CSCI 5410

Assignment 4 – Part C

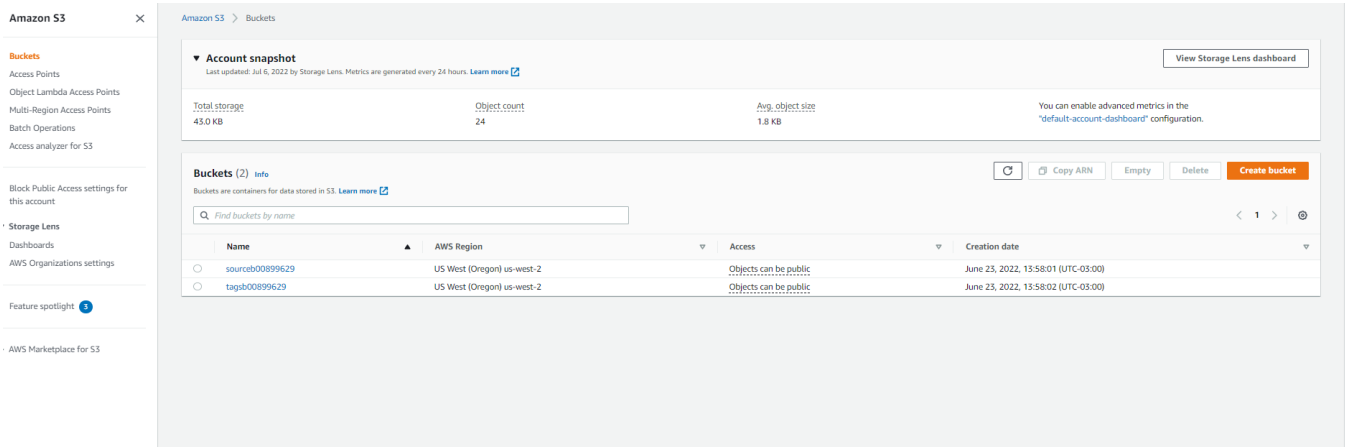
Name: Benny Daniel Tharigopala

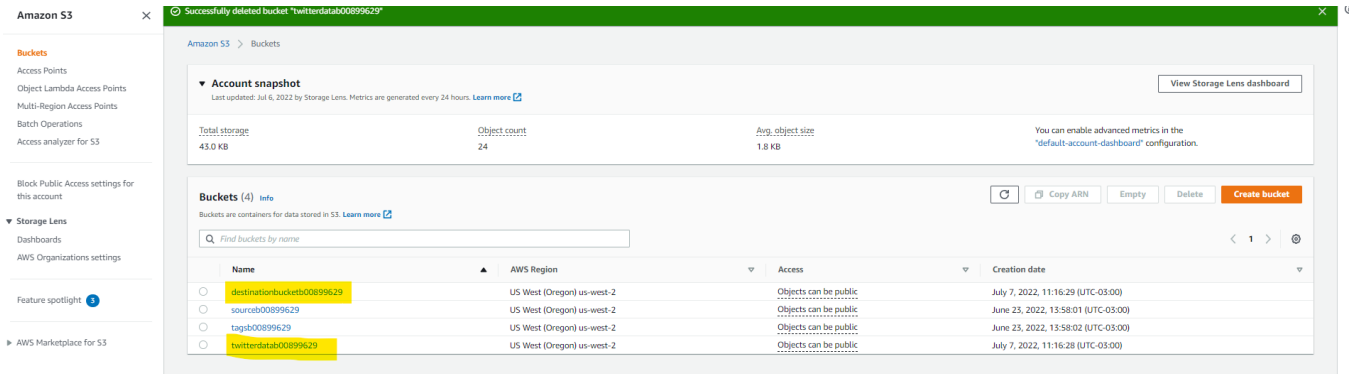
Banner ID: B00899629

GitLab URL: https://git.cs.dal.ca/benny/csci5410_B00899629_Benny_Tharigopala

Event-driven Serverless application with AWS Lambda & Comprehend

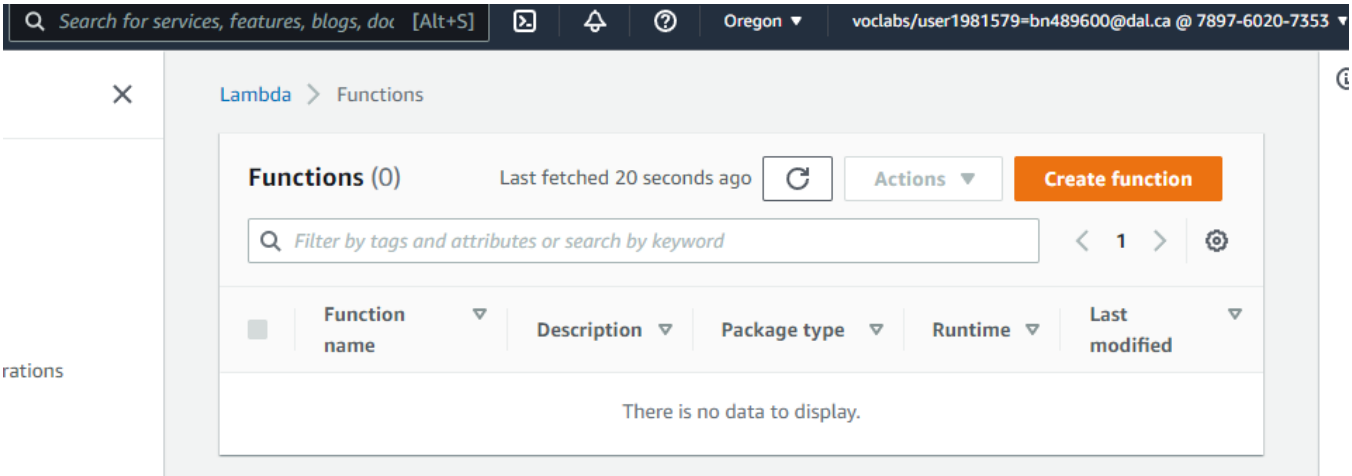
S3Buckets Operations:

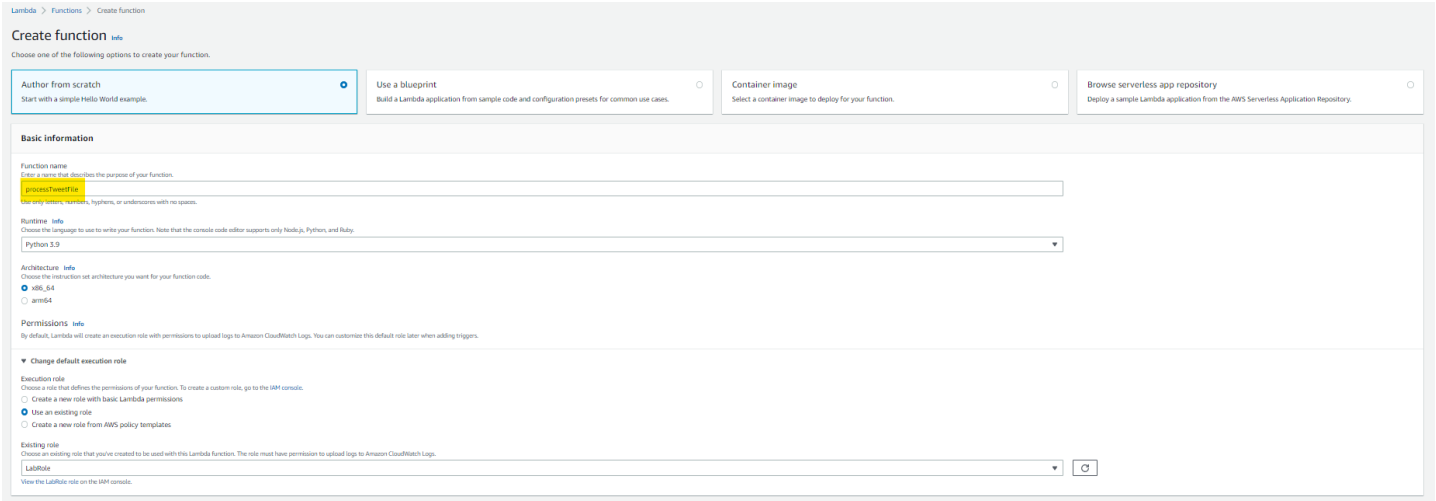


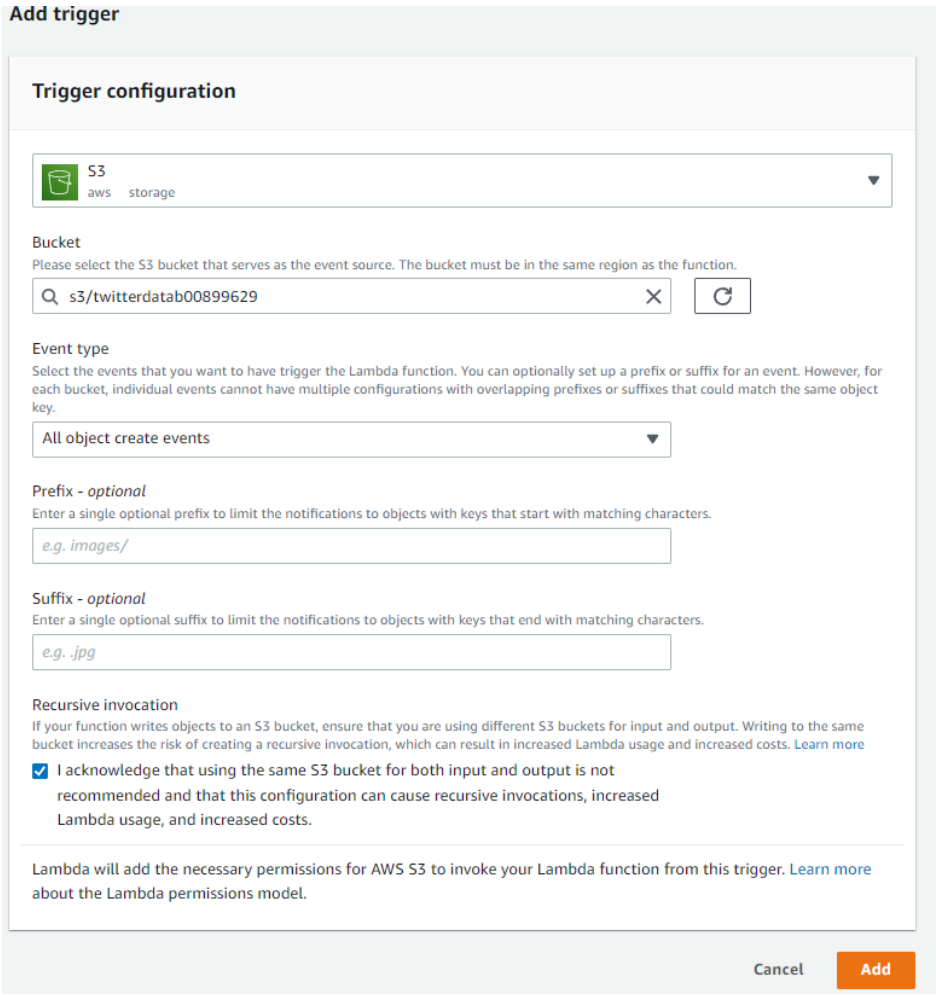


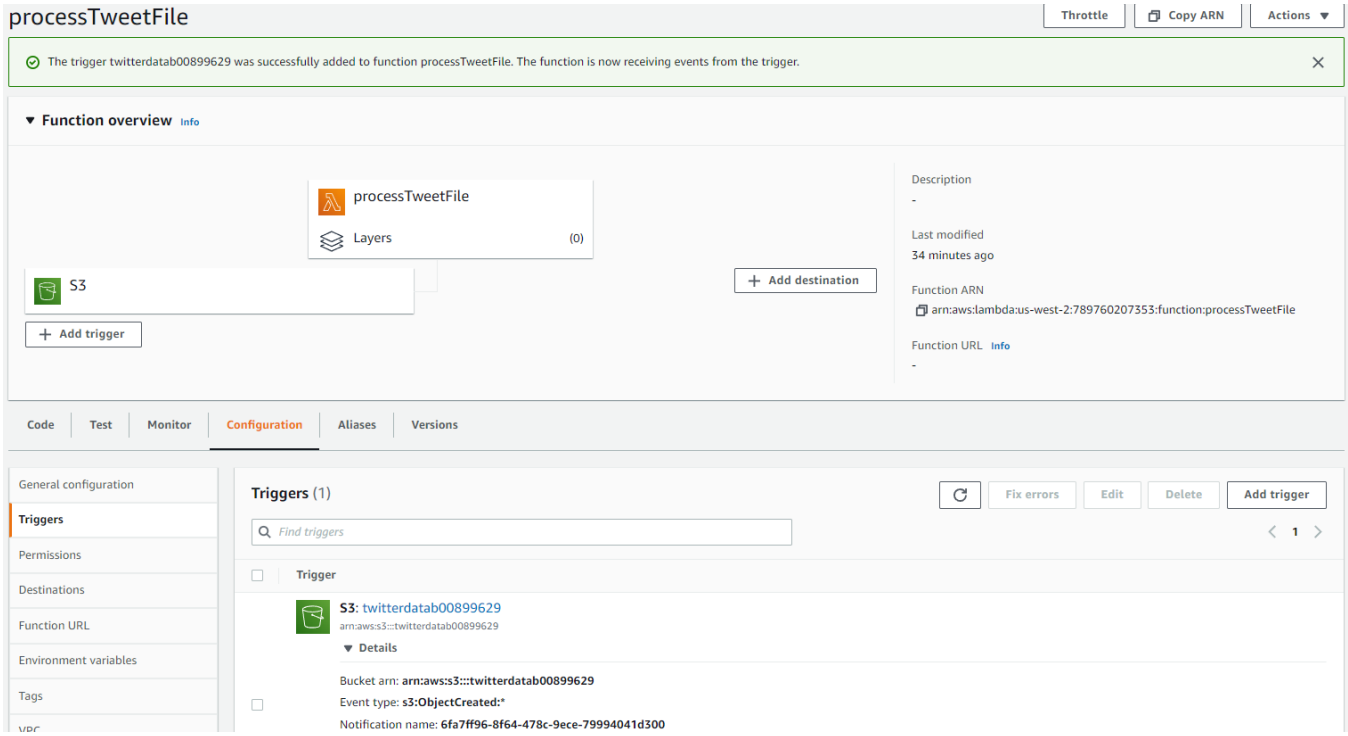
Lambda Functions:

Function – “extractFeatures”:



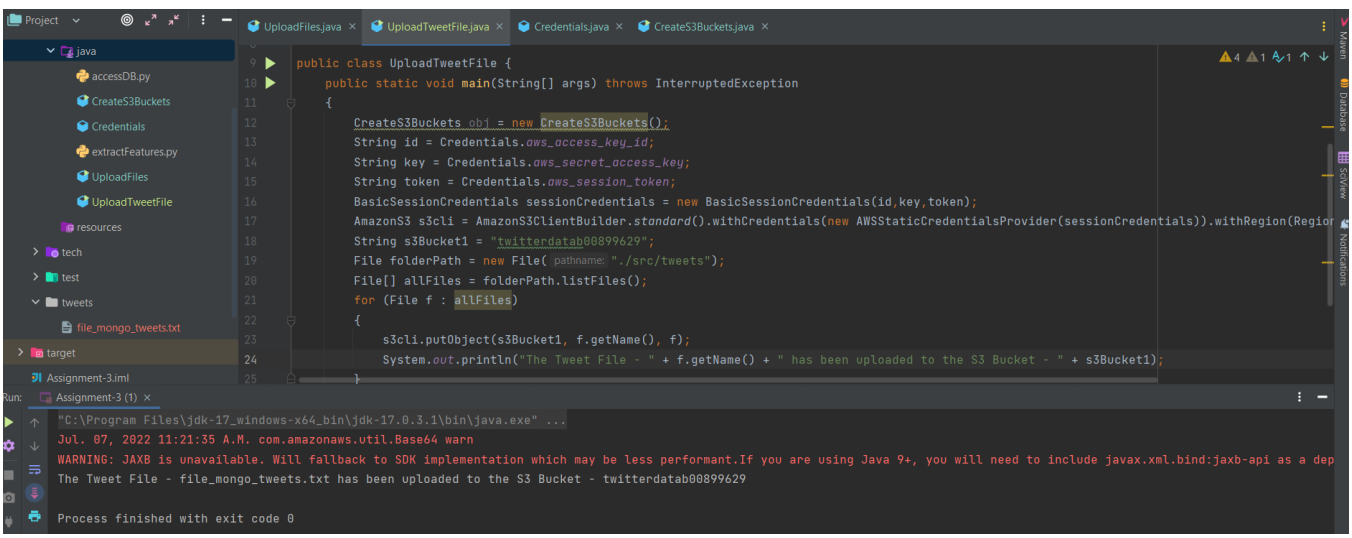

Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.
Basic information
Function name
Enter a name that describes the purpose of your function.
processTweetFile
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole
View the LabRole role on the IAM console.


