CAB432 – Cloud Computing

Assignment Two:

‘Cloud Tweeter’

The Queensland University of Technology

Semester Two, 2020

Student: Gavin Smith

Student Number: n10138196

25th October, 2020

Contents

Introduction

Purpose & description

'Cloud Tweeter' is a JavaScript powered web application that prompts the user for a keyword to search. Upon searching the User is then displayed a live twitter feed which they can stop and start anytime with the hit of a button, while also being able to research for another tweet while the app is running. Furthermore, the application encompasses 'Natural API", providing users with a sentiment analysis of every tweet. The application is shown in Figures One and Two;

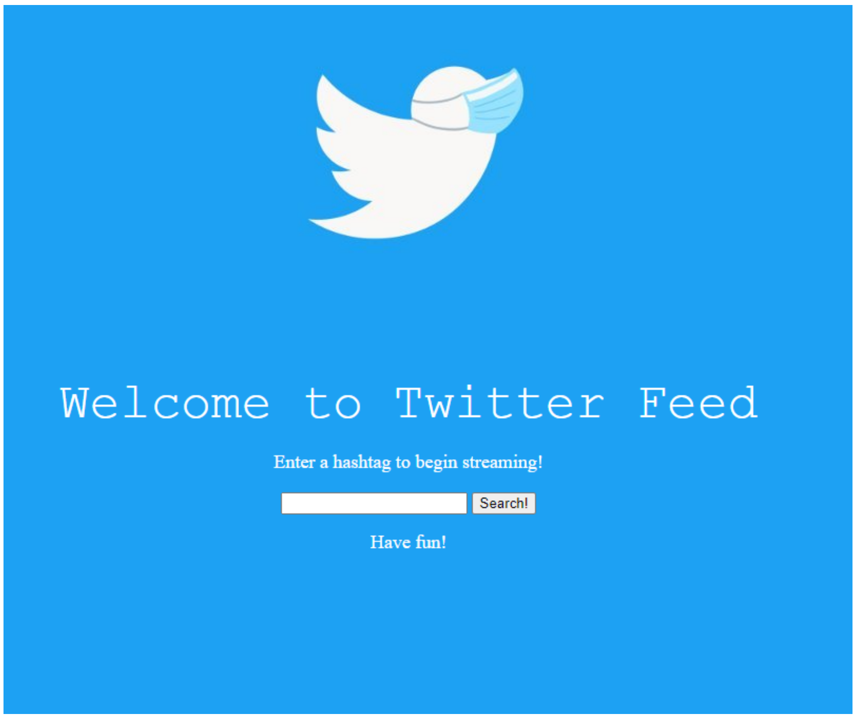
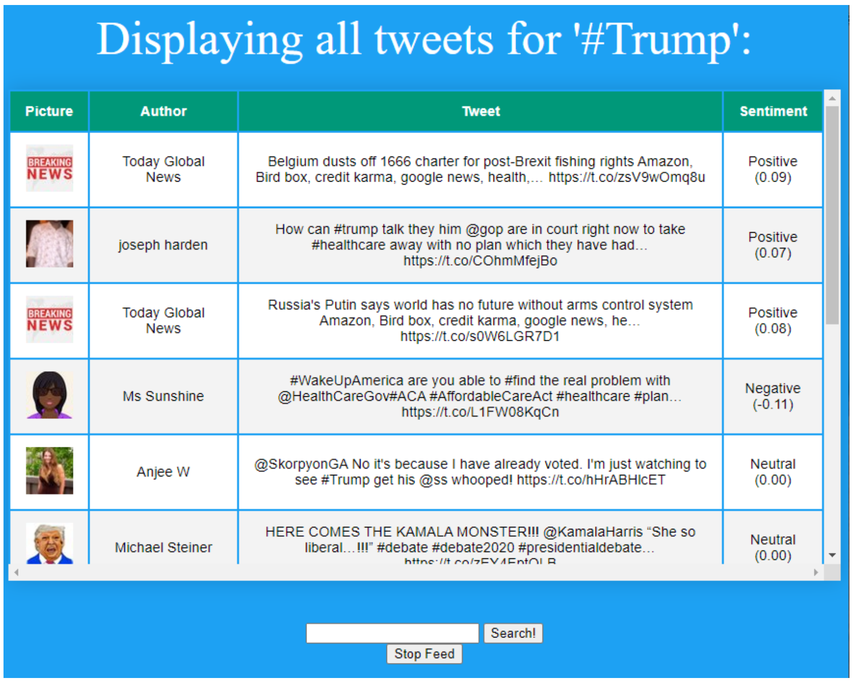


Figure One - Home Page



Displaying all tweets for '#Trump':

Picture	Author	Tweet	Sentiment
	Today Global News	Belgium dusts off 1666 charter for post-Brexit fishing rights Amazon, Bird box, credit karma, google news, health,... https://t.co/zsV9wOmq8u	Positive (0.09)
	joseph harden	How can #trump talk they him @gop are in court right now to take #healthcare away with no plan which they have had... https://t.co/COhmMfejBo	Positive (0.07)
	Today Global News	Russia's Putin says world has no future without arms control system Amazon, Bird box, credit karma, google news, he... https://t.co/s0W6LGR7D1	Positive (0.08)
	Ms Sunshine	#WakeUpAmerica are you able to #find the real problem with @HealthCareGov#ACA #AffordableCareAct #healthcare #plan... https://t.co/L1FW08KqCn	Negative (-0.11)
	Anjee W	@SkorpyonGA No it's because I have already voted. I'm just watching to see #Trump get his @ss whooped! https://t.co/hHrABHIcET	Neutral (0.00)
	Michael Steiner	HERE COMES THE KAMALA MONSTER!!! @KamalaHarris "She so liberal…!!!" #debate #debate2020 #presidentialdebate... https://t.co/zFY4FntOLR	Neutral (0.00)

Figure Two – Application Page, Returning all instances of '#Trump'

Services Used:

Twitter API (v.1.1)

Twitter API allows the user to post, interact, and retrieve data. We specifically utilised Twitter API as a means of pulling tweets and each tweets corresponding ‘Author’ and ‘Profile Picture’.

Docs: https://developer.twitter.com/en/docs/twitter-api

NaturalNode API

‘NaturalNode’ is a ‘nodejs’ based language processor that allows users to query a string of text and analyses the information providing the user with a number of outputs ranging from *string distances, classifications, phonetics* and much more. Specifically, we are utilising NaturalNode’s Sentiment Analysis algorithm.

Docs: https://github.com/NaturalNode/natural

Use Cases

US 1

As an	Individual
I want	To track live events based of keywords
So that	I can stay informed on the topic.

