



2017

# Twitter Streams Sentiment Analysis

A sentiment analysis web application of Real-Time Twitter Stream.

**CAB432 Assignment 2 Report**

**YeeChen Fong - n9602976**

**PengYuan Ge - n8936854**

## Table of Content

<b>Introduction</b>	<b>2</b>
<b>Use Cases &amp; Dependent Services</b>	<b>3</b>
Use Case 1	3
Use Case 2	3
<b>Technical Description</b>	<b>3</b>
Architecture	3
Technology	5
Server-side	5
Client-side	7
Issues & Difficulties	8
Overall Implementation	9
<b>Scaling &amp; Performance</b>	<b>10</b>
<b>Testing &amp; Limitation</b>	<b>11</b>
Test Plan & Results	11
Limitations & Compromises	11
<b>Possible Extensions</b>	<b>12</b>
<b>Reference</b>	<b>12</b>
<b>Appendix</b>	<b>13</b>
Brief User Guide	13
Web Application	13
Necessary downloads	16
Instruction for API Keys	16
Deployment Instructions	17

## Introduction

Twitter Streams Sentiment Analysis aims to provide a resource for users to perform sentiment analysis on Tweets coming in from all around the world in real-time. The service computationally identifying and categorizing opinions expressed in a tweets, to determine whether the tweet writer's attitude towards a particular topic is positive, negative, or neutral. Moreover, the service provides the visualisation of the sentiment of tweets based on a live topic filter which the user can modify it by entering or removing “keywords”. These visualisation comes in the form of a Sentiment Activity (Real-time line chart) and a Sentiment Topic (Topic bar chart).

The architecture of the the application is divided into 3 sections, *Tweet Stream*, *Sentiment Analysis Processing*, and *Application Client*. The *Tweet Stream* section consist of a M4 EC2 instance which retrieves Tweets from a Tweet Stream. The *Sentiment Analysis Processing* section consist of an Elastic Load Balancer which manages an Auto-Scaling group consisting of T2 EC2 instance/instances. The Load Balancer receives tweets from the M4 instance and passes the request to the auto-scaling group instance with the lowest load, the instance processes the tweet and stores it in a DynamoDB. The *Application Client* section consist of a T2 EC2 instance which retrieves the tweet object from the DynamoDB and displays it to the front-end.

Services used by the to develop this application includes Twit (a Twitter API client for Node.js), Sentiment (a Sentiment module for Node.js), Socket-IO (a JavaScript socket library for realtime bi-directional communication between web clients and servers), AWS-SDK Js (a JavaScript library that enables developers to use AWS services in node.js), Chart.js (a JavaScript charting tool), Request Js (a HTTP/HTTPS request library for Node.js) and the usual HTML5, CSS, Bootstrap, Javascript, JQuery, EJS, Express and Node.js.

The development process is divided into 3 phases. It begins with developing and testing of the Stream server, Sentiment Analysis server, and the Application client. The next phase is deploying the packages into 3 AWS instances followed by creating the AMIs respectively. The last phase is the create a Launch Configuration using the Sentiment Analysis server AMI, followed by creating an Auto-Scaling group, Load Balancer, and attach the Load Balancer to the Auto-Scaling group.

## Use Cases & Dependent Services

### Use Case 1

As an **user**, I want the application to **filter tweets based on topics** so that **I can see the sentiment of tweets on topics that I am interested**.

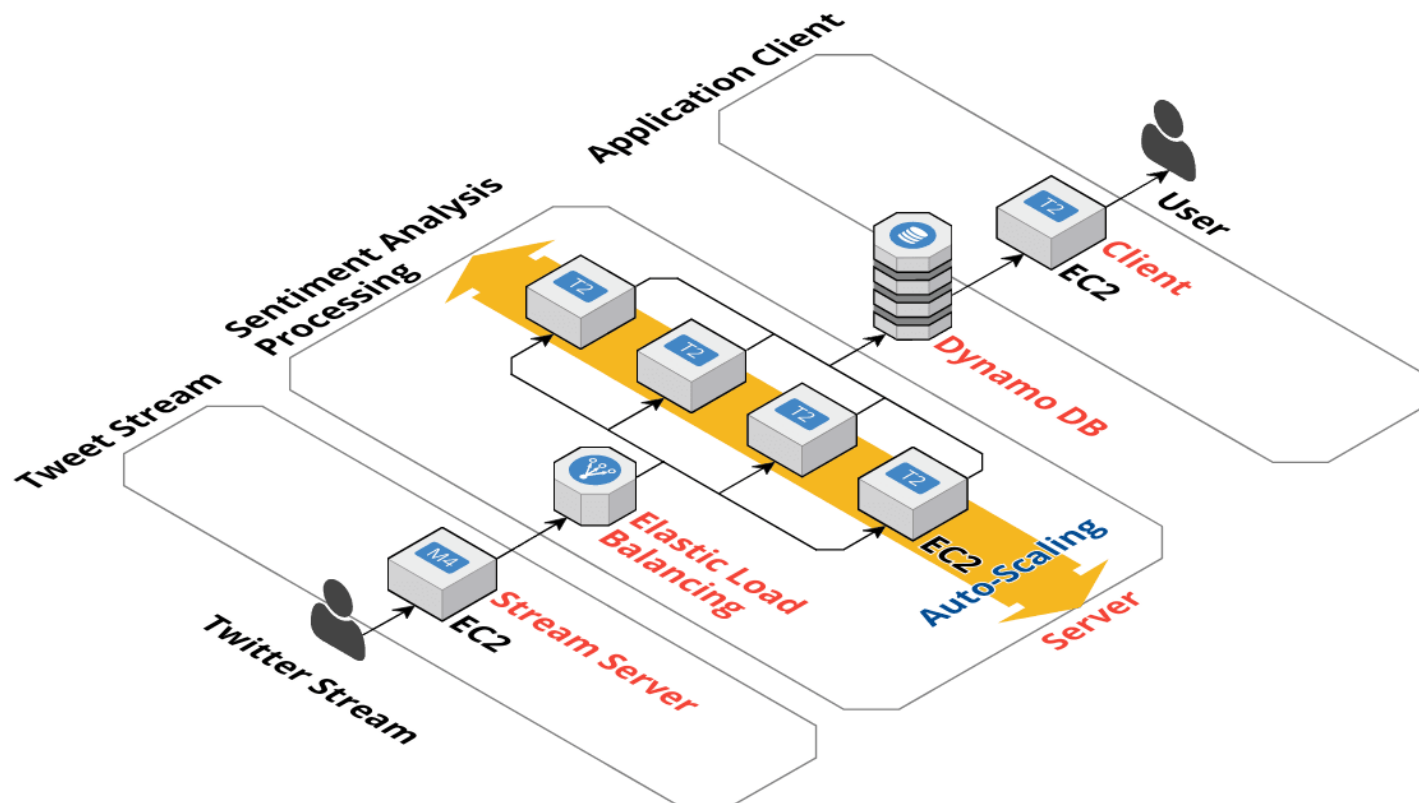
### Use Case 2

As a **user**, I want the application to **visualise the the overall sentiment, the number of positive, negative and neutral tweets, the number of tweets related to the keywords in real-time** so that **I can envision the public's general attitude towards a particular topic when tweeting**.