Add trigger
Trigger configuration
S3
aws storage
Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/twitterdatab00899629
Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg
Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn more
I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.
Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.
Cancel
Add

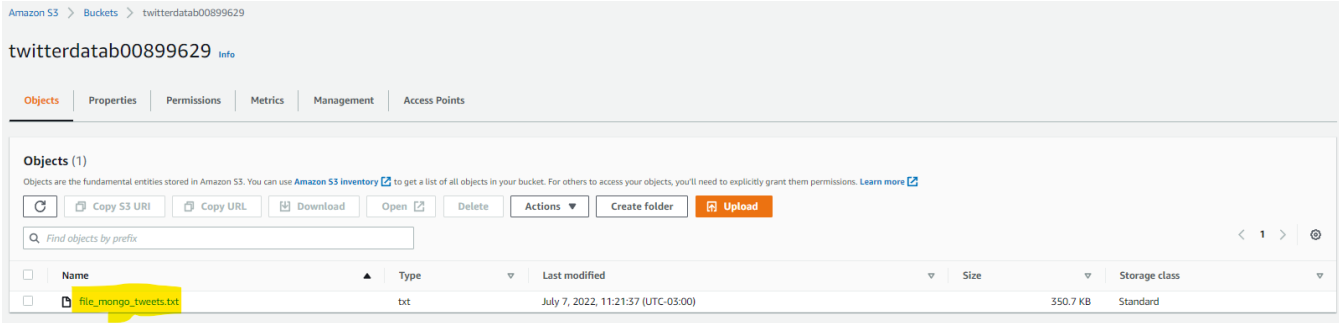


Snips of Operations in Chronological Order:

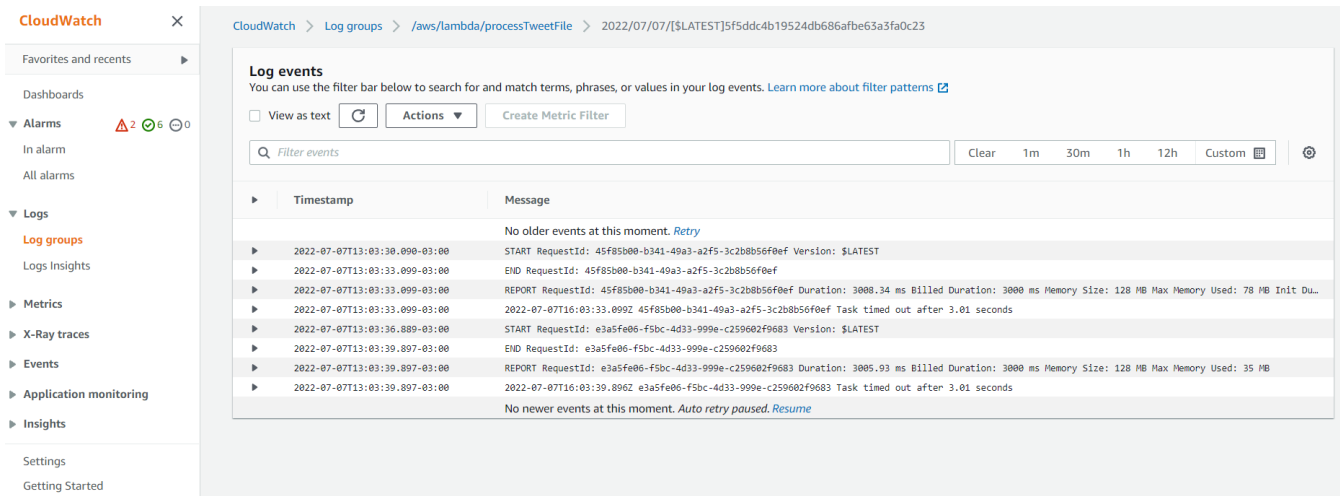
1. Upload Tweet File to the S3 Bucket



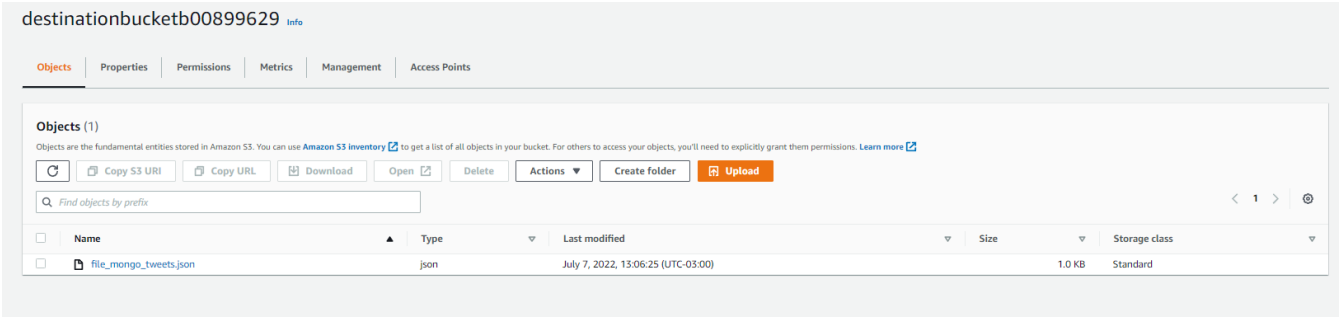
