_function.py
tweets.py
twitterwordcloud.py

Lambda_function.py

The Lambda function python file, imported tweepy which allowed the function to communicate with the Twitter API to collect the tweets. This file also specified the api authorisation keys.

```
auth = tweepy.OAuthHandler(api_key, api_secret)
    auth.set_access_token(api_access_token, api_secret_access_token)
 api = tweepy.API(auth)
```

The getTweets and getTopics function was also imported from tweets.py and hashtags.py files.

```
if functiontype == "tweet":
    tweets = getTweets(tweet_number, api,time)
elif functiontype == "hashtag":
    trends = getTopics(tweet_number, api)
```

Tweets.py

The tweets.py file imported datetime, tweepy, boto3,json and twitterwordcloud to help create the functionality. The four methods created were: getTweets,sortTweets,tweetcloud and save_to_bucket.

The getTweets method creates two empty arrays, one for the tweets and another to store how many retweets the tweet has. The for loop goes through the tweets and uses the Twitter api to query for tweets with the topic of lockdown, with the amount specified by the user.

```
for _ in range(tweettime*2):
    tweetsearch = api.search(q="lockdown", count=amount,lang="en")
```

The next for loop iterates through the tweets in the query and adds them to the array. There is a sleep time of 30 seconds. This method returns the the list of tweets.

```
for tweet in tweetsearch:
        tweet_list.append(tweet.text)
        retweet_list.append(tweet.retweet_count)
```

The sort_tweet method as shown below, creates a list for the tweets and the retweets and zips these two values together to help organise the tweets. The purpose of this method is to return a list of tweets according to the retweet volume.

```
retweet_list,tweet_list = (list(t) for t in zip(*sorted(zip(retweet_list, tweet_list), reverse=True)))

 #trim the list to get the correct amount of tweets
    retweet_list = retweet_list[0:amount]
    tweet_list = tweet_list[0:amount]

#zip the two arrays back up and put them in order using the dictionary
    tweets_by_date = dict(zip(retweet_list,tweet_list))
```

The tweetcloud method as shown below, creates an empty string, tweets are added onto this string using a for loop.

```
for tweet in list_tweets:
      tweetstring += " "+str(tweet)
```

This string is then used to create the wordcloud and outputs this to a .png file.

```
plot_cloud(tweetstring, 'tweets-wordcloud.png')
```

This .png file of the wordcloud is saved to an S3 bucket on AWS using the boto3 client. The save_to_bucket method lists the bucket configuration. This prints the tweets to the tweetlistsfile text file as uploads to the specified bucket.

```
s3_bucket.upload_file('/tmp/tweetlistsfile.txt', BUCKET_NAME,'tweetlistsfile.txt')
```

Hashtags.py

For the hashtags processing, the hashtag data was collected and narrowed down to England use a WOI identifier code.

```
def getTopics(amount, api):
        data = api.trends_place(23424975)
```

This data was placed into an array using a for loop. The data was separated into two groups, the name of the trend and the tweet volume to determine the most popular trends.

```
for trend in data[0]['trends']:
        trends.append((trend['name'], trend['tweet_volume']))
trends = trends[0:amount]
```

These tweets were then stored in the cloud using the save_to_bucket method. The save_to_bucket function, collects the trends from the getTopics method and configured the bucket credentials such as the bucket name. The boto3 client is initialised to connect to this S3 bucket.

```
BUCKET_NAME = "tweetsbucket0301"
s3_bucket = boto3.client('s3')
```

The botoclietn opens the hashtag text file and prints the trends to this file. This text file is uploaded to the S3 bucket.

```
s3_bucket.upload_file('/tmp/hashtaglist.txt', BUCKET_NAME, 'hashtaglist.txt')
```

Twitterwordcloud.py
First the bucket configurations are set and the boto3 client is initaised to the S3 bucket.

```
def plot_cloud(wordstring,filename):
    BUCKET_NAME = "tweetsbucket0301"
    s3_bucket = boto3.client('s3')
```