## Technical Description

### Architecture

The architecture of the the application is divided into 3 sections, ***Tweet Stream***, ***Sentiment Analysis Processing***, and ***Application Client***.



***Tweet Stream section (only server-side)***

The *Tweet Stream* section consist of a M4 EC2 instance which retrieves Tweets from a Tweet Stream from the public Twitter Endpoint. It sends a POST request to the elastic load balancer domain name on port 3000 for each tweet coming in.

***Sentiment Analysis Processing section (only server-side)***

The *Sentiment Analysis Processing* section consist of an Elastic Load Balancer which manages an Auto-Scaling group consisting of T2 EC2 instance/instances. Every incoming POST request from the Stream Server in the *Tweet Stream* section is handled by the Elastic Load Balancer. The Load Balancer check for any server in the Auto-Scaling group which has the lowest load and passes the POST request to it. The server accepts the POST request with the tweet in JSON and performs the Sentiment Analysis on the text of the tweet. A sentiment rating for the tweet is returned from the Sentiment library which is then processed to categorise the “emotion” of the tweet in either one of the three category, “Positive”, “Neutral”, and “Negative”. The sentiment ratings and the “emotion” is then combined with the tweet to create a *Tweet Object* which is then passes into a query to be stored in a DynamoDB.

***Application Client section (client and server-side)***

The *Application Client* section consist of a T2 EC2 instance which receives “keywords” from the user and retrieves the recently stored *Tweet Object* related to the keyword from the DynamoDB and displays it to the front-end every 1 second. The “emotion” of the *Tweet Object* is counted and added to a real-time line chart and a topic bar chart. The real-time line chart refreshes every 5 seconds, show the number of “Positive”, “Neutral”, and “Negative” and total tweets coming in, whilst the topic bar chart shows the number of “Positive”, “Neutral”, and “Negative” tweets for each topic/keyword the tweets are related to. An overall counter is also displayed which shows the overall sentiment of all the tweets coming in.

## Technology

### Server-side

- **Node JS [1]:**  
Open-source, cross-platform, asynchronous javascript runtime framework for building event driven network applications. The node.js server is used to handle serving files as well as send requests and receive response from APIs, and Databases. Used in the *Tweet Stream*, *Sentiment Analysis Processing*, and *Application Client* section.
- **Express JS [2]:**  
A simple web framework that sets the routes and views easily, allowing the client and server-side to be integrated easily, which allows for rapid development. Used in the *Sentiment Analysis Processing*, and *Application Client* section.
- **Request JS [3]:**  
A simple HTTP/HTTPS request library that makes HTTP/HTTPS calls the simplest way possible. GET and POST request are made and data is returned back in JSON format which can be easily converted to Javascript objects. Used in the *Tweet Stream* section.
- **Twit [4]:**  
A Twitter API client for Node.js, supports both the REST and Streaming API. In this application it is used to receive tweet streams. Used in the *Tweet Stream* section.
- **Sentiment [5]:**  
A Sentiment module for Node.js that uses the AFINN-165 wordlist and Emoji Sentiment Ranking to perform sentiment analysis on arbitrary blocks of input text. Returns sentiment rating of tweets. Used in the *Sentiment Analysis Processing* section.
- **Socket-IO [6]:**  
A JavaScript library for realtime web applications. It enables realtime, bi-directional communication between web clients and servers via sockets. Used in the *Application Client* section.
- **AWS-SDK Js [7]:**  
A JavaScript library that enables developers to use AWS services in node.js. It is

used to send and retrieve *Tweet objects* to and from the DynamoDB. Used in the *Sentiment Analysis Processing* section.

- **AWS Elastic Load Balancer:**

An AWS service function that automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses. It is able to manage application traffic in a single Availability Zone or across multiple Availability Zones. Used for the auto-scaling group in the *Sentiment Analysis Processing* section.

- **AWS Auto-Scaling Group:**

An AWS service function that allows dynamically scaled EC2 capacity up or down automatically according to conditions define. Auto Scaling allows automatic increase in the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs. Used in the *Sentiment Analysis Processing* section.

- **AWS DynamoDB:**

A NoSQL database service that stores and retrieves any amount of data, and serve any level of request traffic. Unfortunately, the scaling feature of the database is not available for AWS student accounts.

## Client-side

- **HTML5 & CSS3:**

Markup and stylesheet language used to build web pages. HTML provides the structure of the page and CSS styles the pages, setting the colors, layout and fonts.

- **JavaScript:**

This client-side of Javascript enables interactive features and dynamic functions. It is used to count the emotions of tweets and displays the overall sentiment.

- **Bootstrap:**

Most popular HTML, CSS and JavaScript framework used to develop simple and consistent UI for web page. It contains the HTML and CSS based design template for typography, forms, buttons, navigation and other interface components. The web pages in this services are styled using Bootstrap components.

- **JQuery:**

A Javascript library used for animating effect, event listening and event handling. It is used to handle the real-time and tweet topics visualisation.

- **EJS, Embedded JavaScript templates [8]:**

A template of Express framework. Javascript can be written directly with the HTML markup in EJS template. It also allows Javascript objects to be rendered from the server side (Node.js) to the client side(EJS) with ease.

- **Chart.js [9]:**

A simple JavaScript charting tool. Used to display the real-time tweet counts and topic counts.



## Issues & Difficulties

### 1. One tweet stream per IP address (Twitter Rate Limits)

This prevents user from changing the track words filter by adding new tweet streams if an existing tweet stream exist.

### 2. Tweet stream word tracklist cannot be modified during connection (Twitter Rate Limits)

This prevents user from modifying the word tracklist without disconnecting the existing stream and connecting a new tweet stream.

### 3. Excessive connection attempts causes tweet stream to be throttled (Twitter Rate Limits)

This prevents user from changing the track words filter on the stream as it is not possible to change the filter directly on the Twitter public endpoints without doing a disconnect and reconnect. The problem is, doing a disconnect and reconnect too many times causes throttling and even IP address ban for a period of time. Thus Twitter stream filter endpoints has to be pre-defined “A” - “Z” beforehand.

### 4. Predefined Tweet Stream filter

Keywords “A” - “Z” are overly represented in the tweet stream. Any other keywords has significantly less occurrence to be displayed in the front-end.

### 5. Generating Load

The predefined filter makes it impossible for the application to generate/reduce load via incrementing/decrementing filter keywords. Thus auto-scaling could not be done on the client.

### 6. Non-scalable DynamoDB

The AWS student account do not allow the creation of IAM (Identity and Access Management) roles. The IAM entity is needed to allow the DynamoDB to scale to stores and retrieves any amount of data, and serving any level of request traffic. The DynamoDB's write capacity units and read capacity units has to be manually set (in this application its 25 write capacity units and 5 read capacity units).

## Overall Implementation

### ***Tweet Stream section (only server-side)***

An app.js file create a tweet stream connection to Twitter using a pre-defined track of characters “A” - “Z”. Every single Incoming tweets passing the stream filter is then POST requested to the domain name of the Elastic Load Balancer. The server files are stored in an AMI (Amazon Machine Image) which can deployed at any time.

### ***Sentiment Analysis Processing section (only server-side)***

For the Sentiment Analysis server, apart from the www.js and app.js file which creates the node server, an index.js file is created which allows the incoming POST request with the tweet to be processed by the sentiment library and to be stored in the DynamoDB. The Sentiment Analysis server files are stored in an AMI.