2. Tweet File in the Source S3 Bucket



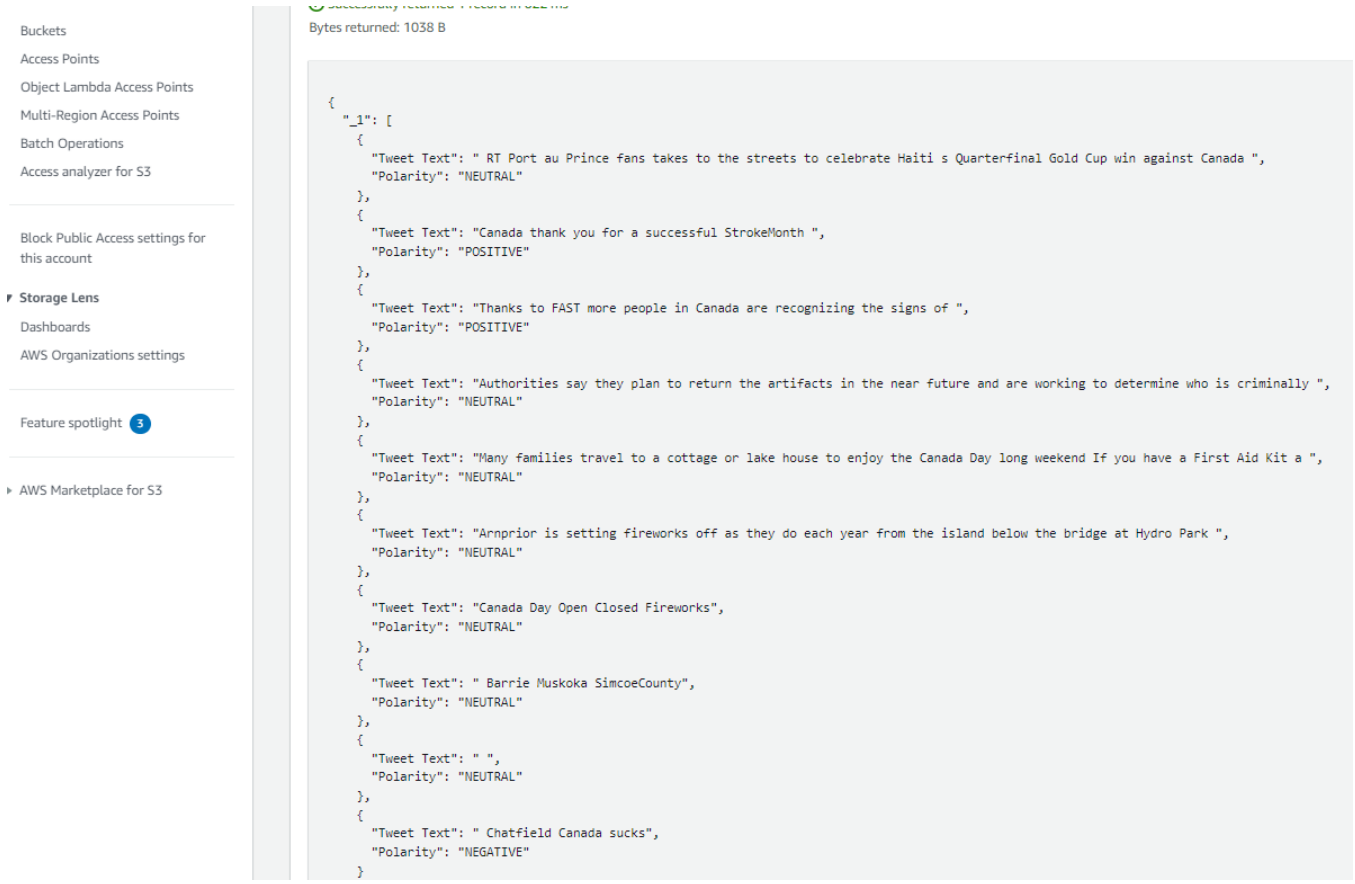
3. CloudWatch Log for the Tweet File Upload



