GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 4150-A Spring 2021
Lab5: Real-time Social Media Sentiment Analysis with AWS Kinesis, Lambda, ElasticSearch Service and DynamoDB

References:

[1] A. Bahga, V. Madisetti, "Cloud Computing Solutions Architect: A Hands-On Approach", ISBN: 978-0996025591

[2] https://aws.amazon.com/documentation/

[3] AWS Kinesis: https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html

[4] AWS Elasticsearch Service: https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/what-is-amazon-elasticsearch-service.html

[5] Tweepy: https://github.com/tweepy/tweepy

[6] Consuming Streaming Data from Twitter: https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data

[7] AWS Comprehend: https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html

[8] CloudFormation: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

Due Date:

The lab report will be **due on <u>April 19, 2021 at 11:59 PM</u>**.

In this laboratory, you will walk through a real-time social media dataflow to analyze sentiment from the text. The typical use of sentiment analysis is to examine, from a text, how people feel about different products, topics, personalities, or issues.

The application uses the following:

1. A Python script that connects to the Twitter API to fetch tweets in real-time.
2. The Python script that runs on a EC2.
3. The tweets coming from Twitter API are being published to AWS Kinesis Data Stream.
4. A Lambda Function that consumes the tweets data from the Kinesis Data Stream.
5. A Lambda function for processing the tweets.
6. DynamoDB for storing analyzed tweets.
7. ElasticSearch Kibana for creating visualizations.

Learning outcomes:

1. Setting up an application to listen in real-time Twitter posts based on a particular topic.
2. Setting up data streams, such as kinesis, to handle high-volume and high-velocity data.
3. Processing data using a serverless architecture.
4. Storing processed data in a DynamoDB table.
5. Creating visualizations using Kibana.

Below is a typical solution architecture for real-time analysis of social media environments. For this architecture we are looking at the Twitter posts to translate and extract insights from those tweets.