Table One – Use Case for the Everyday Person

An individual is hypothetically interested in politics may wish to track the current United States debate. By entering a tweet query such as ‘#Trump’, they will then be presented with a live twitter feed of tweets corresponding to the ‘#Trump’ hashtag. As the application is hosted on AWS EC2, when an event goes viral and multiple users decide to ‘track’ the debate via this application, we can expect traffic requests exceeding the computing capacity of our instance. Therefore, our EC2 will automatically scale up allowing for the application to remain functional when multiple users engage with the application and scale down when the event finishes. The tweets are then cached and stored in Redis and can be accessed for review by the user.

US 2

As a	Business
I want	To track products and ideas
So that	I can adjust my business and sell items that are ‘Talked about’

Table Two – Use Case for a Business

Similar to an Individual User Case described above, a business can utilize this application to keep a live feed of ideas and trends running to provide the organization with information. As multiple businesses incorporate this application, the EC2 will scale when there is peak demand, perhaps during periods such as Christmas Sales, and scale down when demand for products are low and business no longer need tracking. Similar to above, the tweets are then cached and stored in Redis and can be accessed by the business later and utilize the tweet sentiment as a means of analysis.

US 3

As a	Government Body
I want	To track emergencies
So that	We can provide assistance

Table Three – Uses Case for a Government Body

Governments can utilize this application as a means of filtering incident reports and emergencies by following a specific topic such as “#Bushfire”. The user can then keep up to date of all information relating to the topic *Bushfires* and respond accordingly. The more users will, like above, allow for the EC2 instance to scale both up and down as the crisis peaks and then detracts.

Technical breakdown

Architecture

Setting up the architecture, we utilize *Pug* as a HTML JavaScript engine template as a means for rendering our information into a functional display and incorporate Express.js in our back-end development. By setting up our twitter tokens, we have access to the Twitter API which allows us to send requests to the Twitter server. After creating a variable that holds the users search, we are then able to pull all tweets that correspond to the query. Upon receiving the tweet information in a JSON format, we can then perform our data cleanup, whereby we extract all the relevant information being ‘Tweet’, ‘Author’, and ‘Profile Picture’ and then trigger our Natural API to perform a sentiment analysis. We are then able to push all the information to the front end after being ‘stringified’, and caching on Redis, where it is finally formatted, pushed via WebSocket, and served on a Pug template for the user to interact with.

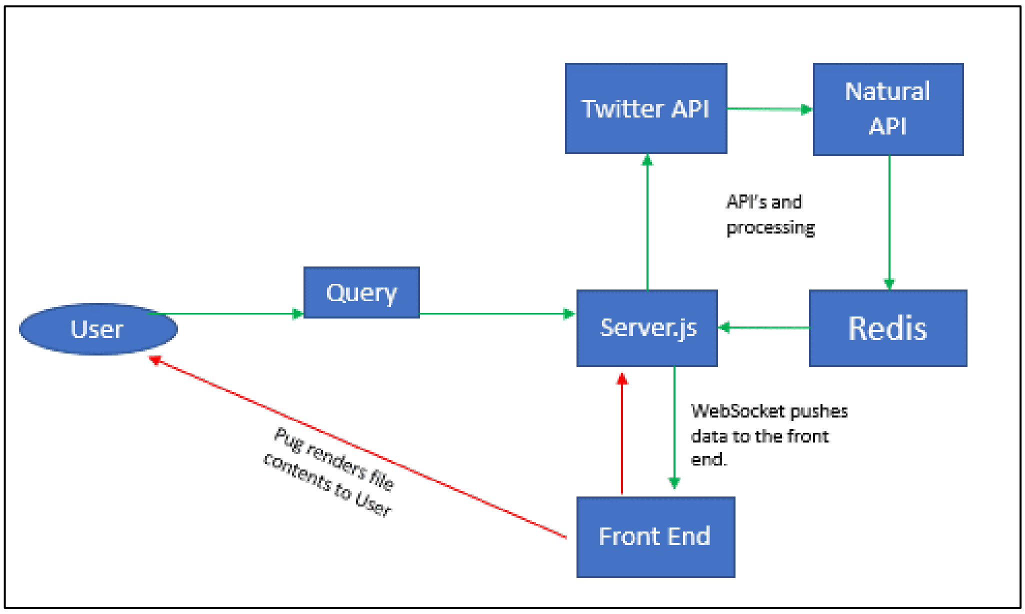
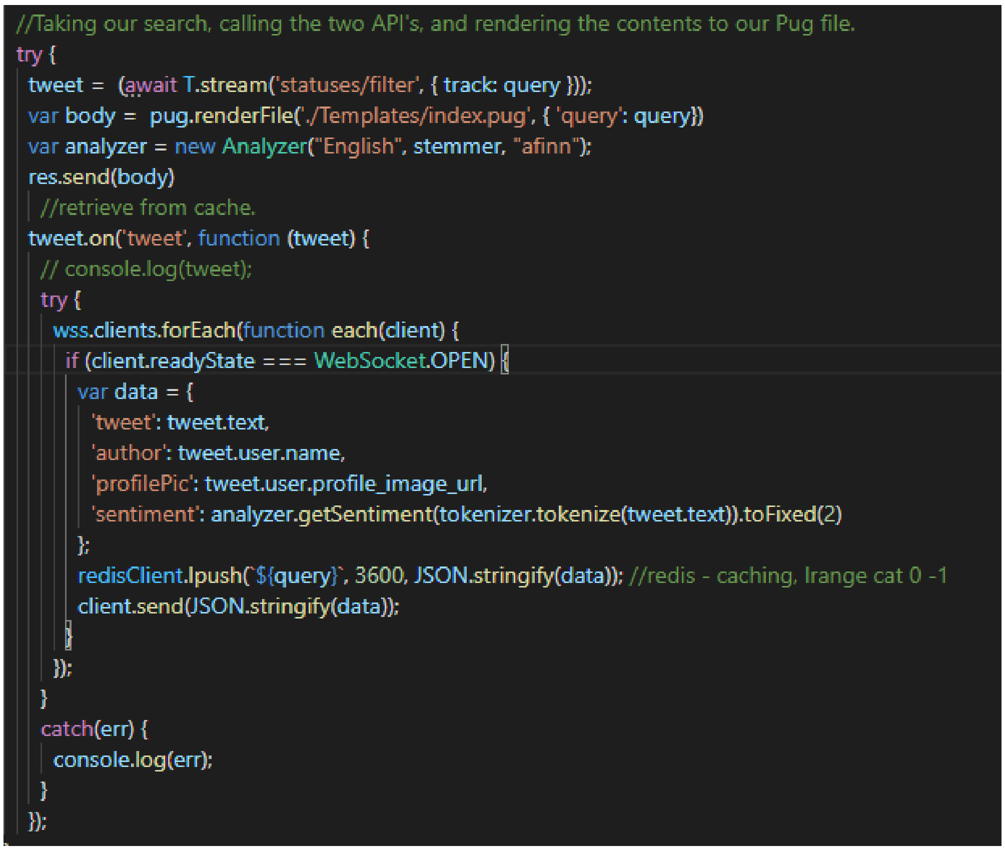