4. Processed Tweets & Polarities file in the destination bucket



5. Tweets and Polarities File – Contents



Code Blocks

CreateS3Bucket.java [1-4]

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class CreateS3Buckets
{

    public static void createBucket(AmazonS3 s3cli,String bucketName)
    {
        var it = s3cli.createBucket(bucketName);
    }

    public static void main(String[] args)
    {
        String id = Credentials.aws_access_key_id;
        String key = Credentials.aws_secret_access_key;
        String token = Credentials.aws_session_token;
```

```
        BasicSessionCredentials sessionCredentials = new
BasicSessionCredentials(id,key,token);
        AmazonS3 s3cli = AmazonS3ClientBuilder.standard().withCredentials(new
AWSStaticCredentialsProvider(sessionCredentials)).withRegion(Regions.DEFAULT_REGION)
.build();
        createBucket( s3cli," twitterdatab00899629");
        createBucket( s3cli," destinationbucketb00899629");

    }
}
```

UploadTweetFile.java

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import java.io.File;
import java.util.concurrent.TimeUnit;

public class UploadTweetFile {
    public static void main(String[] args) throws InterruptedException
    {
        CreateS3Buckets obj = new CreateS3Buckets();
        String id = Credentials.aws_access_key_id;
        String key = Credentials.aws_secret_access_key;
        String token = Credentials.aws_session_token;
        BasicSessionCredentials sessionCredentials = new
BasicSessionCredentials(id,key,token);
        AmazonS3 s3cli = AmazonS3ClientBuilder.standard().withCredentials(new
AWSStaticCredentialsProvider(sessionCredentials)).withRegion(Regions.US_WEST_2).buil
d();
        String s3Bucket1 = "twitterdatab00899629";
        File folderPath = new File("./src/tweets");
        File[] allFiles = folderPath.listFiles();
        for (File f : allFiles)
        {
            s3cli.putObject(s3Bucket1, f.getName(), f);
            System.out.println("The Tweet File - " + f.getName() + " has been
uploaded to the S3 Bucket - " + s3Bucket1);
        }
    }
}
```

ProcessTweetFile.py (Lamda Function to Process Tweet Files and Invoke AWS Comprehend on the Tweet Texts)[5-10]

```
import urllib.parse
import re
```

```
import json
import urllib.parse
import re
import json
import boto3
import logging


def lambda_handler(event, context):
    s3Client = boto3.client("s3")
    s3Upload = boto3.resource('s3')
    comprehendClient = boto3.client('comprehend')

    data = []

    destinationBucket = "destinationbucketb00899629"
    sourceBucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
encoding='utf-8')

    json_file_name = key.split('.')
    tweets = s3Client.get_object(Bucket=sourceBucket,
Key=key)['Body'].read().decode("utf-8")

    tweets = re.sub(r'[A-Za-z0-9]*@[A-Za-z]*\.?[A-Za-z0-9]*', "", tweets)

    tweetsWithoutEmojis = re.compile("["u"\U0001F600-\U0001F64F""]+", flags=re.UNICODE)

    tweets = tweetsWithoutEmojis.sub(r'', tweets)

    tweets = re.sub(r"[^a-zA-Z0-9\n]+", ' ', tweets)

    count = 0

    for tweet in tweets.splitlines():
        if len(tweet)>0:
            sentiment=comprehendClient.detect_sentiment(Text=tweet,LanguageCode='en')['
Sentiment']
            data.append({'Tweet Text': tweet,'Polarity':sentiment})
            count = count + 1

        if count == 10:
            json_str = json.dumps(data)
            s3Upload.Object(destinationBucket, json_file_name[0]+ ".json").put(Body =
(json_str))
```

```
            print(data)
            count = 0

    json_str = json.dumps(data)

    print(json_str)
```