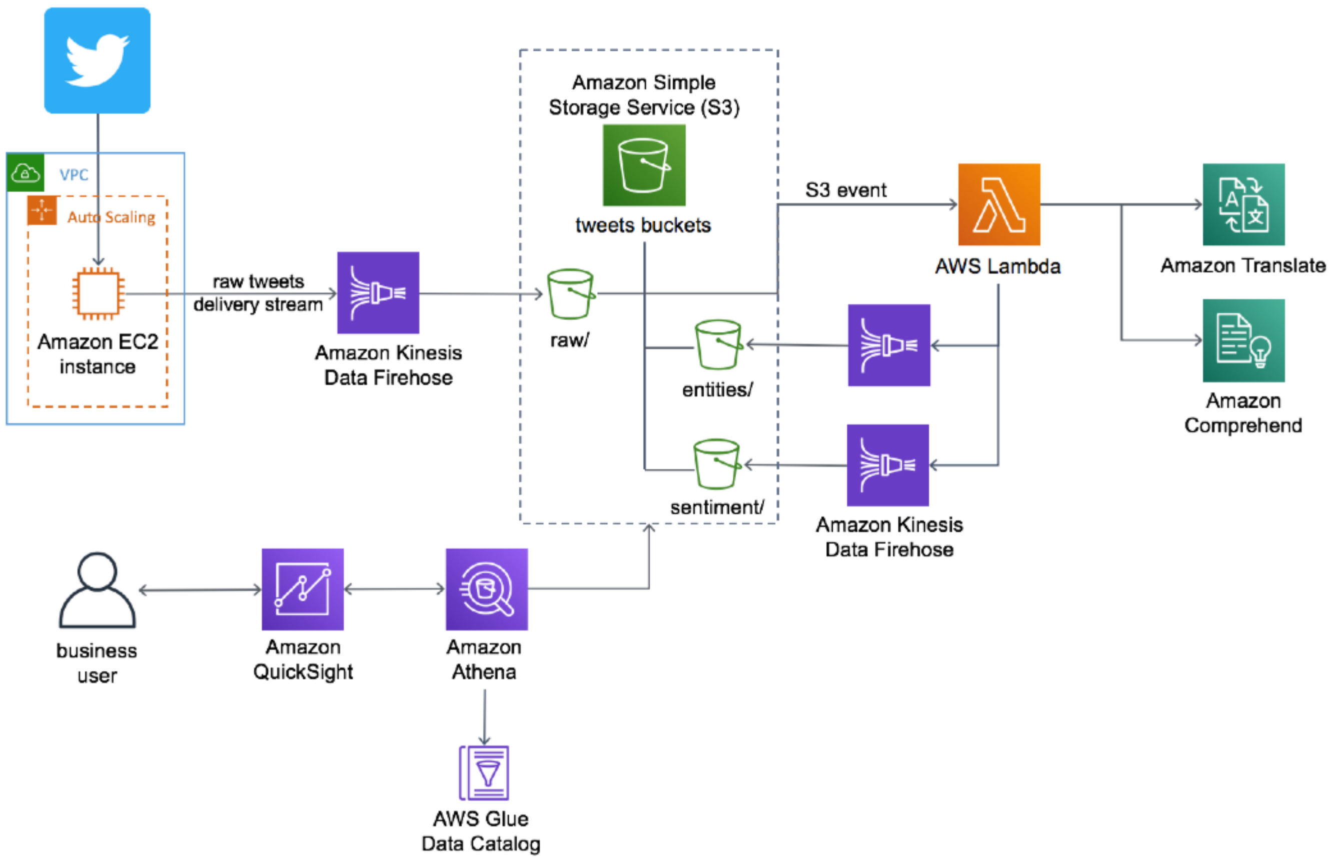

Figure 1: An overview of a typical production social media analytics architecture. This solution monitors and ingests specified tweets using stream processing and leverages a serverless architecture and machine learning services to translate and extract insights from those tweets.

For our lab, we will implement a miniature version of this architecture and explore data streaming concepts over a serverless platform. Below is the architecture for this lab.

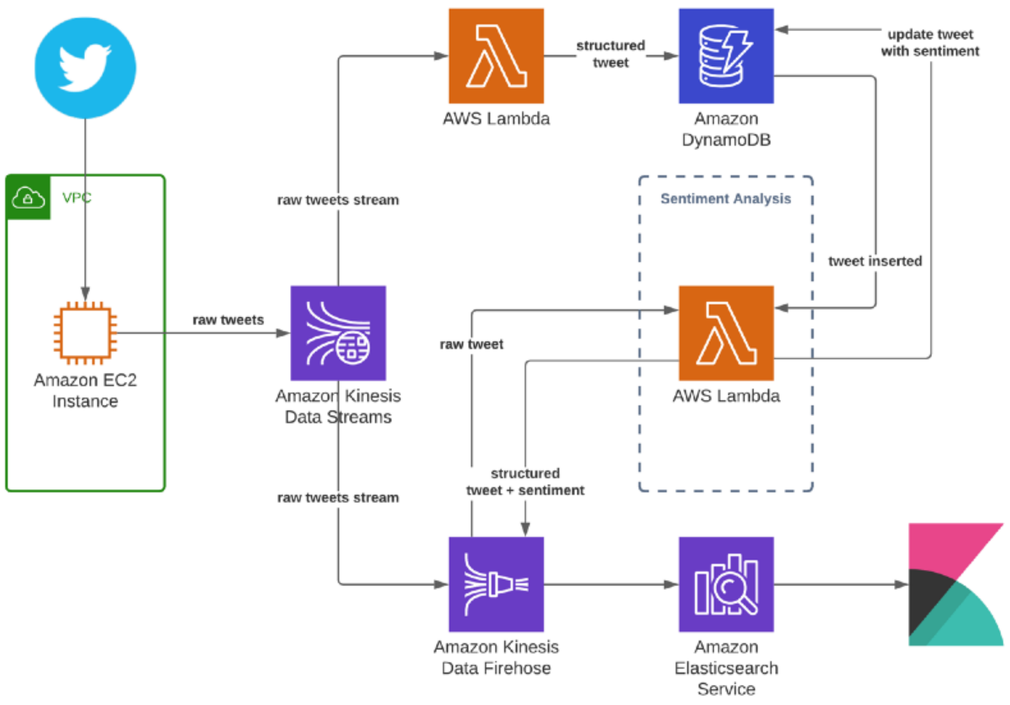

Figure 2: An overview of the Sentiment Analysis architecture. The tweets coming from the Twitter API are in injected to the Kinesis Data Streams and deliver to other services, such as DynamoDB and Elasticsearch using Kinesis Data Firehose. Each service use the same Lambda Function to retrieve the sentiment from the tweets.