The AMI is used to create the Launch Configuration, followed by an Auto-Scaling group with 2 policies to be trigger after breaching the alarm threshold:

- Increase by an instance when a Scaling group instance’s Maximum CPU Utilisation  $\geq 40\%$  for 60 seconds.
- Decrease an instance when a Scaling group instance’s Maximum CPU Utilisation  $\leq 10\%$  for 60 seconds.

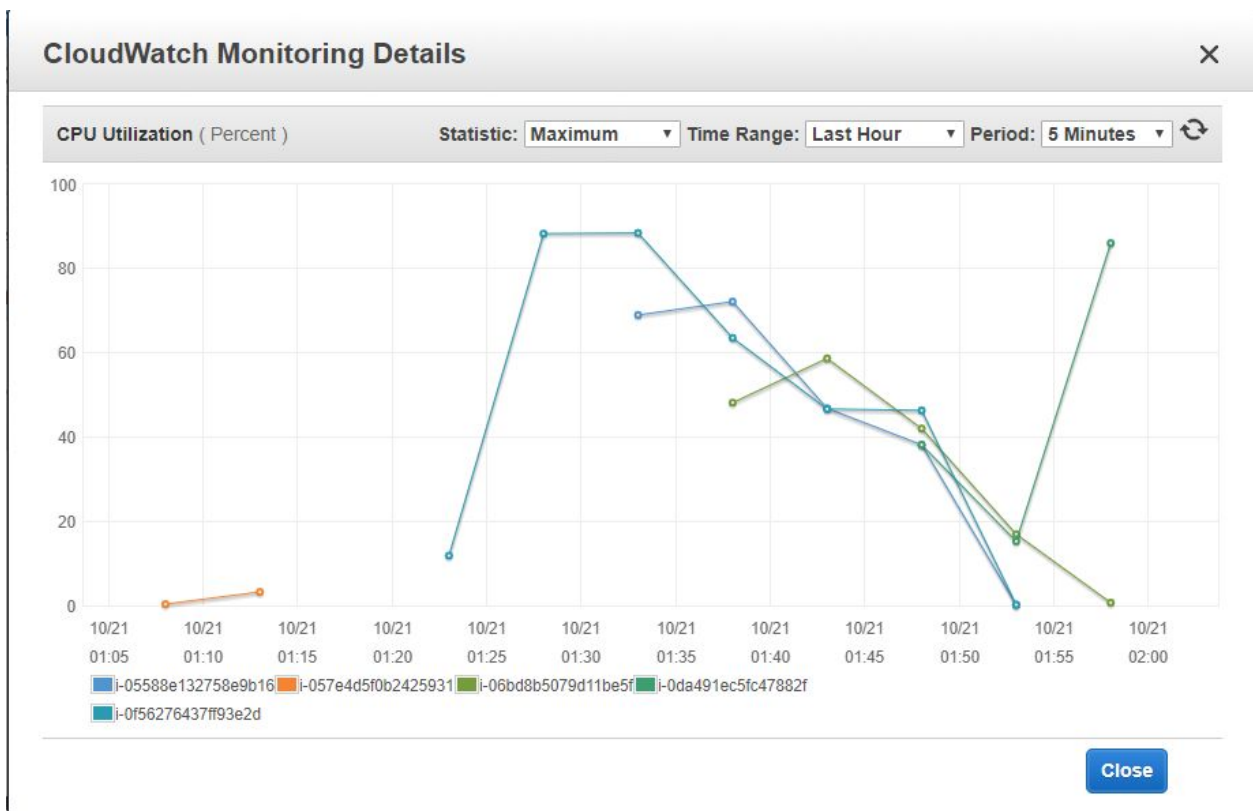
An Elastic Load Balancer is created with the Load Balancer Protocol set to HTTP and port 3000, the Instance Protocol set to HTTP and port 3000. The health check is set to ping at the instances via TCP at port 3000 with response timeout at 10 seconds, health check intervals at 60 seconds, unhealthy threshold at 2 consecutive health check failures, and healthy threshold at 5 consecutive health check successes.

### ***Application Client section (client and server-side)***

Apart from the www.js and app.js file which creates the node server, a socket.js file is created to constantly receive one second old stored tweet from the DynamoDB. The socket.js receives keywords from the user and whenever a tweet with text related to the keywords matches, the tweet is then send via socket to the client index.js which then processes the “emotion” counts and visualise the results in a real-time chart and a topic chart.

## Scaling & Performance

As mentioned in the Architecture and Overall Implementation sections, the scaling of the instances occurs in the comes from the *Sentiment Analysis Processing* section. The scaling is possible through the load generated via processing the tweets and storing the tweets in the DynamoDB. To be safe, a retrieving function to get the exact tweet that was just stored is also implemented in the Sentiment Analysis server to generate more load.



In the diagram shown above, when the the stream server in the *Tweet Stream* section starts (at 01.23 am PST; servers are located in Oregon, US), the first server in the auto-scaling group takes roughly 5 minutes to generate huge load before peaking at around 87% maximum CPU utilisation. The auto-scaling group then keeps on adding more instances until all instances in the scaling group has less than 40% maximum CPU utilisation (not shown in diagram due to the stream server stops early).

When the stream server stops, the maximum CPU utilisation in some instances drops below 10% and the scaling policy starts to terminate some instances. This continues until one instance remains (not shown in the diagram, but it works).

## Testing & Limitation

### Test Plan & Results

ID	Purpose	Expected/Actual	Pass?	ScreenLink
1	Insert keywords	E: Display tweets with the matching topic A: Display tweets with the matching topic	Pass	1.0 - 2.0
2	Clear Real-Time chart	E: Real-Time line chart is cleared A: Real-Time line chart is cleared	Pass	3.0
3	Clear Tweet Topic chart	E: Tweet topic bar chart is cleared A: Tweet topic bar chart is cleared	Pass	4.0
4	Clear tweet counts and overall sentiment	E: Tweet counts and overall sentiment are cleared A: Tweet counts and overall sentiment are cleared	Pass	5.0, 8.0, 9.0
5	CPU exceeds 40% in an auto-scaling instance	E: Increase an instance in the auto-scaling group A: Increase an instance in the auto-scaling group	Pass	N/A
6	CPU fails below 10% in an auto-scaling instance	E: Decrease an instance in the auto-scaling group A: Decrease an instance in the auto-scaling group	Pass	N/A
7	Load Balancer	E: Balances load well among the auto-scaling instances A: Balances load well among the auto-scaling instances	Pass	6.0

### Limitations & Compromises

#### 1. Predefined Tweet Stream filter:

Keywords "A" - "Z" are overly represented in the tweet stream. Any other keywords has significantly less occurrence to be displayed in the front-end. Unfortunately, this has to be compromised as Twitter does not allow too many disconnect and reconnection of tweet stream. Twitter would also throttled the tweet stream and even IP ban the streaming client due to excessive disconnect and reconnect.

#### 2. Generating Load:

The predefined filter stream a lot of tweet, which allows the Sentiment Analysis servers to process and store the Tweet object in the DynamoDB to generate load. But load wasn't generated quick and it wasn't enough to demonstrate the scaling policies, thus the Sentiment Analysis servers is coded to retrieves the same Tweet object that has been stored immediately to generate more load.