Figure Three – Architecture of Our Application

Response filtering / data object correlation

By analysing the block of code shown below, we are able to discuss the response filtering and data manupulation pefromed in our 'server.js' file. After calling the Twitter API by sending it a query, the API then open a WebSocket and creates a variable data which holds the three twitter elements 'Tweets', 'Author', and 'Picture', while also analysing and performing a sentiment analysis on the 'Tweets' via Natural API and then storing our variables in Redis before finally deploying onto our Pug file for viewing.



```
//Taking our search, calling the two API's, and rendering the contents to our Pug file.
try {
  tweet =  (await T.stream('statuses/filter', { track: query }));
  var body =  pug.renderFile('./Templates/index.pug', { 'query': query})
  var analyzer = new Analyzer("English", stemmer, "afinn");
  res.send(body)
    //retrieve from cache.
  tweet.on('tweet', function (tweet) {
    // console.log(tweet);
    try {
      wss.clients.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
          var data = {
            'tweet': tweet.text,
            'author': tweet.user.name,
            'profilePic': tweet.user.profile_image_url,
            'sentiment': analyzer.getSentiment(tokenizer.tokenize(tweet.text)).toFixed(2)
          };
          redisClient.lpush(`${query}`, 3600, JSON.stringify(data)); //redis - caching, lrange cat 0 -1
          client.send(JSON.stringify(data));
        }
      });
    }
    catch(err) {
      console.log(err);
    }
  });
```

Figure Four – Main Block of Code that Utlises the 2 API's

Scaling and Performance

After setting up a clean AWS instance, we then upload our files to the main directory in our Ubuntu server. After testing initial functionality, our application loads in the AWS cloud as seen below in Figure Five;

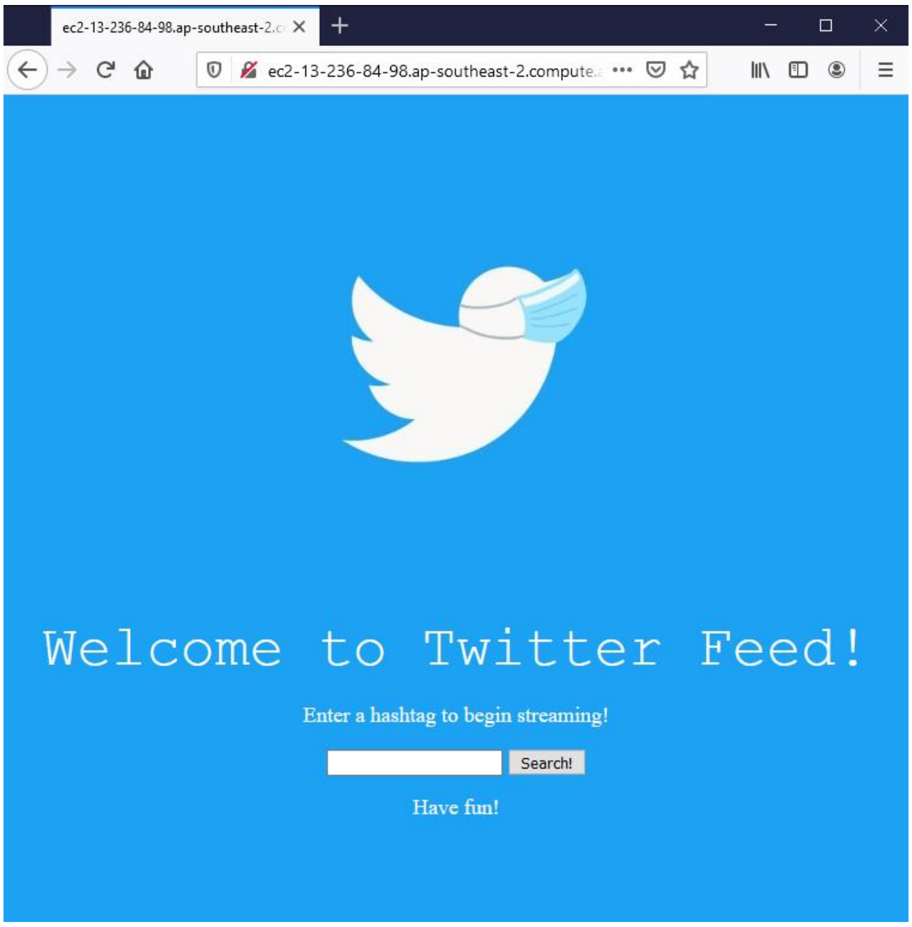
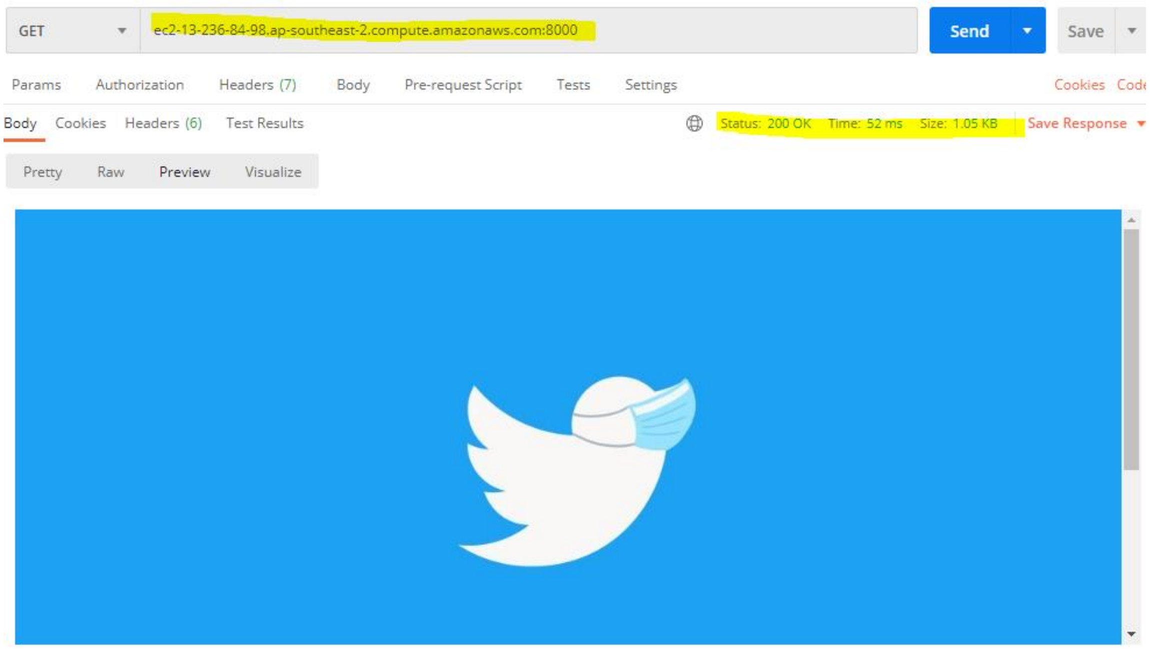