The instructions to create the minimal architecture are described below. Please, follow the instructions step-by-step; do not jump to another step without completing the current one you are working on. If you fail to follow these steps, you will create issues in your architecture that might not be easy to track.

Instructions:

1. Get familiar with the code

- Inspect the code provided to you and make sure you understand how everything works.

2. Create an application in the Twitter Developer Console and obtain credentials

- Go to **Developer Portal** (https://developer.twitter.com) and sign up. Once you sign up, you will be redirected to the Developer Portal Page.
- Click on **Projects and Apps** on the left panel and click **Overview**. Scroll down and click on **Create App**.

- Enter the name of the application.
- Once you create the App, click **Overview** on the left panel and select the application you created.
- Obtain the credentials needed to access the API. Under **Keys and tokens**, copy the **API key, API secret key, Access Token,** and **Access token secret**.

3. Create an S3 Bucket to store the code for the EC2 and Lambda Functions

- Make sure to **allow all public access** to the bucket.
- For the name of the bucket, use **code-{LASTNAME}-{YEAR}-ece{COURSENUMBER}** (e.g., code-corporan-2021-ece4150).
- Create two folders in this bucket. One called '**ec2**', and another one called '**lambda**'.
- Add a policy in the bucket to enable public access to the files uploaded as we did in the previous laboratories.
- Upload the **listener.zip** file on '**ec2**', and the **kinesis-to-dynamo.zip** and **sentiment-analysis.zip** files on '**lambda**'.

4. Create an S3 Bucket to store failed deliveries

- **DO NOT ALLOW PUBLIC ACCESS** the bucket.
- For the name of the bucket, use **firehose-{LASTNAME}-{YEAR}-ece{COURSENUMBER}** (e.g., firehose-corporan-2021-ece4150).

5. Store credentials in AWS Secrets Manager (2 points)

- Store each key/token in **separate secret instances** by going to the AWS Secret Manager console and **storing a new secret**. Each time you save a new secret, click **Store a new secret**.
- Use the **Other type of secrets** option and store the key/token in a **plaintext** in the first line of the text field. **DO NOT INCLUDE ADDITIONAL LINES**.
- Save the the secret as **the name of the key**. These are the keys:
  - **TwitterAPIKey**
  - **TwitterAPISecretKey**
  - **TwitterAccessToken**
  - **TwitterAccessSecret**

6. Create a Kinesis Data Stream (2 points)

- Create a new Kinesis Data Stream named **twitter-stream** to capture the tweets coming from the listener in the EC2 instance.
- For the number of open shards, enter **only one** (1).

7. Create DynamoDB table to store the tweets and (2 points)

- Create a DynamoDB table named **Tweet** to store the structured tweet from the Lambda Function.
- For the table, use **id_str**, type *String*, as partition key, and **timestamp**, type *Number*, as sort key.

8. Create IAM role and policy for the Lambda Function used to store data (2 points)

- Create a new IAM policy named **kinesis-to-dynamo-access-policy** and define permissions for the Lambda Function to **ONLY** access **Kinesis Data Stream, DynamoDB** and **CloudWatch Logs** services created above as described below.
  - ◉ The **ONLY** actions the Lambda Function can perform on the **Kinesis** are **GetRecords, GetShardIterator, DescribeStream, ListShards, and ListStreams**.
  - ◉ The **ONLY** action the Lambda Function can perform on the **DynamoDB** is **PutItem**.
  - ◉ The **ONLY** actions the Lambda Function can perform on the **CloudWatch Logs** are **CreateLogGroup, CreateLogStream,** and **PutLogEvents**.
    - In **Resources**, under **log-group**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log group to be created, and under **Log group name**, add an asterisk (*).
    - In **Resources**, under **log-stream**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log stream to be created and store the logs, under **Log group name**, add **/aws/lambda/***, and under **Log stream name**, add an asterisk (*).
- Create a new IAM role named **kinesis-to-dynamo-access-role** for a **Lambda** use case and attach the policy created above (**kinesis-to-dynamo-access-policy**).