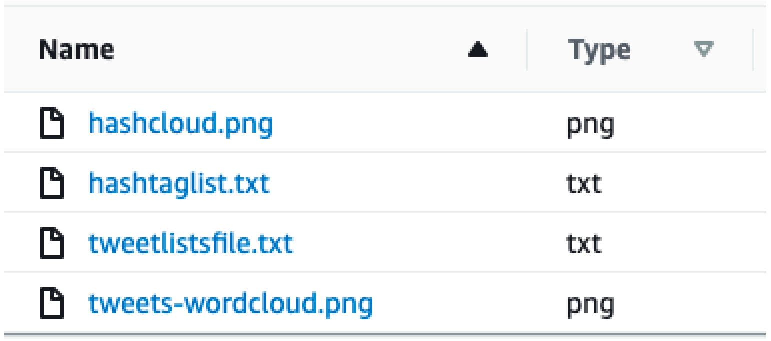
Next, the word cloud is generated and configured with the formatting.The tweets or trends are imported from the relevant functions and stored as a string.

```
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='salmon',
    colormap='Pastel1', collocations=False, stopwords = STOPWORDS).generate(wordstring)
```

The wordcloud is saves to a tmp file on lambda. This tmp file is uploaded to the S3 bucket.

```
wordcloud.to_file('/tmp/'+ filename )
s3_bucket.upload_file('/tmp/'+filename, BUCKET_NAME, filename)
```

Data Storage - S3 Bucket
For the data storage feature of the application a S3 Bucket was created on AWS. Public access was granted to allow for the user to view the tweets, hashtags and wordclouds. A bucket called tweetsbucket0301 was created. The bucket consists of four files generated by the lambda function which was triggered by the command line interface python script which determined the amount, type and time of the information required. Everytime the python script runs, the lambda function is triggered and the new data is generate and the files in the S3 buckets are updated.



Name ▲	Type ▽
hashcloud.png	png
hashtaglist.txt	txt
tweetlistsfile.txt	txt
tweets-wordcloud.png	png

User Interface - Python Script
To implement the Command line user interface, a python script was created.
The python script would have to trigger the lambda function and pass on the payload which includes the type of data (tweets,hashtags), amount and time. The payload variables are collected from the user input.

```
#create payload - dictionary
        lambdafunction = {
        "type": lambda_type,
        "no": number,
        "time": time
        }
```

The trigger function, contains the aws credentials. The boto3 client is created to connect to the lambda function. The AWS credentials are passed through the boto3 client to connect to the lambda function.

```
#create client to connect to lambda function
        client = boto3.client('lambda',region_name=AWS_REGION,
                        aws_access_key_id= AWS_ACCESS_KEY,
                        aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
                        aws_session_token= AWS_SESSION_TOKEN)
```

The boto3 client is invoked and the invocation type is set to Request response which collects the response from the lambda function. The payload variables are also included.

```
        result =    client.invoke(FunctionName=
        "arn:aws:lambda:us-east-1:199124270429:function:getTweets",
        InvocationType='RequestResponse',
        Payload=json.dumps(lambdafunction))
```

Multi-threading was used to start the lambda process whilst triggering the lambda function at the same time.

```
p = Process(target=trigger,args=(lambdafunction,))
        p.start()
```

If the lambda type selected by the user was tweets, a bucket object would be initalised.

```
bucket_object = s3.Object(BUCKET_NAME,FILE_NAME)
```

This object would be read and converted from a string to a dictionary in python using the ast.literal_eval()

```
bucket_content = bucket_object.get()['Body'].read().decode('utf-8')
                dic_content = ast.literal_eval(bucket_content)
```

A for loop is used to print all the tweets and the amount of retweets values together to the user.

```
for retweet, tweet in dic_content.items():
                        print("Retweets: " + str(retweet))
                        print("Tweet: " + tweet)
                        print(" ")
```

A for loop was used to display the hashtags.

```
for trends in dic_content:
                        print("Trend: " + str(trends))
                        print(" ")
```

Options for the command line user interface was printed to the user.