ec2-13-236-84-98.ap-southeast-2.c
ec2-13-236-84-98.ap-southeast-2.compute.
Welcome to Twitter Feed!
Enter a hashtag to begin streaming!
Search!
Have fun!

Figure Five – Application via AWS

By then following the Practical demonstration from earlier in the semester, we build an ‘AWS Image’ and setup our load balancing and scaling configurations. By then navigating to postman, we send a collection of requests to our EC2 server forcing our instances to increase and scale up, and then eventually down once the Postman requests are complete. The procedure and results are shown below in Figure Six and Seven.


GET ec2-13-236-84-98.ap-southeast-2.compute.amazonaws.com:8000

Status: 200 OK Time: 52 ms Size: 1.05 KB

Figure Six – Postman GET Requests.

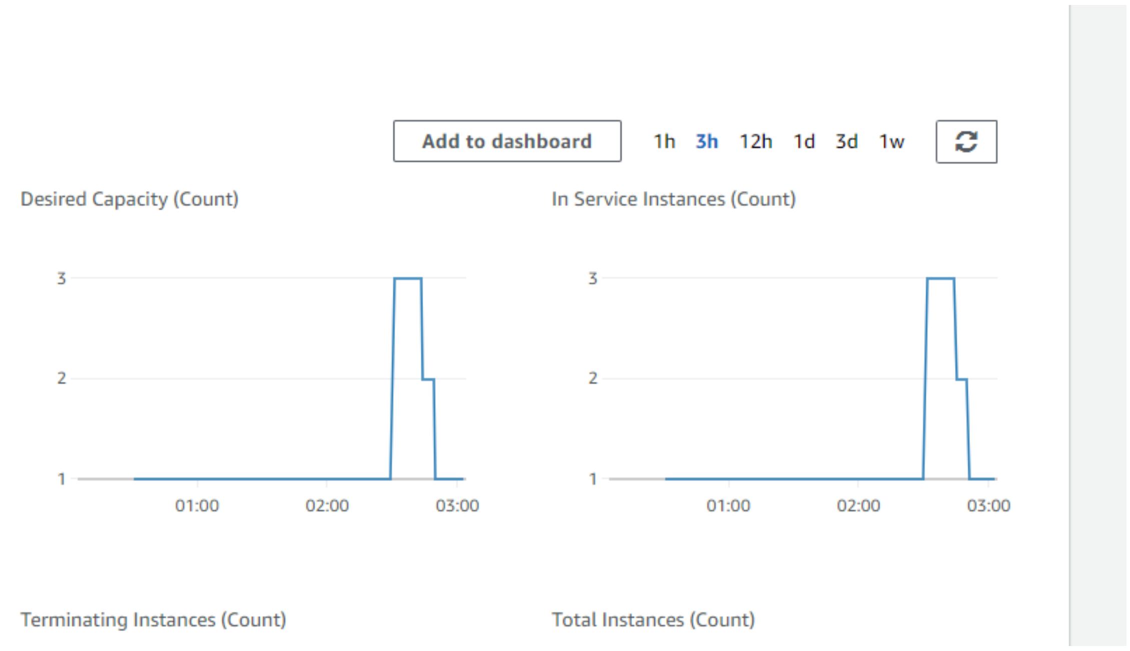
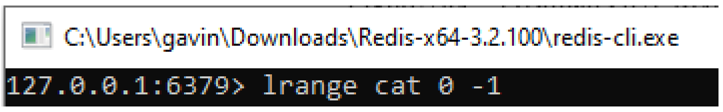


Figure Seven – Evidence of Scaling.

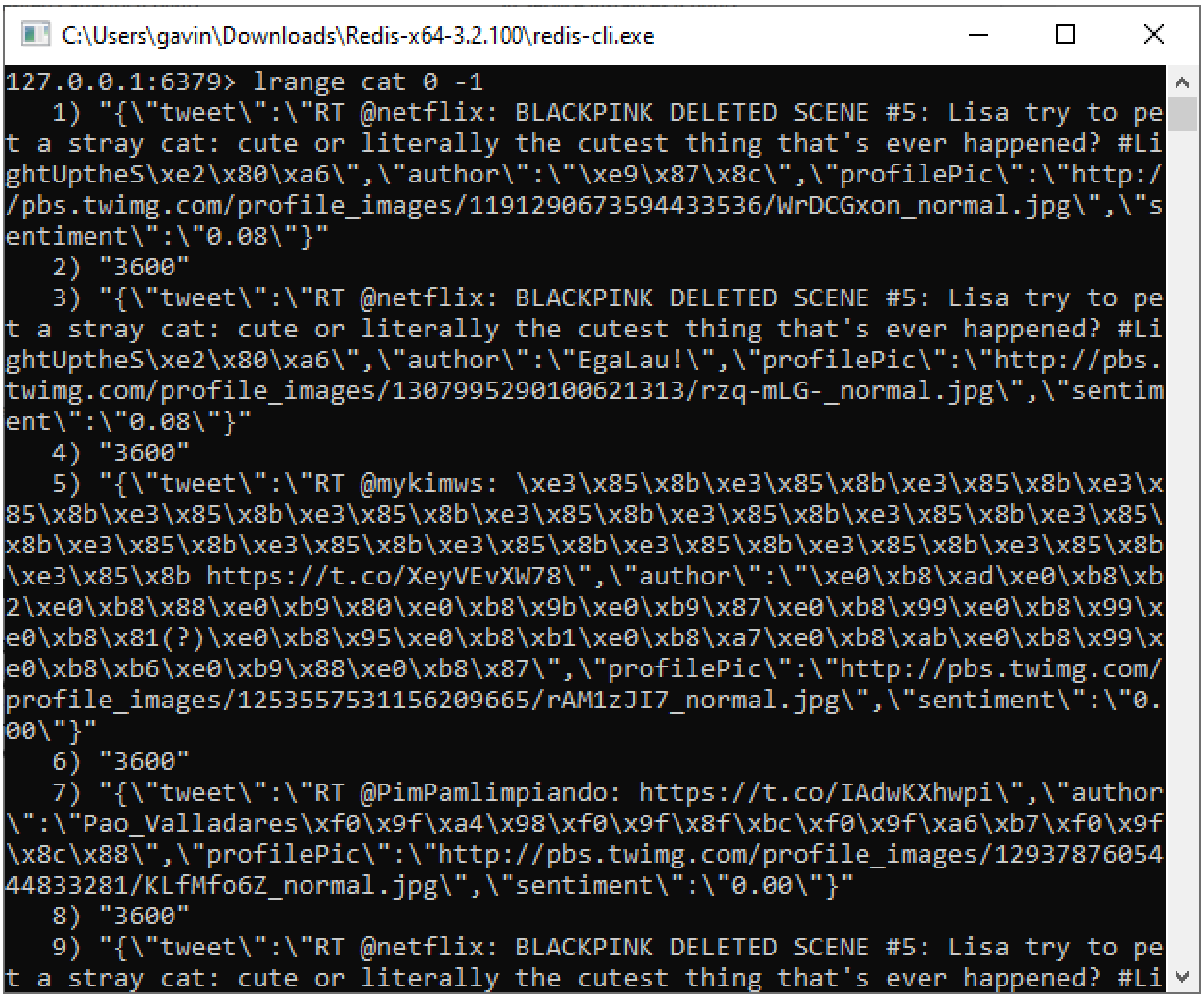
Persistence:

Unfortunately, I was only able to demonstrate one form of persistence, and that is Redis. Redis was used to track the query and then parse the data into a cache that the user would be able to access in the future. By querying ‘cat’ multiple times in our application, we are then able to use the Redis command;



```
127.0.0.1:6379> lrange cat 0 -1
```

Which then returned the tweet information, and author.



```
127.0.0.1:6379> lrange cat 0 -1
   1) "{\"tweet\":\"RT @netflix: BLACKPINK DELETED SCENE #5: Lisa try to pe
t a stray cat: cute or literally the cutest thing that's ever happened? #Li
ghtUptheS\xe2\x80\xa6\",\"author\":\"\xe9\x87\x8c\",\"profilePic\":\"http:/
/pbs.twimg.com/profile_images/1191290673594433536/WrDCGxon_normal.jpg\",\"s
entiment\":\"0.08\"}"
   2) "3600"
   3) "{\"tweet\":\"RT @netflix: BLACKPINK DELETED SCENE #5: Lisa try to pe
t a stray cat: cute or literally the cutest thing that's ever happened? #Li
ghtUptheS\xe2\x80\xa6\",\"author\":\"EgaLau!\",\"profilePic\":\"http://pbs.
twimg.com/profile_images/1307995290100621313/rzq-mLG-_normal.jpg\",\"sentim
ent\":\"0.08\"}"
   4) "3600"
   5) "{\"tweet\":\"RT @mykimws: \xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x
85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\
x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b\xe3\x85\x8b
\xe3\x85\x8b https://t.co/XeyVEvXW78\",\"author\":\"\xe0\xb8\xad\xe0\xb8\xb
2\xe0\xb8\x88\xe0\xb9\x80\xe0\xb8\x9b\xe0\xb9\x87\xe0\xb8\x99\xe0\xb8\x99\x
e0\xb8\x81(?)\xe0\xb8\x95\xe0\xb8\xb1\xe0\xb8\xa7\xe0\xb8\xab\xe0\xb8\x99\x
e0\xb8\xb6\xe0\xb9\x88\xe0\xb8\x87\",\"profilePic\":\"http://pbs.twimg.com/
profile_images/1253557531156209665/rAM1zJI7_normal.jpg\",\"sentiment\":\"0.
00\"}"
   6) "3600"
   7) "{\"tweet\":\"RT @PimPamlimpiando: https://t.co/IAdwKXhwpi\",\"author
\":\"Pao_Valladares\xf0\x9f\xa4\x98\xf0\x9f\x8f\xbc\xf0\x9f\xa6\xb7\xf0\x9f
\x8c\x88\",\"profilePic\":\"http://pbs.twimg.com/profile_images/12937876054
44833281/KLfMfo6Z_normal.jpg\",\"sentiment\":\"0.00\"}"
   8) "3600"
   9) "{\"tweet\":\"RT @netflix: BLACKPINK DELETED SCENE #5: Lisa try to pe
t a stray cat: cute or literally the cutest thing that's ever happened? #Li
```

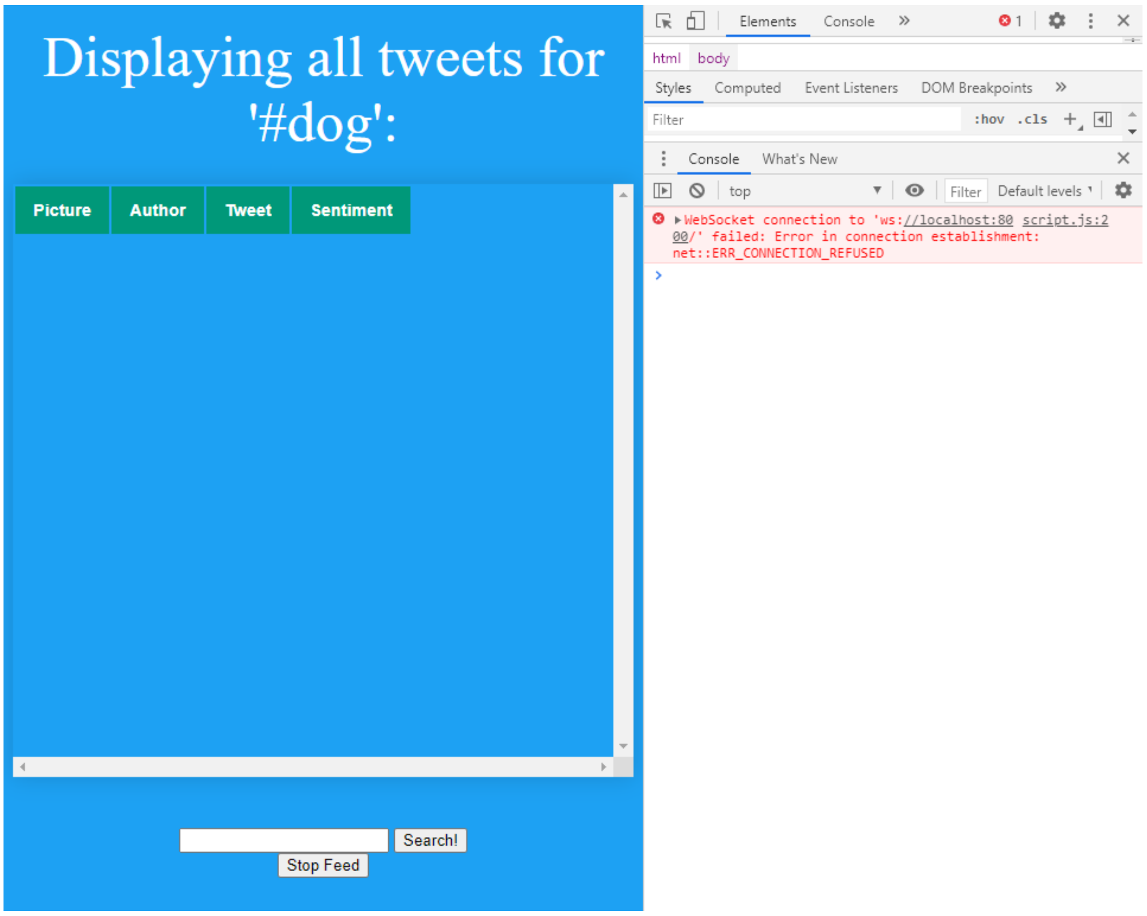
Test Plan:

Task	Expected Outcome	Result	Appendix
Load Home Page	Application loads	PASS	1
Search for '#Coronavirus'	Displays related tweet	PASS	2
While streaming, re-search '#Trump'	Displays related tweet	PASS	3
Test Stop Feed Button	Feed stops	PASS	4
Test Start Feed Button	Feed resumes	PASS	5

Table Four – Testing.

Difficulties / Exclusions / unresolved & persistent errors /

Having used WebSocket's as a means of routing the data to the front end, I have had significant trouble with my application serving on AWS. While my application is functional when run locally, it uploads to AWS however the displaying of the tweets is not shown. Figure Eight below shows the issue at hand. Havin spent the remainder of my time until submission trying to fix this problem, I discovered there was a problem regarding AWS, and WebSocket's on the 22nd of October I trialed my application via AWS to complete the scaling procedure but was met with;



```
WebSocket connection to 'ws://localhost:8000/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED    script.js:2
```

Figure Eight – WebSocket Issues

After countless hours going through 'StackOverflow' I discovered other users were met with the same issue. It turns out when processing the data in our 'script.js' file, I was serving the data to 'Localhost', which was perfect for our local environment, but would cause issues when migrating to AWS. Therefore, the code needed to be changed from;

```
//Local Environment:
const socket = new WebSocket('ws://localhost:8000')
```

Figure Nine – Local Environment.

To;

```
//AWS Environment:
const socket = new WebSocket('ws://3.25.95.67:8000'); //Change to desired IP
```

Figure Ten – AWS Environment.

This change allowed us to dynamically serve our application through our AWS.

While thoroughly enjoyable, I found aspects of the assignment incredibly difficult, running into various problems, particularly with AWS. When debugging and testing, my application wouldn’t run in a local environment at the same time as an Instance was booted, as the twitter keys don’t allow for multiple ‘same time’ requests across platforms, which I found incredibly frustrating.

Furthermore, due to several instances running on AWS I was unable to develop locally and present screenshots of my application across various functions. For this reason, my appendices relating to my tests are somewhat outdated.

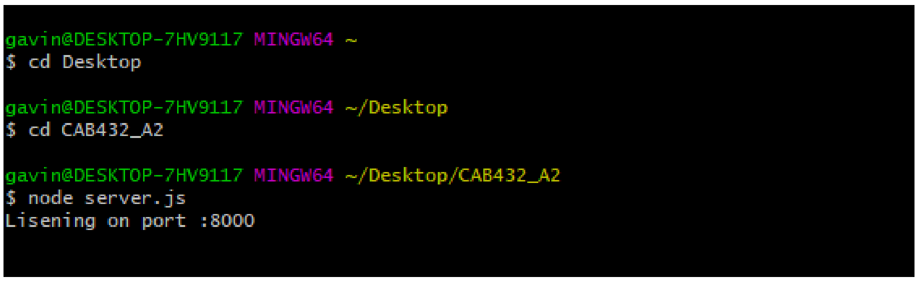
Extensions

If given the opportunity, I would have extended the application by including further API’s, and possibly included a Google Map imaging of tweet locations in the form of a ‘heat map’. I would also extend the application to be more visually aesthetic by incorporating various CSS or Bootstrap elements.

Furthermore, I would’ve liked to have extended this project by demonstrating its capabilities across both AWS and Azure and perform a statistical analysis between the two platforms. I would measure elements such as efficiency, responsiveness, and various other components.

User guide

- Navigate to the Directory that contains 'server.js', and follow the commands shown below in figure INSERT.



```
gavin@DESKTOP-7HV9117 MINGW64 ~
$ cd Desktop

gavin@DESKTOP-7HV9117 MINGW64 ~/Desktop
$ cd CAB432_A2

gavin@DESKTOP-7HV9117 MINGW64 ~/Desktop/CAB432_A2
$ node server.js
Lisening on port :8000
```

Figure Eleven – Navigating Directories.

- Navigate to 'localhost:8000' and you will be met with the home screen;