****Failure to comply with the instructions will result in the assignment of zero on this step****

9. Create a Lambda function to store data in DynamoDB (2 points)

- Create a new **Lambda Function** named **kinesis-to-dynamo**.
- For runtime, use **Python 3.8** and attach the **role** created in the previous step (**kinesis-to-dynamo-access-role**).
- In **Configuration**, under **General configuration**, increase the memory to **2048 MB** and timeout to **five (5) minutes**.
- In **Configuration**, under **Environment variables**, add a new variable called **TWEET_TABLE** and the value of the that variable is the name of the DynamoDB table created above (Tweet).
- Upload the code used for this function using the link to the S3 bucket.
- Add a new trigger from **Kinesis** using the one created in the above (**twitter-stream**). For the **batch size** enter ten (10) and for **batch window** enter sixty (60).

10. Create IAM role and policy for the Lambda Function used for Sentiments (2 points)

- Create a new IAM policy named **sentiments-access-policy** and define permissions for the Lambda Function to **ONLY** access **DynamoDB** and **CloudWatch Logs** services created above as described below.
  - ◉ The **ONLY** actions the Lambda Function can perform on the **DynamoDB** are **UpdateItem, GetRecords, GetShardIterator, DescribeStream,** and **ListStreams**.
    - In **Resources**, under **stream**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your table is located, under **Table name**, enter the table name created above, and under **Stream label**, add an asterisk (*).
    - In **Resources**, under **table**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your table is located, and under **Table name**, enter the table name created above.
  - ◉ The **ONLY** actions the Lambda Function can perform on the **CloudWatch Logs** are **CreateLogGroup, CreateLogStream,** and **PutLogEvents**.
    - In **Resources**, under **log-group**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log group to be created, and under **Log group name**, add an asterisk (*).
    - In **Resources**, under **log-stream**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log stream to be created and store the logs, under **Log group name**, add **/aws/lambda/***, and under **Log stream name**, add an asterisk (*).
- Create a new IAM role named **sentiments-access-role** for a **Lambda** use case and attach the policy created above (**sentiments-access-policy**).

****Failure to comply with the instructions will result in the assignment of zero on this step****

11. Create a Lambda function to analyze sentiments (2 points)

- Create a new **Lambda Function** and named **sentiments-analysis**.
- For runtime, use **Python 3.8** and attach the **role** created in the previous step (**sentiments-access-role**).
- In **Configuration**, under **General configuration**, increase the memory to **2048 MB** and timeout to **five (5) minutes**.
- In **Configuration**, under **Environment variables**, add a new variable called **TWEET_TABLE** and the value of the that variable is the name of the DynamoDB table created above (Tweet).
- Upload the code used for this function using the link to the S3 bucket.

12. Enable DynamoDB Stream and create an event trigger (2 points)

- On your DynamoDB table, enable **DynamoDB Stream** under **Exports and streams**.

- Select **New image** as the version of the time that you would like to push to the DynamoDB stream.
- Create a new trigger to invoke the Lambda Function created for sentiment analysis **sentiments-analysis**).
- For the **batch size**, select one (1).

13.Create an Elasticsearch Domain (2 points)

- Create a new Elasticsearch Domain named **twitter-search** for **Development and testing**. Make sure you are using the latest version (7.9).
- For the **instance type**, under **Data nodes**, select the **t2.small.elasticsearch** instance with one (1) node.
- Increase the EBS storage size to **30GiB** under **Data node storage**
- **Allow public access** to the elasticsearch instance under **Network configuration**.
- Under **Access policy**, add a **Custom access policy** to allow access to elasticsearch using your own IPv4 address instead of using an IAM role.
  - ◉ *This method is better for security reasons since you only will have access using that IP address. However, if you change the network or move to another place to keep working, you have to remove the old IP address and add a new one. Go to https://whatismyipaddress.com to check your current IPv4 address.*
- Under Encryption, uncheck **Require HTTPS for all traffic to the domain** to allow unsecured traffic.