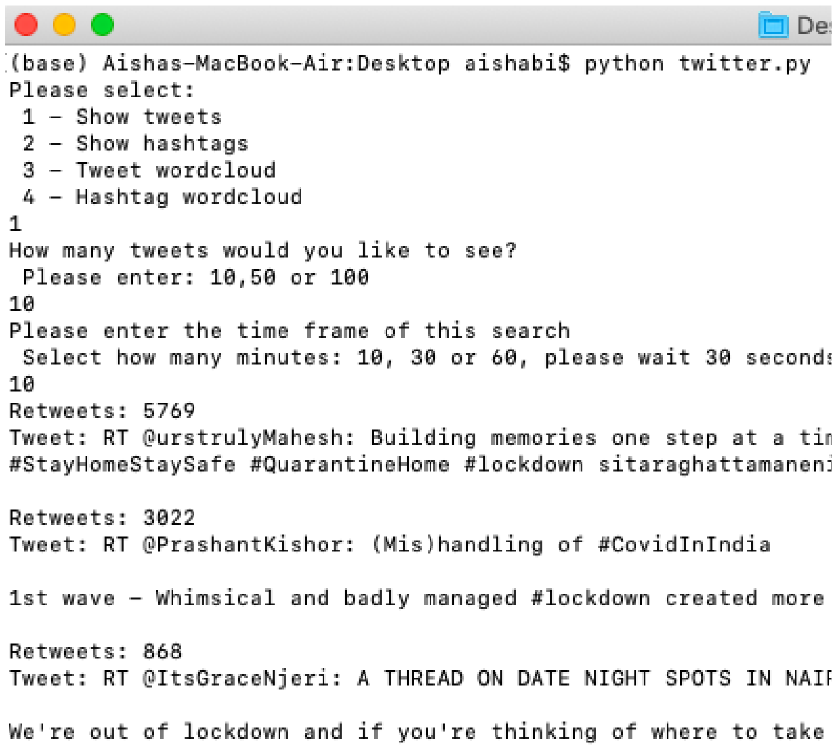
print('Please select: \n 1 - Show tweets   \n 2 - Show hashtags \n 3 - Tweet wordcloud \n 4 - Hashtag wordcloud')

option = input()

Depending on the user input a message will display to the user asking for the amount and the time frame.

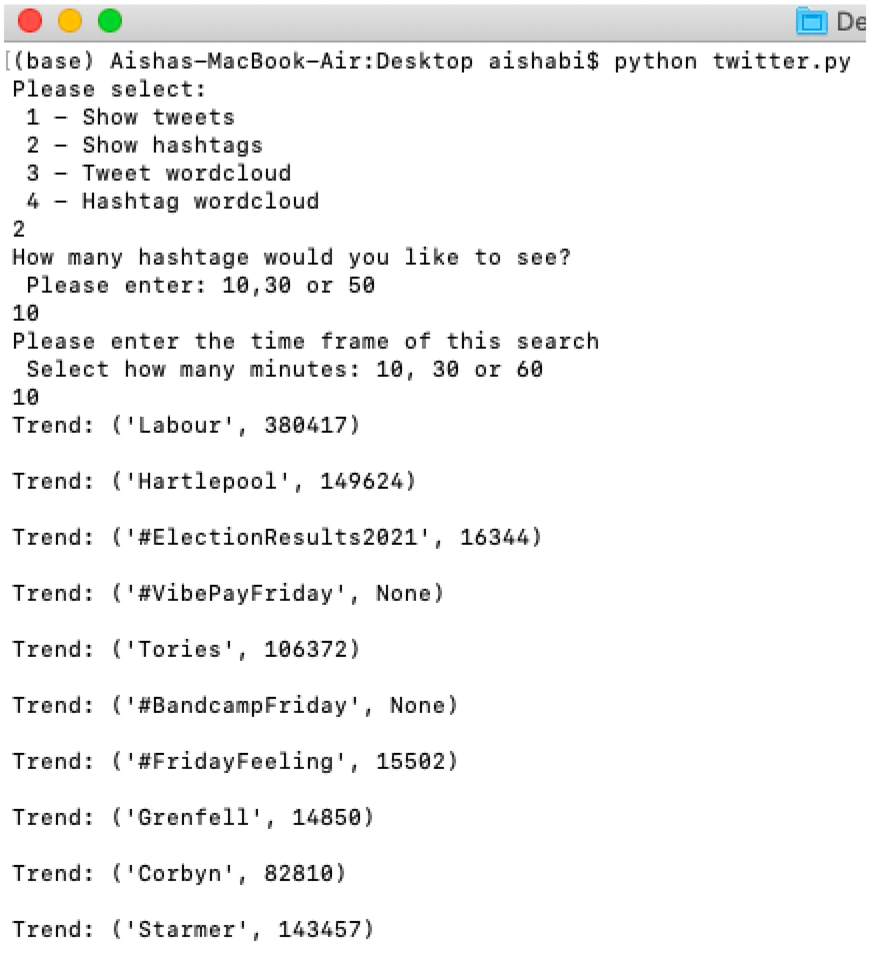
Application demonstration:

Tweets:



```
(base) Aishas-MacBook-Air:Desktop aishabi$ python twitter.py
Please select:
 1 - Show tweets
 2 - Show hashtags
 3 - Tweet wordcloud
 4 - Hashtag wordcloud
1
How many tweets would you like to see?
 Please enter: 10,50 or 100
10
Please enter the time frame of this search
 Select how many minutes: 10, 30 or 60, please wait 30 second
10
Retweets: 5769
Tweet: RT @urstrulyMahesh: Building memories one step at a ti
#StayHomeStaySafe #QuarantineHome #lockdown sitaraghattamanen

Retweets: 3022
Tweet: RT @PrashantKishor: (Mis)handling of #CovidInIndia

1st wave - Whimsical and badly managed #lockdown created more

Retweets: 868
Tweet: RT @ItsGraceNjeri: A THREAD ON DATE NIGHT SPOTS IN NAI

We're out of lockdown and if you're thinking of where to take
```

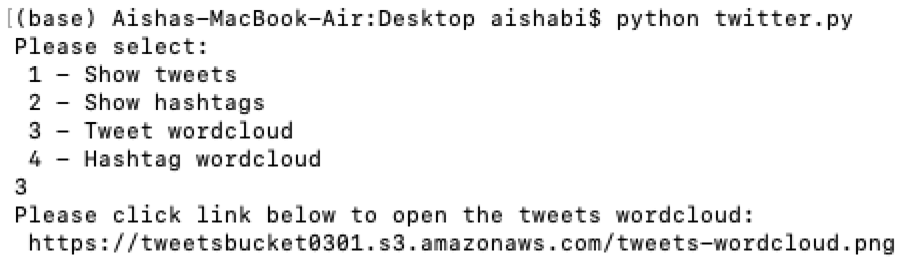
Hashtags:



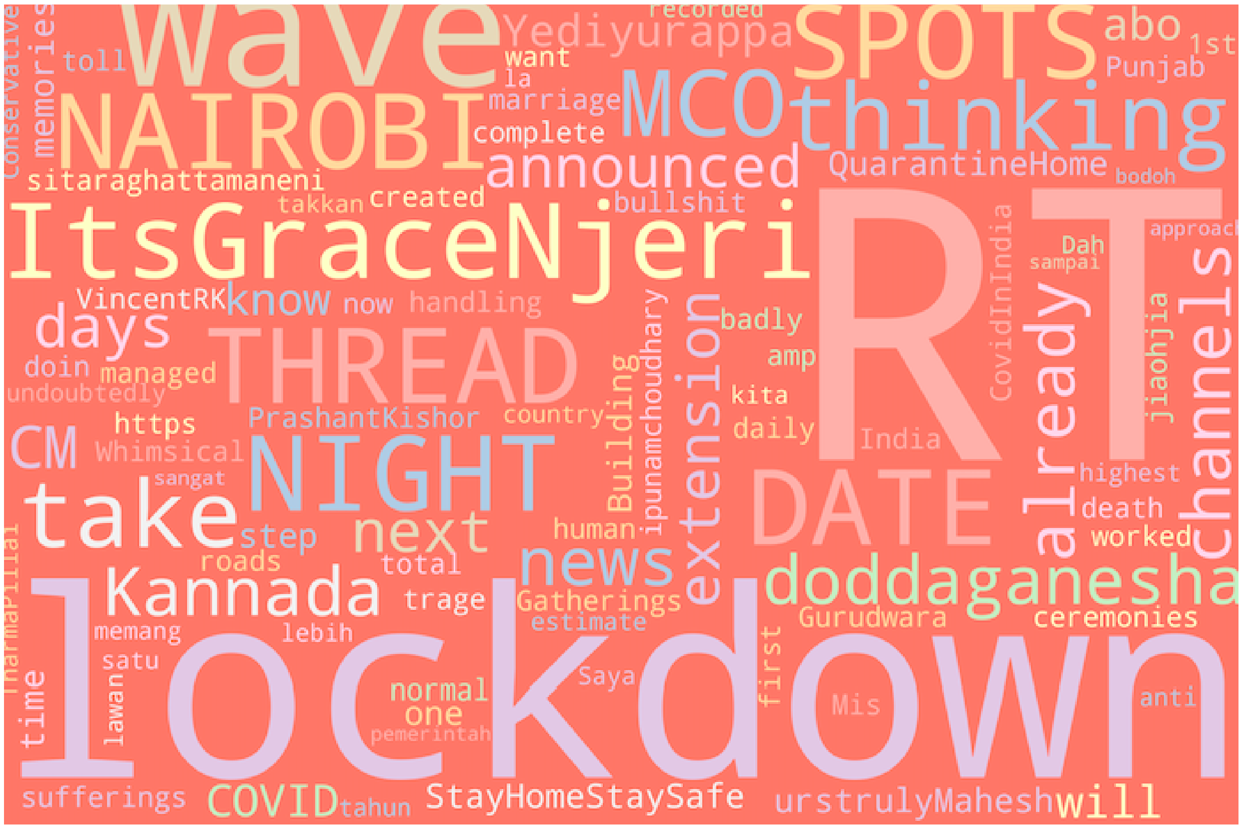
```
[(base) Aishas-MacBook-Air:Desktop aishabi$ python twitter.py
 Please select:
  1 - Show tweets
  2 - Show hashtags
  3 - Tweet wordcloud
  4 - Hashtag wordcloud
 2
 How many hashtage would you like to see?
  Please enter: 10,30 or 50
 10
 Please enter the time frame of this search
  Select how many minutes: 10, 30 or 60
 10
 Trend: ('Labour', 380417)

 Trend: ('Hartlepool', 149624)

 Trend: ('#ElectionResults2021', 16344)

 Trend: ('#VibePayFriday', None)

 Trend: ('Tories', 106372)

 Trend: ('#BandcampFriday', None)

 Trend: ('#FridayFeeling', 15502)

 Trend: ('Grenfell', 14850)

 Trend: ('Corbyn', 82810)

 Trend: ('Starmer', 143457)
```

