Part-B

Name: Kishoreganesh Sundararajan   Banner ID: B00934548
Gitlab: https://git.cs.dal.ca/kishoregan/csci5410-summer-23-b00934548.git

Table of Contents:

Table of Figures:

All services demonstrated in this assignment are owned by Amazon web service [1].

1. Flowchart of operations

The operations begins by creating an index.html file with profile information, including name, banner number, email, and a declaration of independent work. A Java program using AWS SDK for Java is written to create an S3 bucket. The index.html file is then uploaded to the S3 bucket. Hosting is enabled for the bucket, and the S3 bucket policy is modified to allow the hosting of the static webpage. The flow is shown in **Figure 1**.
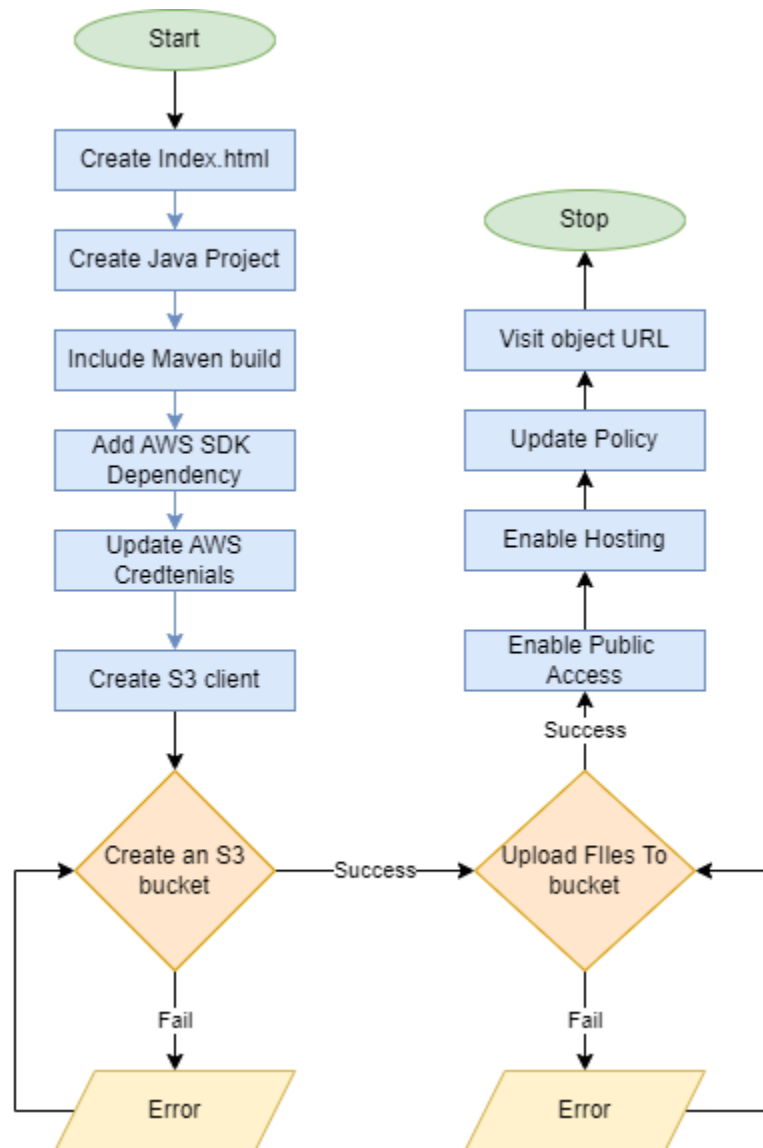

Figure 1: Flowchart of operations.

2. Overall observation on Java AWS SDK

The AWS SDK for Java simplifies working with the Amazon S3 [2] service by allowing us to create an S3 client [3], set up credentials, create an S3 bucket, and uploading a file. The SDK makes it simple to create an S3 client by using the S3Client class, which serves as a gateway for interacting with the S3 service. It streamlines the authentication process by allowing us to set up credentials directly in the code, such as specifying our AWS access key ID and secret access key (However it is not the best practice). In addition, the SDK provides the CreateBucketRequest [4] class for creating an S3 bucket, which allows us to specify the bucket name and other settings.