14.Create IAM role and policy for Kinesis Firehose (2 points)

- Create a new IAM policy named **firehose-access-policy** and define permissions for Kinesis Firehose to **ONLY** access **S3, Lambda, Elasticsearch, Kinesis Data Stream** and **CloudWatch Logs** services created above as described below.
  - ◉ The **ONLY** actions the Kinesis Firehose can perform on the **S3** are **AbortMultipartUpload, GetBucketLocation, GetObject, ListBucket, ListBucketMultipartUploads,** and **PutObject**.
    - In **Resources**, under **bucket**, click **Add ARN**. Under **Bucket name**, insert the name of the bucket created above (for failed deliveries).
    - In **Resources**, under **object**, click **Add ARN**. Under **Bucket name**, insert the name of the bucket created above (for failed deliveries), and under **Object name**, add an asterisk (*).
  - ◉ The **ONLY** actions the Kinesis Firehose can perform on the **Lambda** are **GetFunctionConfiguration** and **InvokeFunction**.
    - In Resources, under function, click Add ARN. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your lambda function created above is located, and under **Function name**, add **sentiments-analysis:$LATEST**—*This will ensure that firehose use the latest version of the lambda function.*

- ◉ The **ONLY** actions the Kinesis Firehose can perform on the **Elasticsearch** are **DescribeElasticsearchDomains, DescribeElasticsearchDomain, DescribeElasticsearchDomainConfig, ESHttpGet, ESHttpPost,** and **ESHttpPut**.
  - In **Resources**, under **domain**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your elasticsearch is located, and under **Domain name**, add the name of the domain.
    - (e.g., arn:aws:es:{REGION}:{ACCOUNT_ID}:domain/twitter-search)
  - In **Resources**, under **domain**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your elasticsearch is located, and under **Domain name**, add the name of the domain, a forward slash, and an asterisk (*)
    - (e.g., arn:aws:es::{REGION}:{ACCOUNT_ID}:domain/twitter-search/*)
- ◉ The **ONLY** actions the Kinesis Firehose can perform on the **Kinesis** are **GetRecords, GetShardIterator, DescribeStream,** and **ListShards**.
- ◉ The **ONLY** actions the Lambda Function can perform on the **CloudWatch Logs** are **CreateLogGroup, CreateLogStream,** and **PutLogEvents**.
  - In **Resources**, under **log-group**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log group to be created, and under **Log group name**, add an asterisk (*).
  - In **Resources**, under **log-stream**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where you want your log stream to be created and store the logs, under **Log group name**, add **/aws/kinesisfirehose/***, and under **Log stream name**, add an asterisk (*).

- Create a new IAM role named **firehose-access-role** for a **Kinesis Firehose** use case and attach the policy created above (**firehose-access-policy**).

****Failure to comply with the instructions will result in the assignment of zero on this step****

15. Create a Kinesis Data Firehose (2 points)

- Create a new Kinesis Data Firehose named **twitter-firehose** to capture the tweets coming from the Kinesis Data Stream.
- Choose **Kinesis Data Stream** as a **Source**.
- **Enable Data transformation** under **Transform source records with AWS Lambda**.
- Use the lambda created in the previous step (**sentiments-analysis**).
- Select **Amazon Elasticsearch** as the destination and select the elasticsearch domain created in the previous steps.
- Under **Index**, enter **twittersentiment**.
- Decrease the **Retry duration** to sixty (60).
- Under **S3 backup**, select the S3 bucket created above for failed deliveries and **Failed records only** under **Backup mode**.

- Decrease the **buffer size** to one (1) MiB and the **buffer interval** to sixty (60) under **Elasticsearch buffer condition**s.
- Attach the IAM role created above under **Permissions**.