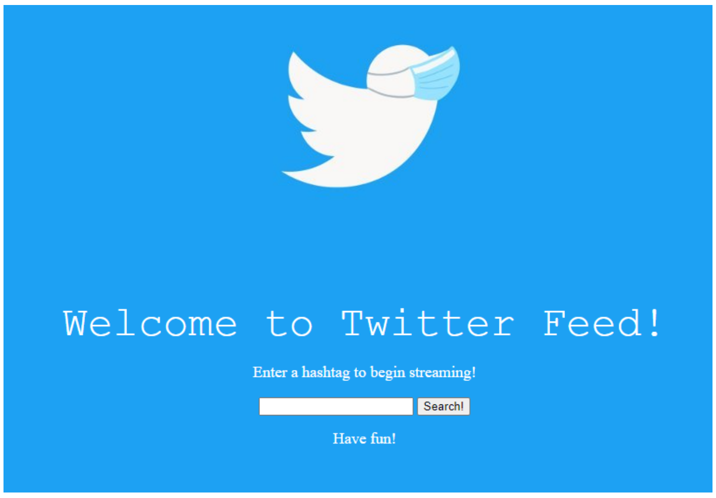
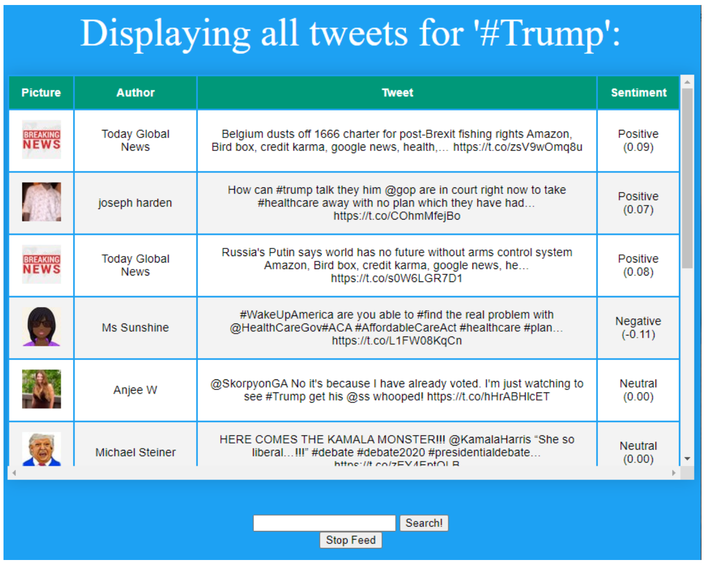


Figure Twelve – Application Home.

- Search for a topic, such as '#Trump, which will yield the results;



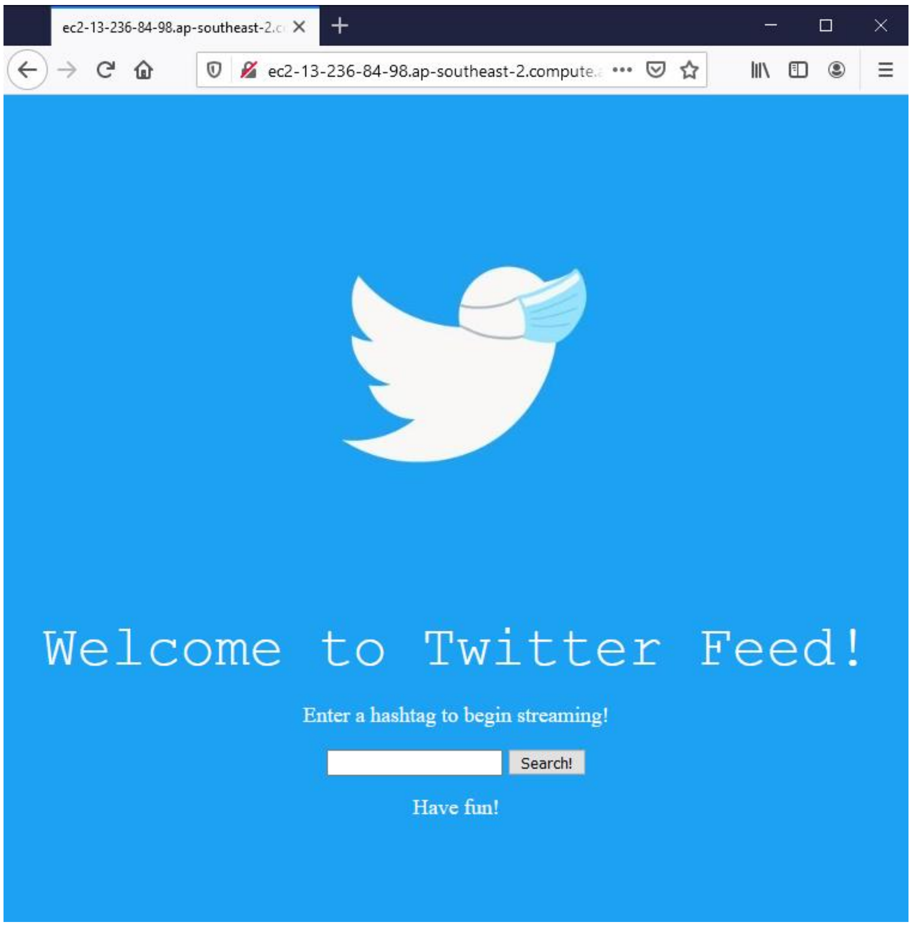
Displaying all tweets for '#Trump':

Picture	Author	Tweet	Sentiment
	Today Global News	Belgium dusts off 1666 charter for post-Brexit fishing rights Amazon, Bird box, credit karma, google news, health,... https://t.co/zsV9wOmq8u	Positive (0.09)
	joseph harden	How can #trump talk they him @gop are in court right now to take #healthcare away with no plan which they have had... https://t.co/COhmMfejBo	Positive (0.07)
	Today Global News	Russia's Putin says world has no future without arms control system Amazon, Bird box, credit karma, google news, he... https://t.co/s0W6LGR7D1	Positive (0.08)
	Ms Sunshine	#WakeUpAmerica are you able to #find the real problem with @HealthCareGov#ACA #AffordableCareAct #healthcare #plan... https://t.co/L1FW08KqCn	Negative (-0.11)
	Anjee W	@SkorpyonGA No it's because I have already voted. I'm just watching to see #Trump get his @ss whooped! https://t.co/hHrABHlcET	Neutral (0.00)
	Michael Steiner	HERE COMES THE KAMALA MONSTER!!! @KamalaHarris "She so liberal…!!!" #debate #debate2020 #presidentialdebate…	Neutral (0.00)

Figure Thirteen – Application in Use.

- The user is then able to enter another query or stop/start the feed using the controls.