## Possible Extensions

A feature that could be added is to display the total number of positive and negative words from all the tweet on the web application. This allows the user to view most common to least common positive/negative words. This can be done by getting the positive and negative words of each tweet via the sentiment library and perform a count on each of them. The total count of these positive and negative words is then displayed on a word cloud where the most common positive/negative words is the largest, whilst the least common positive/negative words is the smallest.

## Reference

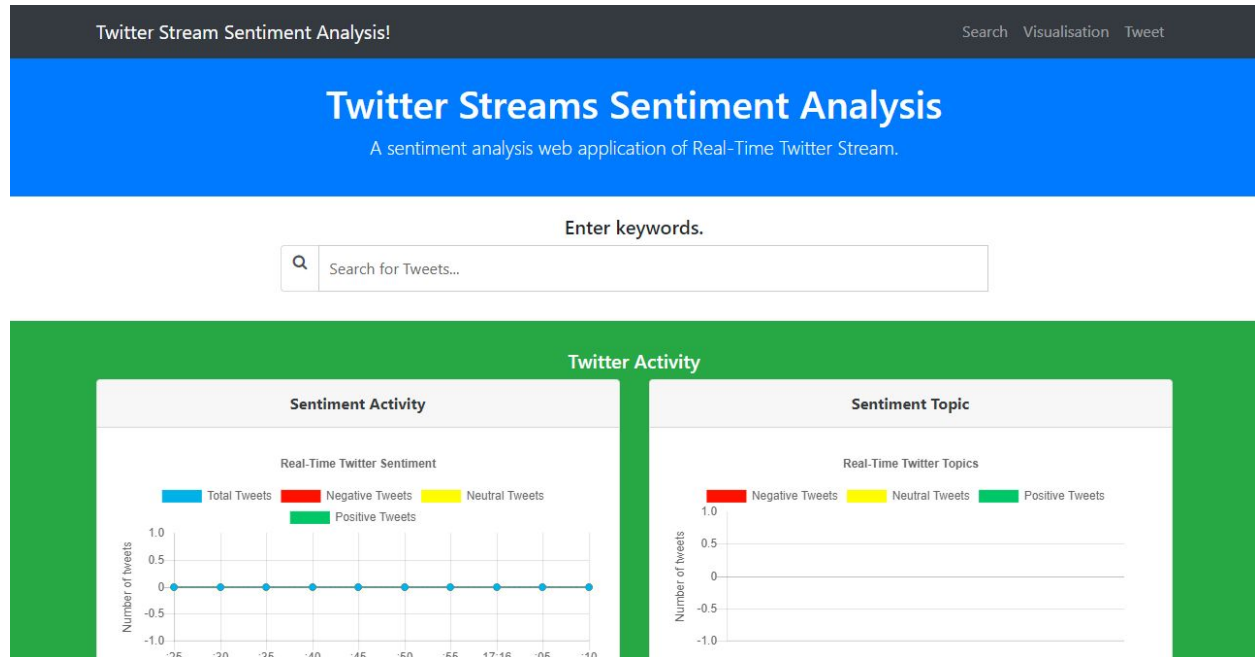
- [1] N. Foundation, "Node.js", *Node.js*, 2017. [Online]. Available: <https://nodejs.org/en/>. [Accessed: 16- Sep- 2017].
- [2] Express, "Express - Node.js web application framework", *Expressjs.com*, 2017. [Online]. Available: <https://expressjs.com>. [Accessed: 16- Sep- 2017].
- [3] M. Rogers, "request", npm, 2017. [Online]. Available: <https://www.npmjs.com/package/request>. [Accessed: 22- Oct- 2017].
- [4] A. Sliwinski, "sentiment", npm, 2017. [Online]. Available: <https://www.npmjs.com/package/sentiment>. [Accessed: 22- Oct- 2017].
- [5] T. Tezel, "twit", npm, 2017. [Online]. Available: <https://www.npmjs.com/package/twit>. [Accessed: 22- Oct- 2017].
- [6] Socket IO, "Socket.IO", *Socket.io*, 2017. [Online]. Available: <https://socket.io/>. [Accessed: 22- Oct- 2017].
- [7] Amazon Web Services, "aws-sdk", npm, 2017. [Online]. Available: <https://www.npmjs.com/package/aws-sdk>. [Accessed: 22- Oct- 2017].
- [8] EJS, "EJS -- Embedded JavaScript templates", *Ejs.co*, 2017. [Online]. Available: <http://ejs.co/>. [Accessed: 16- Sep- 2017].
- [9] Chart JS, "Chart.js | Open source HTML5 Charts for your website", *Chartjs.org*, 2017. [Online]. Available: <http://www.chartjs.org/>. [Accessed: 22- Oct- 2017].

## Appendix

### Brief User Guide

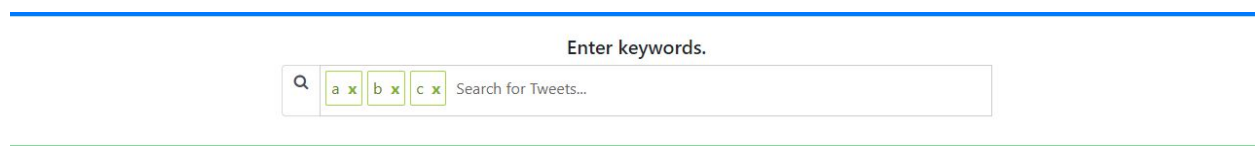
### Web Application

#### 1.0



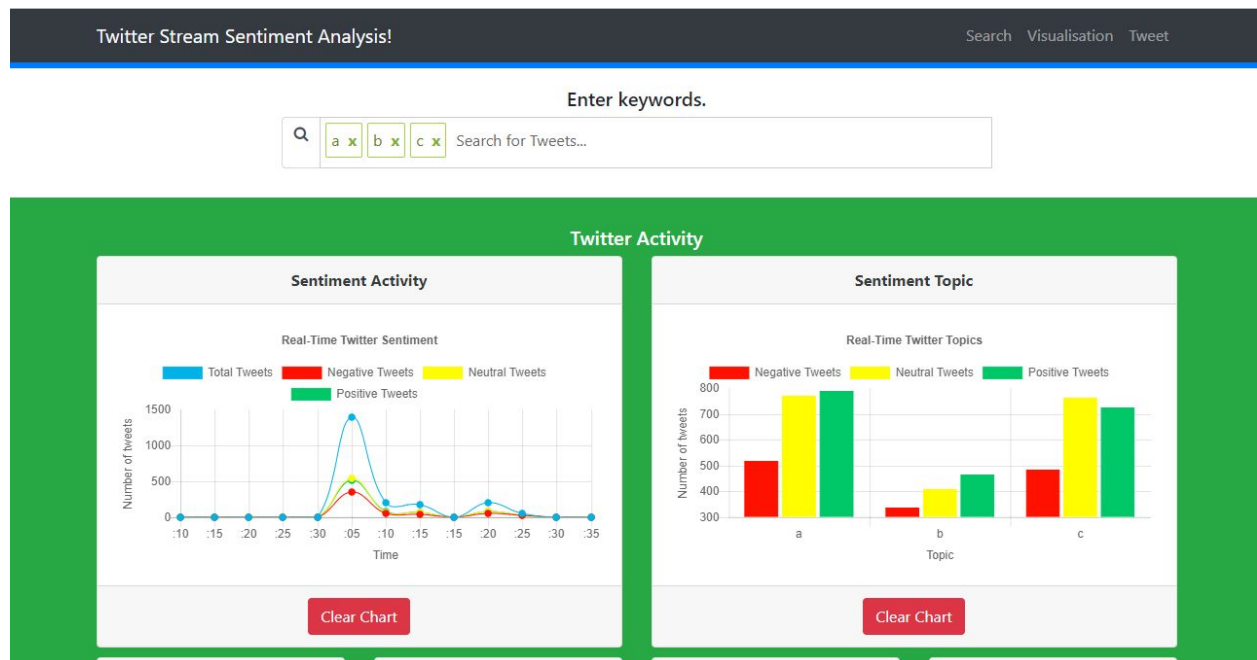
1. This is the web application page. Enter some “keywords” to start searching for tweets with sentiment result.