16. Create IAM role and policy for the EC2 Instance (2 points)

- Create a new IAM policy named **ec2-to-kinesis-access-policy** and define a permission for the EC2 to **ONLY** access **Kinesis, Elasticsearch,** and **Secret Manager** services created above as described below.
  - ◉ The **ONLY** action the EC2 can perform on the **Kinesis** is **PutRecord**.
  - ◉ The **ONLY** action the EC2 can perform on the **Secret Manager** is **GetSecretValue**.
    - In **Resources**, under **secret**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your secrets are located, and under **Secret id**, add the first few letters of the name are similar to all the keys (e.g., **Twitter**).
      - (e.g., arn:aws:secretsmanager:{REGION}:{ACCOUNT_ID}:secret:Twitter*)
    - In **Resources**, under **domain**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your elasticsearch is located, and under **Domain name**, add the name of the domain, a forward slash, and an asterisk (*)
  - ◉ The **ONLY** actions the EC2 can perform on the **Elasticsearch** are **ESHttpGet, ESHttpPut,** and **ESHttpHead**.
    - In **Resources**, under **domain**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your elasticsearch is located, and under **Domain name**, add the name of the domain.
      - (e.g., arn:aws:es:{REGION}:{ACCOUNT_ID}:domain/twitter-search)
    - In **Resources**, under **domain**, click **Add ARN**. DO NOT CHANGE THE ACCOUNT ID. Under **Region**, insert the region where your elasticsearch is located, and under **Domain name**, add the name of the domain, a forward slash, and an asterisk (*)
      - (e.g., arn:aws:es::{REGION}:{ACCOUNT_ID}:domain/twitter-search/*)
- Create a new IAM role named **ec2-to-kinesis-access-role** for an **EC2** use case and attach the policy created above (**ec2-access-policy**).

****Failure to comply with the instructions will result in the assignment of zero on this step****

17. Create EC2 instance to host the twitter listener (2 points)

- Create a new EC2 instance with the free-tier **Ubuntu Server 20.04 LTS (HVM) SSD Volume Type** image. For the instance type, choose the free tier **t2.micro**.
- When you click **Review and Launch**, do the following steps:
  - ◉ Under **Security Groups**, click **Edit security groups** on the right. Rename the security group name and description; name the security group **Lab5-access-SG**, and add **"SG for Lab5 to**

allow access to the EC2" for the description. Click **Review and Launch** to save and go back.

  - Under **Instance Details**, click **Edit instance details** on the right. Under the **IAM role**, attached the role created above. Click **Review and Launch** to save and go back.

- Make sure to add your key-pair to access your instance from your computer. You can also access the instance through the EC2 Console.

18. Configure EC2 and Create Elasticsearch index (2 points)

- Enter to the EC2 using SSH or login through the console.
- Update your EC2 instance by running **`sudo apt update`**.
- Install Python3 and pip3 and Python3 by running **`sudo apt install python3-pip`**.
- Download the **listener.zip** file from the S3 bucket using the **wget** or **curl** command
- Install the unzip library to uncompress the file by running **`sudo apt install unzip`**.
- Uncompress the zip file by running **`unzip -q listener.zip`**
- Install the dependencies for the listener by running **`pip3 install -r requirements.txt`**.
- Get the **elasticsearch domain endpoint** (without the https:// part) from the previously created elasticsearch domain by going to AWS Elasticsearch console.
- The name of **elasticsearch index** that you will create is **twittersentiment**.
- Create a python file called **config.py** and add the following four lines of code:

```
ES_URL='search-{DOMAIN_NAME}.us-east-1.es.amazonaws.com'
ES_INDEX='twittersentiment'
CF = False
KINESIS_STREAM = 'twitter-stream'
```

- Create the **twittersentiment** index by running the **es_index.py** script. The output response should look like this:

```
ES URL: search-{DOMAIN_NAME}.us-east-1.es.amazonaws.com
ES INDEX: twittersentiment
index twittersentiment does not exist
creating index twittersentiment...
Index Created
```

19. Run Twitter listener from EC2 and inspect your services from EC2 (2 points)

- Run the **`listener.py`** script and check for the twitter output in the terminal.
- Check all your services, such DynamoDB, Lambda Functions, Kinesis, and Elasticsearch.
- You should apply all your knowledge from your previous laboratories to troubleshoot and identify failures.