Wordcloud:

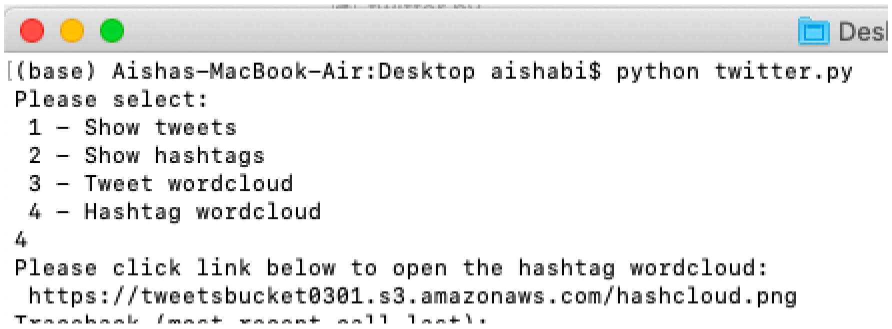
Tweets wordcloud:

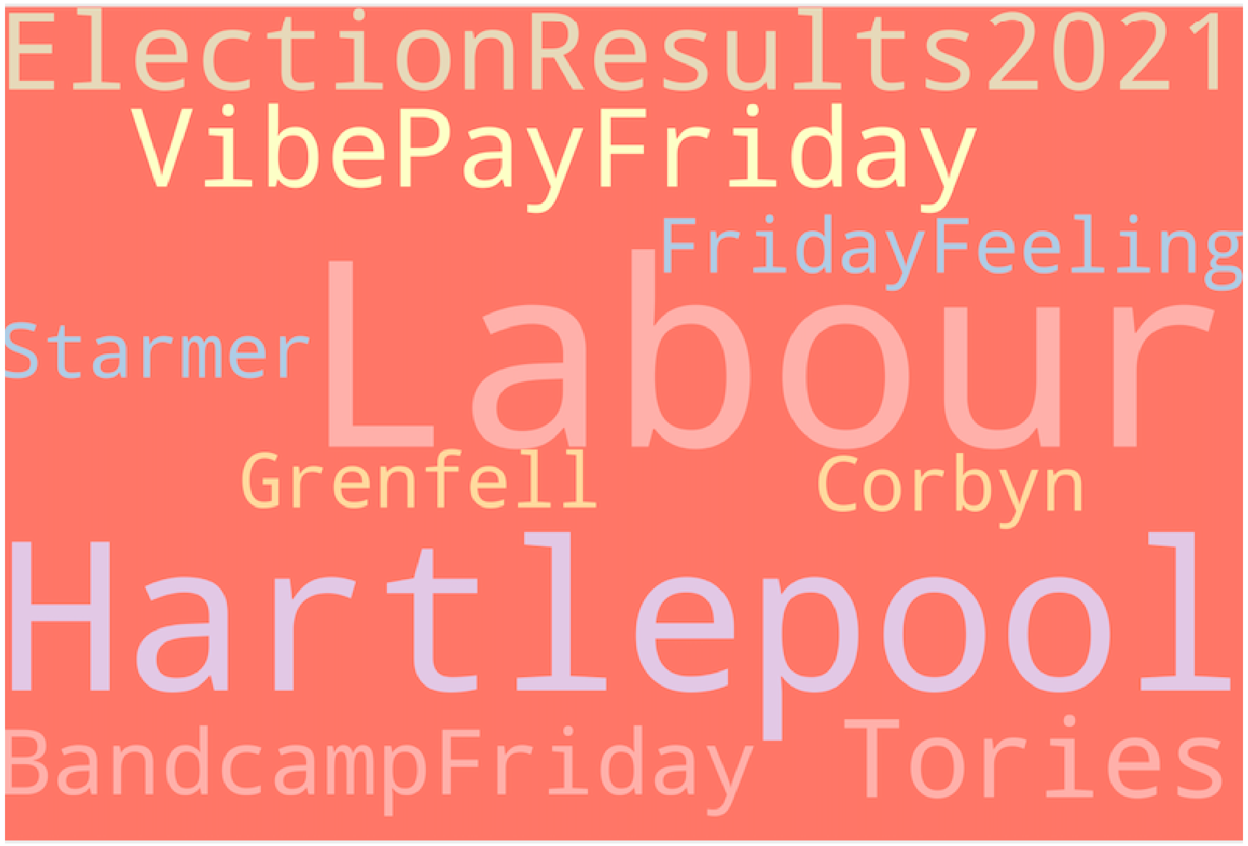


```
[(base) Aishas-MacBook-Air:Desktop aishabi$ python twitter.py
 Please select:
  1 - Show tweets
  2 - Show hashtags
  3 - Tweet wordcloud
  4 - Hashtag wordcloud
 3
 Please click link below to open the tweets wordcloud:
  https://tweetsbucket0301.s3.amazonaws.com/tweets-wordcloud.png
```



Hashtags wordcloud:


```
[(base) Aishas-MacBook-Air:Desktop aishabi$ python twitter.py
Please select:
  1 - Show tweets
  2 - Show hashtags
  3 - Tweet wordcloud
  4 - Hashtag wordcloud
4
Please click link below to open the hashtag wordcloud:
  https://tweetsbucket0301.s3.amazonaws.com/hashcloud.png
```



Conclusion

In conclusion, the project successfully met the objectives created to achieve the aim of designing and developing an application to process real-time data and analytics of a data stream. The lambda function receives real time data from the Twitter API. The lambda function processes the data and creates a dynamic list of tweets which time and amount values collected from the user interface. The application can also create this list for trending hashtags. Both the tweet data and hashtag data can be visualised using a word cloud. The lambda function also analyses the tweet data. Finally, a command