#### 2.0



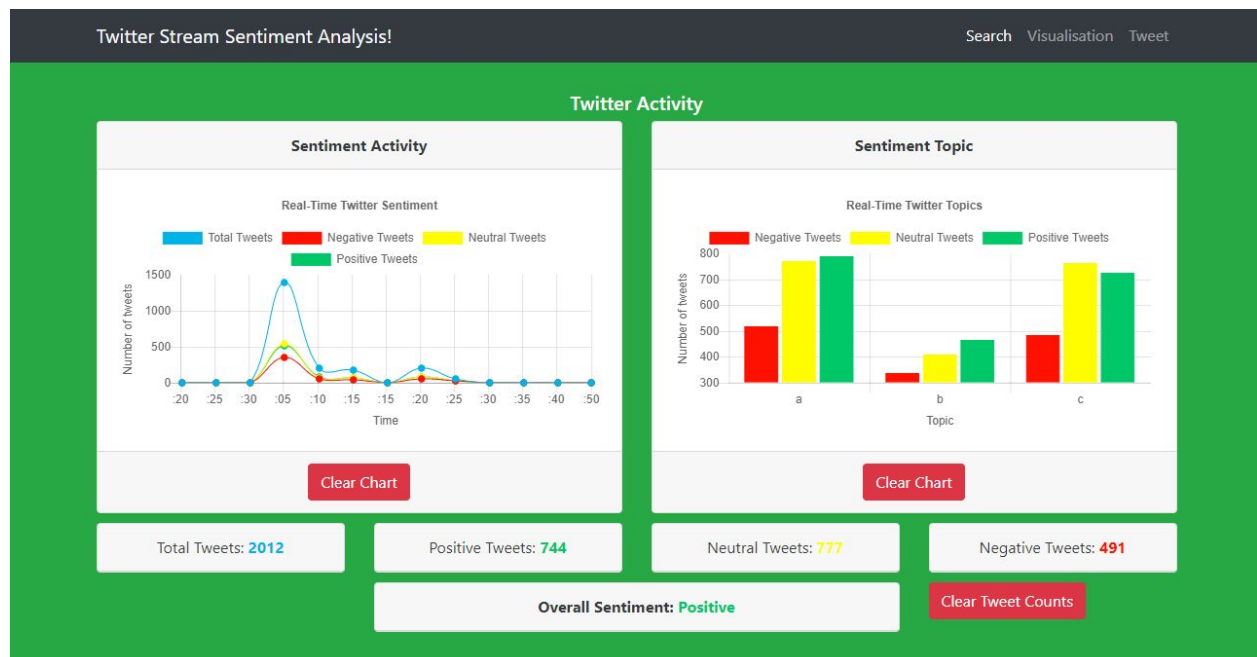
2. You can enter any “keyword” you desire. Let’s go with a, b, and c (since these letter would display lots of tweet).

### 3.0



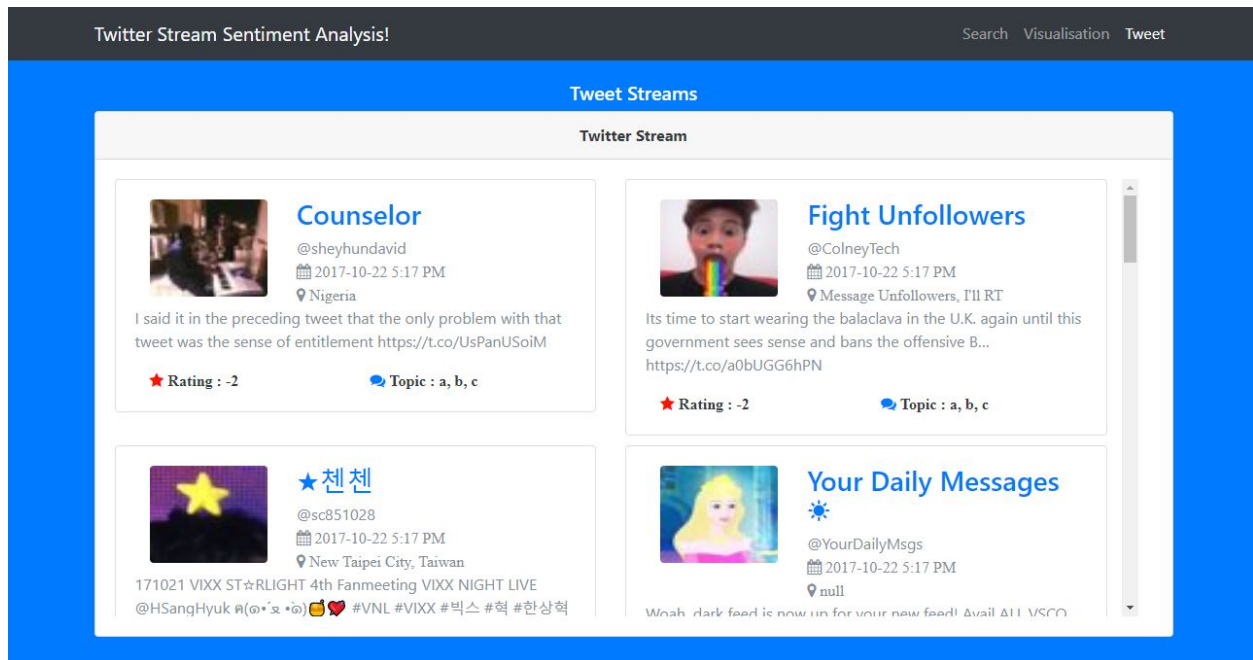
3. Watch as the Real-time line chart (known as Sentiment Activity) and the Topic bar chart (known as Sentiment Topic) starts to populate .