20. Create an Index pattern in Kibana to Visualize the data from Elasticsearch

- From the **Elasticsearch domain**, copy the Kibana URL and paste it on your web browser.
- Click **Explore on my own** and click **Use Elasticsearch data** on the top right.
- Close the helper window that appears in the right size and click **Create index pattern**.
- Enter the **twittersentiment** index in the box (it should appear below) and click **Next step**.
- Under **Time field**, select **timestamp**, then **Create index pattern**.

21. Create a map visualization in Kibana

- From the menu in the top left, select **Discover** to the data coming into your elasticsearch domain.
- On the top right, you can adjust the period of your data.
- Under **Available fields**, click the location field and then **visualize** to observe the location where the tweets are coming from.
- In the right panel, click **options** under the index name. Under **Base layer settings**, enable the **WMS map server**. Add the following configurations and then click **Update**:
    - WMS URL: https://basemap.nationalmap.gov/arcgis/services/USGSImageryTopo/MapServer/WMSServer
    - WMS Layer: 0
    - WMS Version: 1.3.0
- Save the visualization under the name **Map**.

22. Create a new dashboard and add the map in Kibana

- From the menu in the top left, select **Dashboard** and click **Create new dashboard**.
- From the toolbar in the top, click **Add** and insert the map created in the previous step and close the helper window.
- Save the dashboard by clicking Save in the toolbar and name it **Dashboard**.

23. Create a pie chart to visualize the sentiment of tweets in Kibana

- From the menu in the top left, select **Visualize** *(if you still see the map, click visualize from the path in the top left corner closed to the menu).*
- Click **Create visualization** and select a pie chart and the index name.
- Under **Buckets**, click **Add** and select **Split slices**.
- Under **Aggregation**, select **Range** and add the following ranges and click **Update**:
    - Greater than or equal to (≥) -100 and less than (<) 0
    - Greater than or equal to (≤) 0 and less than (<) 1

    - Greater than or equal to (≤) 1 and less than (<) 100
- Save the visualization under the name **Pie**.

24. Update the dashboard with the new pie chart in Kibana

- From the menu in the top left, select **Dashboard**.
- From the toolbar in the top, click **Edit** and then **Add** insert the pie chart created in the previous step and close the helper window.
- Save the dashboard by clicking **Save**.

25. Create a line chart to visualize the average sentiment of tweets in Kibana

- From the menu in the top left, select **Visualize** *(if you still see the pie chart, click visualize from the path in the top left corner closed to the menu)*.
- Click **Create visualization** and select a line chart and the index name.
- Under Metrics, click **Y-axis**, and under **Aggregation**, select **Average**.
- Under **Buckets**, click **Add** and select **X-axis**.
- Under **Aggregation**, select **Date Histogram**.
- Under the index name, click **Metrics and axes**. Under **Line mode** select **smoothed**.
- Click **Update**.
- Save the visualization under the name **Line**.

26. Update the dashboard with the new line chart in Kibana

- From the menu in the top left, select **Dashboard**.
- From the toolbar in the top, click **Edit** and then **Add** insert the line chart created in the previous step and close the helper window.
- Save the dashboard by clicking **Save**.

27. Create a custom table of results in Kibana

- From the menu in the top left, select **Discover**.
- From the **Available fields**, and select the following fields: **timestamp, tweet_text, and tweet_user_id**.
- Save the table under the name **Table**.

28. Update the dashboard with the custom table in Kibana

- From the menu in the top left, select **Dashboard**.

- From the toolbar in the top, click **Edit** and then **Add** insert the table created in the previous step and close the helper window.
- Save the dashboard by clicking **Save**.

Challenges

For each of the challenges that you are about to do, provide a detailed description of your design in a step-by-step basis. For example, if you are asked to implement X in the current architecture, list the steps that you have to perform to implement X (e.i. Remove Z from the Y, Created a new function C that computes S). Also, provide also visual diagrams or pseudo code for each of the challenges.

1. Update the Lambda Function for Sentiment Analysis to use AWS Comprehend (5 points)

- Modify the Lambda Function to extract the sentiment from the tweets using a AWS Comprehend instead of the current function. A modify architectural diagram is shown in figure 3.