line user interface was successfully implemented, collecting the user input and passing the payload and triggering the lambda function. The lambda function processes the tweets through the twitter API and stores the results in the S3 bucket. The python script retrieves this data from the S3 bucket and displays to the user.

References

Amazon, 2021, What is Amazon S3? [online] Available at: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html> [Accessed 6 May 2021].

Anders, M. (2011) Python 3 web development beginner’s guide use Python to create, theme and deploy unique web applications. 1st edition. Olton, Birmingham: Packt Pub.

Baun, C. et al. (2011) Cloud Computing. Dordrecht: Springer, pp 25-29.

Erdoğan, N. (2019) Developing with S3 : AWS with Python and Boto3 series.

Rountree, D. et al. (2014) The basics of cloud computing : understanding the fundamentals of cloud computing in theory and practice . 1st edition. Amsterdam: Syngress, an imprint of Elsevier.

Tweepy, 2021 API Reference — tweepy 3.5.0 documentation. [online] Available at: <https://docs.tweepy.org/en/v3.5.0/api.html> [Accessed 7 May 2021].

Twitter, 2021, Getting access to the Twitter API. [online] Available at: <https://developer.twitter.com/en/docs/twitter-api/getting-started/getting-access-to-the-twitter-api> [Accessed 7 May 2021].