### 4.0



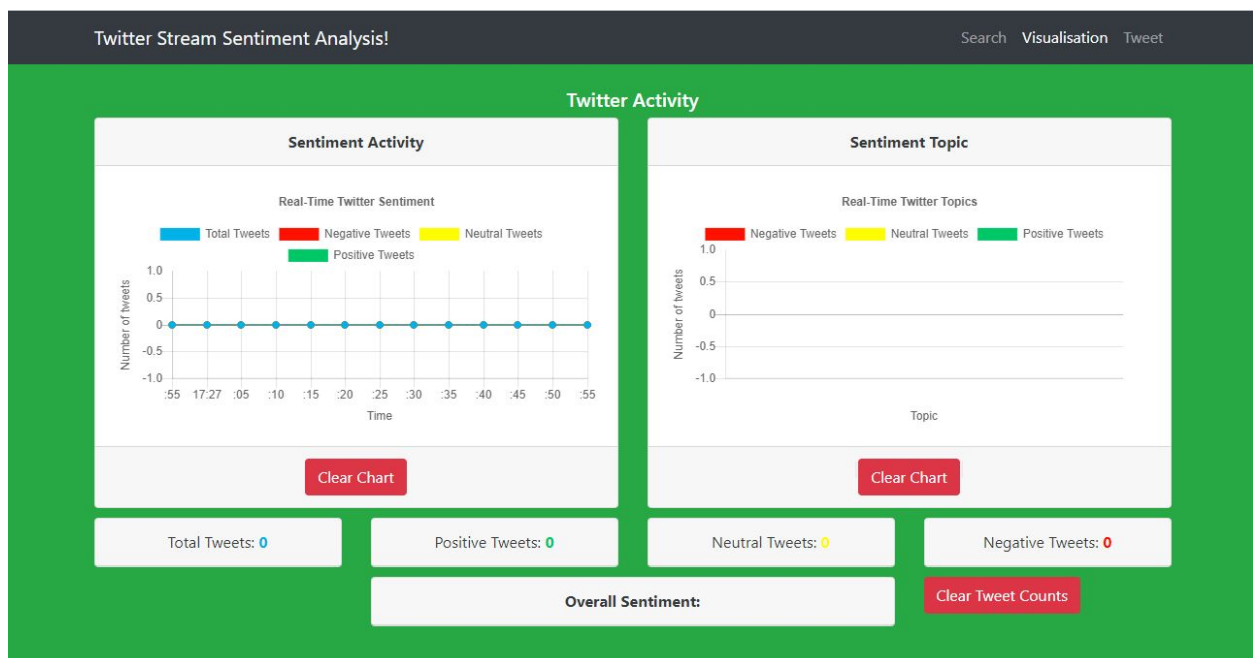
4. The total, positive, neutral, and negative tweet count is also displayed.

## 5.0



5. Here you can look at the tweets displayed.

## 6.0



6. Clear the charts and tweet counts by pressing on the “Clear Chart” and “Clear Tweet Count” buttons.



## Necessary downloads

### ***Tweet Stream section - Stream server***

- Request - <https://www.npmjs.com/package/request>, npm install request
- Twit - <https://www.npmjs.com/package/twit>, sudo npm install twit

### ***Sentiment Analysis Processing section - Sentiment Analysis server***

- AWS SDK - <https://www.npmjs.com/package/aws-sdk>, sudo npm install aws-sdk
- Body parser - <https://www.npmjs.com/package/body-parser>, sudo npm install body-parser
- Cookie parser - <https://www.npmjs.com/package/cookie-parser>, sudo npm install cookie-parser
- Dataformat - <https://www.npmjs.com/package/dateformat>, sudo npm install dateformat
- Express - <https://www.npmjs.com/package/express>, sudo npm install express
- Sentiment - <https://www.npmjs.com/package/sentiment>, sudo npm install sentiment
- Request - <https://www.npmjs.com/package/request>, sudo npm install request

### ***Application Client section - Client***

- AWS SDK - <https://www.npmjs.com/package/aws-sdk>, sudo npm install aws-sdk
- Body parser - <https://www.npmjs.com/package/body-parser>, sudo npm install body-parser
- ChartJS - <https://www.npmjs.com/package/chartjs>, sudo npm install chartjs
- Cookie parser - <https://www.npmjs.com/package/cookie-parser>, sudo npm install cookie-parser
- EJS - <https://www.npmjs.com/package/ejs>, sudo npm install ejs
- Express - <https://www.npmjs.com/package/express>, sudo npm install express
- Socket IO - <https://www.npmjs.com/package/socket.io>, sudo npm install socket.io

## Instruction for API Keys

None!!

## Deployment Instructions

1. Create a new T2 EC2 instance for Stream server, Sentiment Analysis server, and Application Client. Allow port 22 (SSH), 80 (HTTP), and 3000 (Node.js) to be accessible.
2. Deploy the code in the respective EC2 instance in this order:
  - `$ sudo apt-get update`
  - `$ sudo apt-get install git`
  - `$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -&& sudo apt-get install -y nodejs`
  - `$ sudo apt-get install npm`
  - `$ sudo npm install npm -g`
  - `$ git clone https://gitlab.com/markyeechen/cab432assign2.git` (require username and password)
  - `$ cd cab432assign2`
  - `$ git fetch --all` (only needed if code has been updated)
  - `$ git reset --hard origin/master` (only needed if code has been updated)
  - Only In ***Tweet Stream section - Stream server:***
    - `$ cd cab432assign2/twitter/stream`
    - `$ npm install`
  - Only In ***Sentiment Analysis Processing section - Sentiment Analysis server:***
    - `$ cd cab432assign2/twitter/server`
    - `$ npm install`
  - Only In ***Application Client section - Client:***
    - `$ cd cab432assign2/twitter/app`
    - `$ npm install`
  - `$ sudo npm install pm2 -g`
  - `$ sudo pm2 start npm -- start`
  - `$ sudo pm2 startup` (allow it to start the npm when an instance starts up)
  - `$ sudo pm2 unstartup` (when you need to change some files, enter this first to shut down npm start)
3. Save the instances into an AMI respectively.

#### 4. Create a Launch Configuration as shown below.

- Select the AMI for the Sentiment Analysis server, in this case it's CAB432A2-Serverv2.

1. Choose AMI

2. Choose Instance Type

3. Configure details

4. Add Storage

5. Configure Security Group

6. Review

Cancel and Exit

Create Launch Configuration

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.


Quick Start

My AMIs

AWS Marketplace

Community AMIs


☐ Free tier only ⓘ

**Amazon Linux**  
Free tier eligible

**Amazon Linux AMI 2017.09.0 (HVM), SSD Volume Type** - ami-e689729e  
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.  
Root device type: ebs    Virtualization type: hvm

Select


64-bit

**Red Hat**  
Free tier eligible

**Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type** - ami-9fa343e7  
Red Hat Enterprise Linux version 7.4 (HVM), EBS General Purpose (SSD) Volume Type  
Root device type: ebs    Virtualization type: hvm

Select


64-bit

**SUSE Linux**  
Free tier eligible

**SUSE Linux Enterprise Server 12 SP3 (HVM), SSD Volume Type** - ami-8a887ff2  
SUSE Linux Enterprise Server 12 Service Pack 3 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.  
Root device type: ebs    Virtualization type: hvm

Select

64-bit

**Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-6e1a0117  
Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical

Select

64-bit

aws

Services

Resource Groups

🌟

🔔

yeechen @ 9300-3046-2974

Oregon

Support

1. Choose AMI

2. Choose Instance Type

3. Configure details

4. Add Storage

5. Configure Security Group

6. Review

Cancel and Exit


Create Launch Configuration

☐ 64-bit

Root device type


☐ EBS

☐ Instance store

**CAB432A2-Stream** - ami-3d9a5745  
Root device type: ebs    Virtualization type: hvm    Owner: 930030462974


Select

64-bit

**CAB432A2-Serverv2** - ami-3e9e5346  
Root device type: ebs    Virtualization type: hvm    Owner: 930030462974


Select

64-bit

**Basic Docker (Working)** - ami-5944a321  
Image for Prac 2  
Root device type: ebs    Virtualization type: hvm    Owner: 930030462974


Select

64-bit

**Basic Docker** - ami-6f47a017  
Image for Pratical 2  
Root device type: ebs    Virtualization type: hvm    Owner: 930030462974

Select

64-bit

**CAB432A2-Clientv2** - ami-82ab66fa  
Root device type: ebs    Virtualization type: hvm    Owner: 930030462974

Select

64-bit

Feedback

English (US)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

- Select T2 Micro.

aws

Services ▾ Resource Groups ▾

🔔

yeechen @ 9300-3046-2974 ▾

Oregon ▾

Support ▾

1. Choose AMI

2. Choose Instance Type

3. Configure details

4. Add Storage

5. Configure Security Group

6. Review

### Create Launch Configuration

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family ▾	Type ▾	vCPUs ⓘ ▾	Memory (GiB) ▾	Instance Storage (GB) ⓘ ▾	EBS-Optimized Available ⓘ ▾	Network Performance ⓘ ▾
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate

Cancel

Previous

Next: Configure details

Feedback

English (US)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

- Set the Configuration details, enable Cloudwatch monitoring.

1. Choose AMI

2. Choose Instance Type

3. Configure details

4. Add Storage

5. Configure Security Group

6. Review

### Create Launch Configuration

Name ⓘ

CAB432A2-LaunchConfiguration

Purchasing option ⓘ

☐ Request Spot Instances

IAM role ⓘ

Loading... ▾

Monitoring ⓘ

☒ Enable CloudWatch detailed monitoring

[Learn more](#)

▶ Advanced Details

- Proceed with default hard disk volume

**aws** Services ▾ Resource Groups ▾ ⭐

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

### Create Launch Configuration

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes.  
<https://docs.aws.amazon.com/console/ec2/launchinstance/storage> about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/sda1	snap-06b88b216a767d1b0	8	General Purpose (SSD) ▾	100 / 3000	N/A	<input checked="" type="checkbox"/>	No

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Skip to review](#) [Next: Configure Security Group](#)

[Feedback](#) [English \(US\)](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

- Allow port 22 (SSH), 80 (HTTP), and 3000 (Node.js) to be accessible.

**aws** Services ▾ Resource Groups ▾ ⭐

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

### Create Launch Configuration

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below.  
[Learn more](#) about Amazon EC2 security groups.

**Assign a security group:** ☐ Create a new security group ☒ Select an existing security group

Security Group ID	Name	VPC ID	Description	Actions
sg-c4c544be	cab432	vpc-ab977ecc	cab432	<a href="#">Copy to new</a>
sg-b28e1ccf	CAB432A2-AutoScaling-SecurityGroup	vpc-ab977ecc	Security Group for cab432asgn2 Auto Scaling	<a href="#">Copy to new</a>

Inbound rules for sg-b28e1ccf Selected security groups: sg-b28e1ccf.

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	8000	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
Custom TCP Rule	TCP	3000	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

[Cancel](#) [Previous](#) [Review](#)

[Feedback](#) [English \(US\)](#) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

- Finalise the Launch Configuration .

5. Create an Auto Scaling group as shown below.

- Fill up the details, start the group size with 1 instance. Set the subnet to 2 zones (us-west-2a and us-west-2b).

aws Services Resource Groups

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

### Create Auto Scaling Group

**Launch Configuration** CAB432A2-LaunchConfiguration

**Group name** CAB432A2-AutoScalingGroup

**Group size** Start with 1 instances

**Network** vpc-ab977ecc (172.31.0.0/16) | DEFAULT-VPC (default) [Create new VPC](#)

**Subnet** subnet-9d0317f9(172.31.16.0/20) | Default in us-west-2b  
subnet-4f012f39(172.31.32.0/20) | Default in us-west-2a [Create new subnet](#)

Each instance in this Auto Scaling group will be assigned a public IP address.

- Set the auto-scaling group to scale between 1 to 50 (or any really) .

• **Use scaling policies to adjust the capacity of this group**

Scale between 1 and 50 instances. These will be the minimum and maximum size of your group.

- Create an alarm which increases an instance when a Scaling group instance's Maximum CPU Utilisation  $\geq 40\%$  for 4 consecutive periods of 1 minutes (we changed it to 1 consecutive periods of 60 seconds later on).

Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.  
To edit an alarm, first choose whom to notify and then define when the notification should be sent.

☐ Send a notification to: No SNS topics found...

Whenever: Maximum of CPU Utilization

Is:  $\geq$  40 Percent

For at least: 4 consecutive period(s) of 1 Minute

Name of alarm: awsec2-CAB432A2-AutoScalingGroup-High-CPU-U

CPU Utilization Percent

40  
30  
20  
10  
0

10/18 06:00 10/18 08:00 10/18 10:00

CAB432A2-AutoScalingGroup



- Create Alarm

X

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

☐ Send a notification to:

No SNS topics found...

Whenever:

Maximum

of

CPU Utilization

Is:

<=

10

Percent

For at least:

4


consecutive period(s) of

1 Minute

Name of alarm:

awsec2-CAB432A2-AutoScalingGroup-Low-CPU-Ut

CPU Utilization Percent



CAB432A2-AutoScalingGroup

- aws

Services

Resource Groups

🔔

yeechen @ 9300-3046-2974

Oregon

Support

1. Configure Auto Scaling group details

2. Configure scaling policies

3. Configure Notifications

4. Configure Tags

5. Review

## Create Auto Scaling Group

Configure your Auto Scaling group to send notifications to a specified endpoint, such as an email address, whenever a specified event takes place, including: successful launch of an instance, failed instance launch, instance termination, and failed instance termination.

If you created a new topic, check your email for a confirmation message and click the included link to confirm your subscription. Notifications can only be sent to confirmed addresses.

Send a notification to:

Mark

use existing topic

✕

With these recipients:

yeechen.fong@connect.qut.edu.au

Whenever instances:

☒ launch

☒ terminate

☒ fail to launch

☒ fail to terminate

Add notification

- aws

Services

Resource Groups

yeichen @ 9300-3046-2974

Oregon

Support

1. Configure Auto Scaling group details

2. Configure scaling policies

3. Configure Notifications

4. Configure Tags

5. Review

### Create Auto Scaling Group

A tag consists of a case sensitive key-value pair that you can use to identify your group. For example, you could define a tag with Key = Environment and Value = Production. You can optionally choose to apply these tags to instances in the group when they launch. [Learn more.](#)

Key	Value	Tag New Instances
Environment	CAB432A2-AutoScalingGroup	<input checked="" type="checkbox"/>

Add tag

49 remaining

- Finalise auto-scaling group settings .

Services
Resource Groups

yeechen @ 9300-3046-2974
Oregon
Support

1. Configure Auto Scaling group details
2. Configure scaling policies
3. Configure Notifications
4. Configure Tags
5. Review

### Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create Auto Scaling group** to complete the creation of an Auto Scaling group.