2. Cost analysis of your application (5 points)

- How much would it cost, per month, to host this minimal architecture in AWS? Describe the cost per service and grand total per month. (Hint: You can make some assumptions about the rate at which the data flows through the lambda functions). If you have any questions about this topic, come to the office hours.

3. Implement a web application using Flask or Django to display processed data retrieved from DynamoDB and develop your own visualizations (15 points)

- Create a web application using Flask or Django with any visualization method, such as D3 or Highchart.

4. Create a CloudFormation stack for the architecture implemented (15 points)

- (Ask TAs for CloudFormation Template) Add the missing pieces needed in the template to create the CloudFormation stack for the minimal architecture. This stack allows you to deploy and disrupt the architecture anytime. This CloudFormation template starts from **Step #5** in the instruction set.

5. Write a one page literature review about Serverless Computing (30 points)

- Write a one-page minimum review advantage and drawback of serverless and Function-as-a-Service (FaaS). Your review needs to be supported by at least three (3) peer review papers or journals.
- State, in detail, the possible ways the serverless model can be improved (be specific). Support your conclusions.

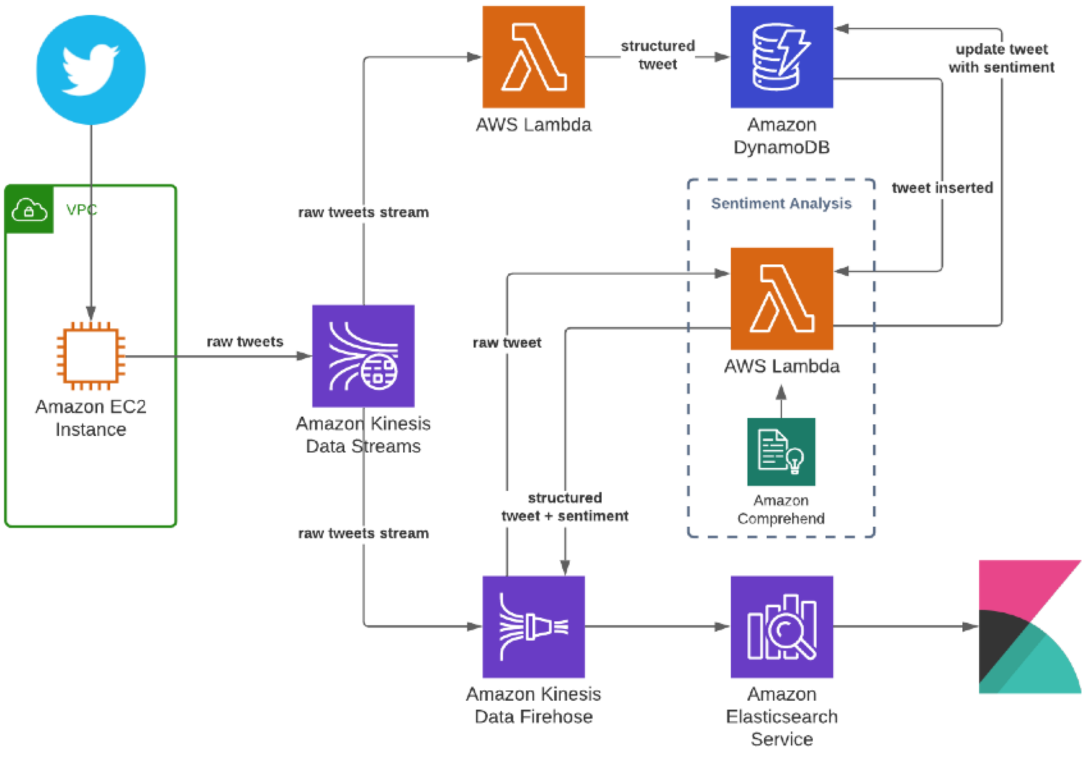

Figure 3: A modify version of the initial architectural diagram using AWS Comprehend

Deliverables

Submit all the codes you have modified and including all the challenges describe above. Once your submission is posted on Canvas (before April 19 at 11:59 PM), schedule an 30 minutes interview review with the teacher assistants to assess and grade your submission. The schedule will be open from **April 14th until April 26th**.