Appendices:


ec2-13-236-84-98.ap-southeast-2.c

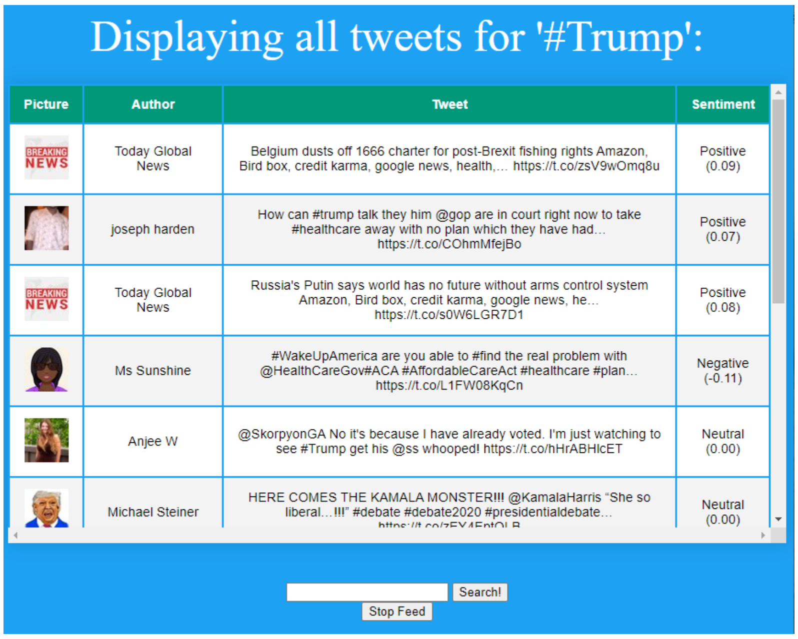
ec2-13-236-84-98.ap-southeast-2.compute.

Welcome to Twitter Feed!

Enter a hashtag to begin streaming!

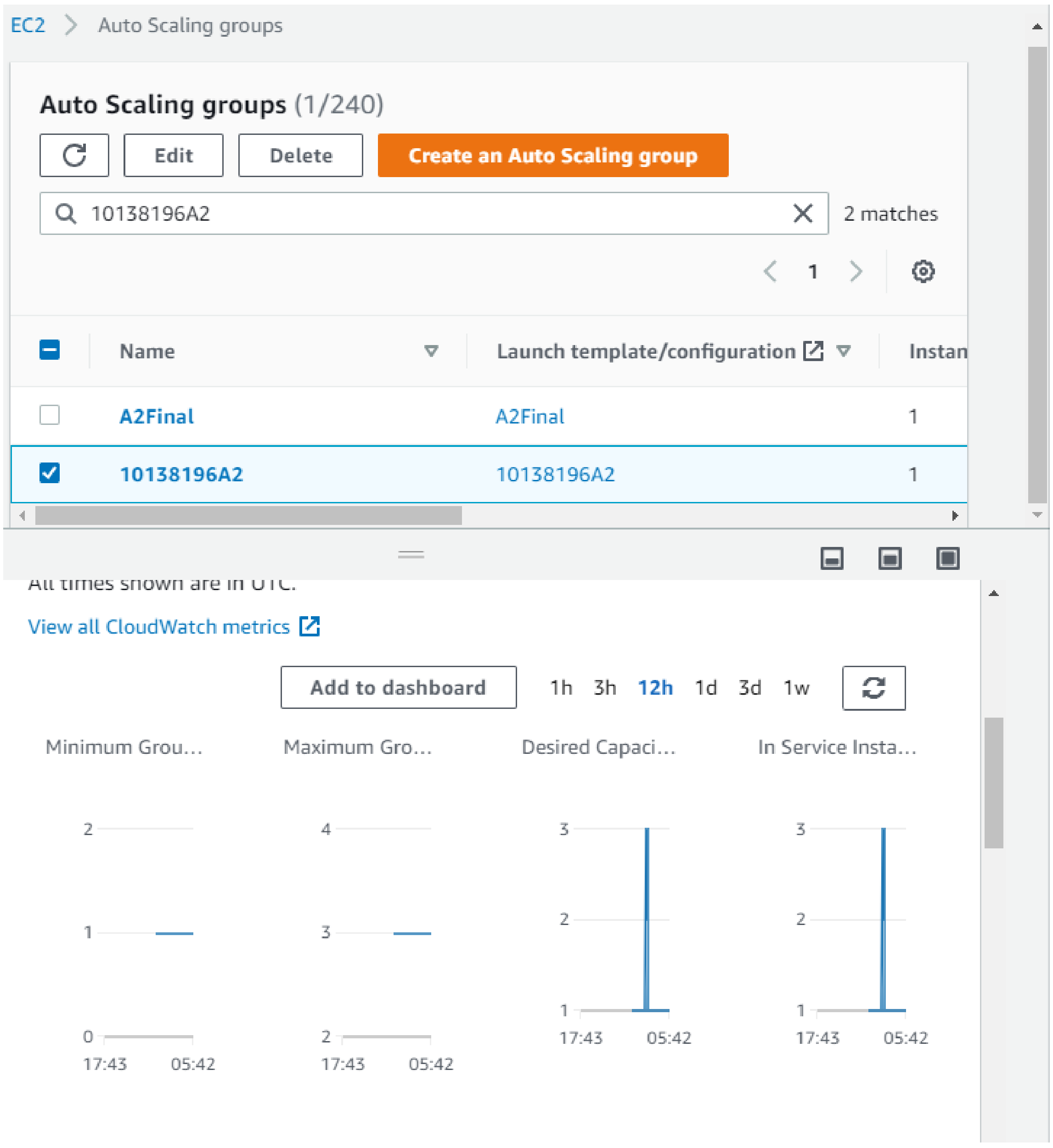
Have fun!

Appendix One – Home Page


Displaying all tweets for '#Trump':

Picture	Author	Tweet	Sentiment
BREAKING NEWS	Today Global News	Belgium dusts off 1666 charter for post-Brexit fishing rights Amazon, Bird box, credit karma, google news, health,… https://t.co/zsV9wOmq8u	Positive (0.09)
	joseph harden	How can #trump talk they him @gop are in court right now to take #healthcare away with no plan which they have had… https://t.co/COhmMfejBo	Positive (0.07)
BREAKING NEWS	Today Global News	Russia's Putin says world has no future without arms control system Amazon, Bird box, credit karma, google news, he… https://t.co/s0W6LGR7D1	Positive (0.08)
	Ms Sunshine	#WakeUpAmerica are you able to #find the real problem with @HealthCareGov#ACA #AffordableCareAct #healthcare #plan… https://t.co/L1FW08KqCn	Negative (-0.11)
	Anjee W	@SkorpyonGA No it's because I have already voted. I'm just watching to see #Trump get his @ss whooped! https://t.co/hHrABHlcET	Neutral (0.00)
	Michael Steiner	HERE COMES THE KAMALA MONSTER!!! @KamalaHarris "She so liberal…!!!" #debate #debate2020 #presidentialdebate…	Neutral (0.00)

Appendix Two, Three, Four, and Five – Application Running



EC2 > Auto Scaling groups

Auto Scaling groups (1/240)

10138196A2 — 2 matches

	Name	Launch template/configuration	Instan
☐	A2Final	A2Final	1
✓	10138196A2	10138196A2	1

All times shown are in UTC.

View all CloudWatch metrics

Appendix Six – Further Scaling Proof linked to the instance of my student number;10138196A2