Auto Scaling Group Details
Edit details

Group name CAB432A2-AutoScalingGroup  
Group size 1  
Minimum Group Size 1  
Maximum Group Size 20  
Subnet(s) subnet-9d0317f9, subnet-4f012f39  
Health Check Grace Period 300  
Detailed Monitoring No  
Instance Protection None

Scaling Policies
Edit scaling policies

Increase Group Size With alarm = awsec2-CAB432A2-AutoScalingGroup-High-CPU-Utilization; Add 1 instances and 300 seconds for instances to warm up  
Decrease Group Size With alarm = awsec2-CAB432A2-AutoScalingGroup-Low-CPU-Utilization; Remove 1 instances

Notifications
Edit notifications

Mark  
(yeechen.fong@connect.qut.edu.au) launch, terminate, fail to launch, fail to terminate

Cancel Previous **Create Auto Scaling group**

Feedback English (US)
© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



## 6. Create a Load Balancer as shown below.

- Select the Classic Load Balancer.

Select load balancer type

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers (new), and Classic Load Balancers. Choose the load balancer type that meets your needs. [Learn more about which load balancer is right for you](#)

Application Load Balancer	Network Load Balancer	Classic Load Balancer
<p>HTTP HTTPS</p> <p>Create</p> <p>Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing, TLS termination and visibility features targeted at application architectures, including microservices and containers.</p> <p><a href="#">Learn more &gt;</a></p>	<p>TCP</p> <p>Create</p> <p>Choose a Network Load Balancer when you need ultra-high performance and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second while maintaining ultra-low latencies.</p> <p><a href="#">Learn more &gt;</a></p>	<p>PREVIOUS GENERATION for HTTP, HTTPS, and TCP</p> <p>Create</p> <p>Choose a Classic Load Balancer when you have an existing application running in the EC2-Classical network.</p> <p><a href="#">Learn more &gt;</a></p>

[Cancel](#)

Feedback English (US) © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- Define the Load Balancer details, with the Load Balancer Protocol set to HTTP and port 3000, the Instance Protocol set to HTTP and port 3000.

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name: CAB432A2-LoadBalancer

Create LB Inside: My Default VPC (172.31.0.0/16) | DEFAULT-VPC

Create an internal load balancer: ☐ (what's this?)

Enable advanced VPC configuration: ☒

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate
HTTP	3000	HTTP	3000	N/A	N/A

- Select 2 subnet zones (us-west-2a and us-west-2b).

#### Select Subnets

You will need to select a Subnet for each Availability Zone where you wish traffic to be routed by your load balancer. If you have instances in only one Availability Zone, please select at least two Subnets in different Availability Zones to provide higher availability for your load balancer.

VPC vpc-ab977ecc (172.31.0.0/16) | DEFAULT-VPC

##### Available subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
<a href="#">+</a>	us-west-2c	subnet-49284f11	172.31.0.0/20	

##### Selected subnets

Actions	Availability Zone	Subnet ID	Subnet CIDR	Name
<a href="#">-</a>	us-west-2a	subnet-4f012f39	172.31.32.0/20	
<a href="#">-</a>	us-west-2b	subnet-9d0317f9	172.31.16.0/20	

[Cancel](#) [Next: Assign Security Groups](#)

[Feedback](#) [English \(US\)](#)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

- Allow port 22 (SSH), 80 (HTTP), and 3000 (Node.js) to be accessible.

[Services](#)
[Resource Groups](#)

yeechen @ 9300-3046-2974
Oregon
Support

1. Define Load Balancer 2. Assign Security Groups 3. Configure Security Settings 4. Configure Health Check 5. Add EC2 Instances 6. Add Tags 7. Review

#### Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security groups to assign to this load balancer. This can be changed at any time.

Assign a security group: ☐ Create a new security group  
☒ Select an existing security group

Filter VPC security groups

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-c4c544be	cab432	cab432	<a href="#">Copy to new</a>
<input checked="" type="checkbox"/> sg-b28e1ccf	CAB432A2-AutoScaling-SecurityGroup	Security Group for cab432asgn2 Auto Scaling	<a href="#">Copy to new</a>
<input type="checkbox"/> sg-72bfd80b	default	default VPC security group	<a href="#">Copy to new</a>
<input type="checkbox"/> sg-eb55fb96	Demo	launch-wizard-1 created 2017-10-04T13:21:41.397+10:00	<a href="#">Copy to new</a>

- The health check is set to ping at the instances via TCP at port 3000 with response timeout at 10 seconds, health check intervals at 60 seconds, unhealthy threshold at 2 consecutive health check failures, and healthy threshold at 5 consecutive health check successes.

Ping Protocol TCP

Ping Port 3000

#### Advanced Details

Response Timeout i 10 seconds

Interval i 60 seconds

Unhealthy threshold i 2

Healthy threshold i 5

- Skip adding EC2 instances.

Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC vpc-ab977ecc (172.31.0.0/16) | DEFAULT-VPC

<input type="checkbox"/>	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input type="checkbox"/>	i-0e0ab11216f78b49b		running	CAB432A2-AutoScaling-SecurityGroup	us-west-2b	subnet-9d0317f9	172.31.16.0/20

Availability Zone Distribution

☐ Enable Cross-Zone Load Balancing ⓘ

☒ Enable Connection Draining ⓘ 300 seconds

Cancel Previous Next: Add Tags

- Configure Load Balancer tags (optional).

Step 6: Add Tags

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value
Environment	CAB432A2-LoadBalancer-I

Create Tag

Cancel Previous Review and Create

- Finalise Load Balancer settings .

**Step 7: Review**  
Please review the load balancer details before continuing

**Define Load Balancer** [Edit load balancer definition](#)

Load Balancer name: CAB432A2-LoadBalancer  
Scheme: Internet-facing  
Port Configuration: 80 (HTTP) forwarding to 3000 (HTTP)

**Configure Health Check** [Edit health check](#)

Ping Target: HTTP:3000/  
Timeout: 10 seconds  
Interval: 60 seconds  
Unhealthy threshold: 2  
Healthy threshold: 5

**Add EC2 Instances** [Edit instances](#)

Cross-Zone Load Balancing: Disabled  
Connection Draining: Enabled, 300 seconds  
Instances:

**VPC Information** [Edit subnets](#)

[Cancel](#) [Previous](#) [Create](#)

## 7. Add the newly configured Load Balancer to the Auto-Scaling group.

**Auto Scaling Group: CAB432A2-AutoScalingGroup**

[Details](#) [Activity History](#) [Scaling Policies](#) [Instances](#) [Monitoring](#) [Notifications](#) [Tags](#) [Scheduled Actions](#) [Lifecycle Hooks](#)

[Cancel](#) [Save](#)

**Launch Configuration** CAB432A2-LaunchConfiguration

**Load Balancers** CAB432A2-LoadBalancer x

**Target Groups**

**Desired** 1  
**Min** 1  
**Max** 20

**Availability Zone(s)** us-west-2a, us-west-2b

**Subnet(s)**

- subnet-9d0317f9(172.31.16.0/20) | Default in us-west-2b x
- subnet-4f012f39(172.31.32.0/20) | Default in us-west-2a x

**Default Cooldown** 300