Credentials.java

```
public class Credentials
{
public static String aws_access_key_id="ASIA3PYLANH44TJNTPFI";
public static String
aws_secret_access_key="QcReBN+F0n/dDqBqXbE7HLmmQkhHd7MInmsNP/Mn";
public static String
aws_session_token="FwoGZXIvYXdzEHUaDKSRjb6kVF9lk+/q0iLAAXd9WYWVu0z0W/12qKuUXbkGQMEda
QzgGP80N4U9ww3GCiQXYLSDDba9monVzbIviCi1UutFnPeAhl40FaSVw27Bdb0tayZ2dQA+K53TKZKYM5DYn
+eDh51Tz9tYwjoB7wcDB1vlyXu/TKWw5JfsGR154L93pA0tHod0cKQlVRoCPBCMI8+Hd+0S9QSYAf3vpPY1q
U1Hgmq/8iJC3aCXBCLf9PsUUEoA9w3GbyFQtlkQrXFylLNBFCHwLqSz+OtXAyiMhdOVBjItLbecv2rApPahR
v/JH6eH0gyOY24+iEowQBuDofl7ni/joEv78yMj4+LpFwLz";
}
```

Citations

[1] "AWS SDK for Java." *Amazon Web Services, Inc.*, 15 Sept. 2012, aws.amazon.com/sdk-for-java/. Accessed 03 July 2022.

[2] "Awsdocs/Aws-Doc-Sdk-Examples." *GitHub*, 5 Sept. 2019, github.com/awsdocs/aws-doc-sdk-examples/blob/main/java/example_code/s3/src/main/java/aws/example/s3/CreateBucket.java. Accessed 04 July 2022.

[3] baeldung. "AWS S3 with Java | Baeldung." *Www.baeldung.com*, 24 July 2017, www.baeldung.com/aws-s3-java. Accessed 04 July 2022.

[4] "Managing Dependencies with AWS SDK for Java – Bill of Materials Module (BOM)." *Amazon Web Services*, 10 Aug. 2015, aws.amazon.com/blogs/developer/managing-dependencies-with-aws-sdk-for-java-bill-of-materials-module-bom. Accessed 04 July 2022.

[5] anubhava. "Python - Regular Expression to Extract Named Entities from Text Just Based on Capitalization." *Stack Overflow*, 9 May 2017, stackoverflow.com/questions/43972800/regular-

expression-to-extract-named-entities-from-text-just-based-on-capitalizat. Accessed 05 July 2022.

[6] AWS. "How to Read Files from S3 Using Python AWS Lambda." *Gcptutorials*, 9 Mar. 2013, www.gcptutorials.com/post/how-to-read-files-from-s3-using-python-aws-lambda. Accessed 05 July 2022.

[7] "Tutorial: Using an Amazon S3 Trigger to Invoke a Lambda Function - AWS Lambda." *Docs.aws.amazon.com*, 17 June 2013, docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html. Accessed 05 July 2022.

[8] "c# - ^[A-Za-Z][A-Za-z0-9]* regular expression?," *Stack Overflow*, Oct. 31, 2009. https://stackoverflow.com/questions/1653425/a-za-z-a-za-z0-9-regular-expression (accessed Jul. 05, 2022).

[9] "Tweets Preprocessing," *kaggle.com*, Apr. 18, 2020. https://www.kaggle.com/code/quentinsarrazin/tweets-preprocessing/notebook (accessed Jul. 05, 2022).

[10] C. Munns, "Using AWS Lambda and Amazon Comprehend for sentiment analysis," *Amazon Web Services*, Apr. 11, 2018. https://aws.amazon.com/blogs/compute/using-aws-lambda-and-amazon-comprehend-for-sentiment-analysis/ (accessed Jul. 05, 2022).