When it comes to uploading a file, we can use the PutObjectRequest class [5], which allows to specify the target bucket, the file's key (name), and the file itself. The SDK handles the complexities of creating API requests, signing them, and managing responses. Overall, the AWS SDK for Java simplifies the process of integrating S3 functionality into our Java application by simplifying the creation of an S3 client, setting up credentials, creating an S3 bucket, and uploading a file. I have simplified my overall observation of AWS JAVA in **Figure 2**.
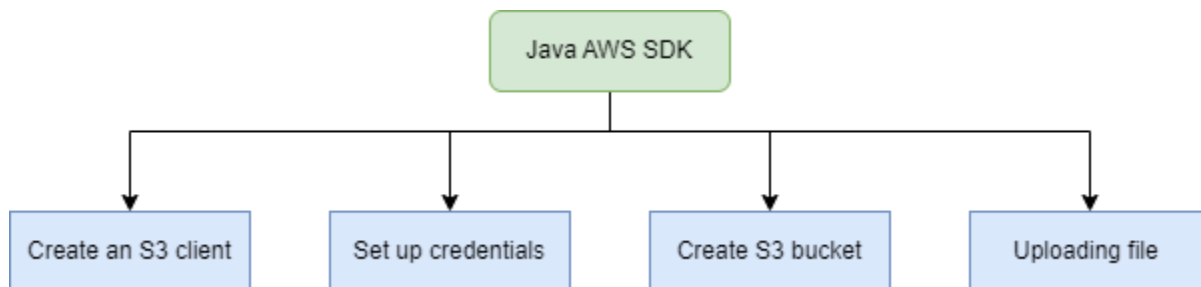
Figure 2: Observations on Java AWS SDK for S3.

3. Screenshots of the S3Bucket and operations

a. Creating an HTML file:

An HTML file named **index.html**, which contains my profile information including full name, banner number, email, and a declaration is written. As shown in **Figure 3**.

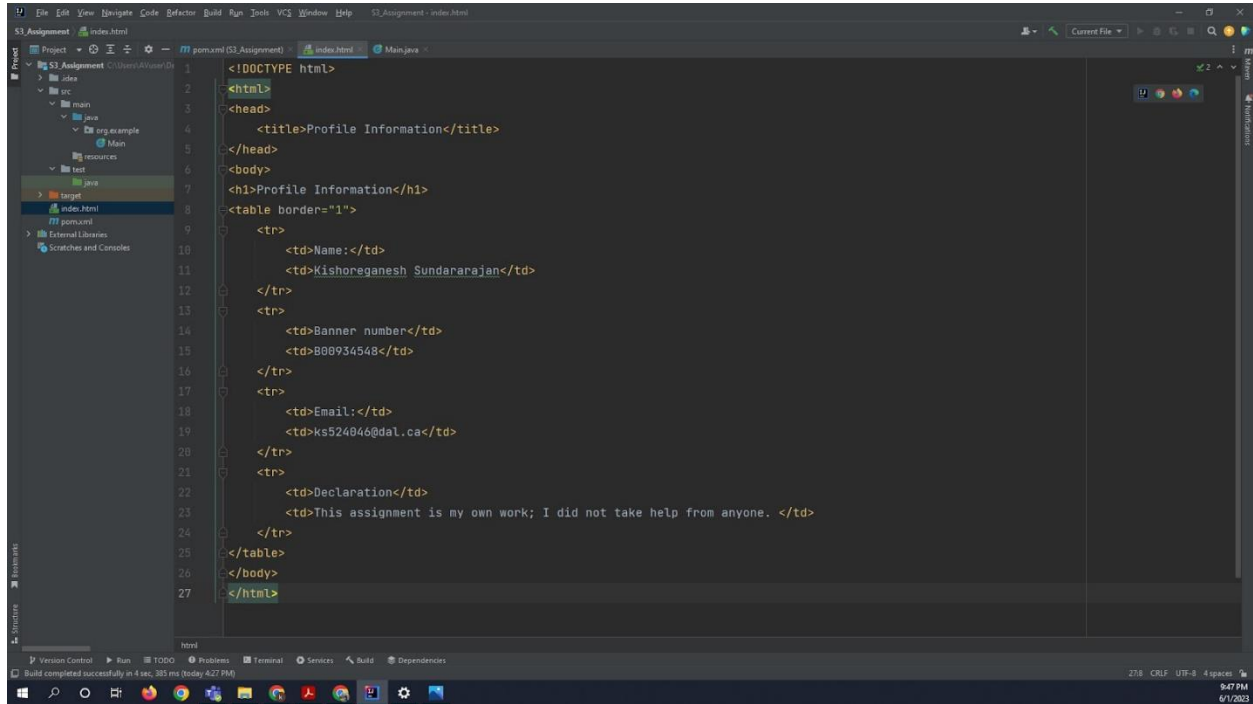*Figure 3: Depicting the index.html file*

b. Writing a Java program using SDK specifications for creating an S3 bucket.

For creating an s3 bucket, we need to have **AWS SDK** for Java as a dependency in the Java Maven project. The maven dependency is added to the **pom.xml** file. The dependency can be taken from the official Maven repository website [6]. And the dependency is added as shown in **Figure 4**.

Figure 4: AWS SDK added in maven dependency.

Before creating the S3 bucket we need to set up the AWS credentials in the local machine using **AWS CLI** [7]. This AWS credentials like access key and secret key can be found in AWS academy in details section as shown in **Figure 5** and **Figure 6** his enables us to protect our access key ID and secret access key from any threats. The credentials are set through the command line as shown in **Figure 7**.

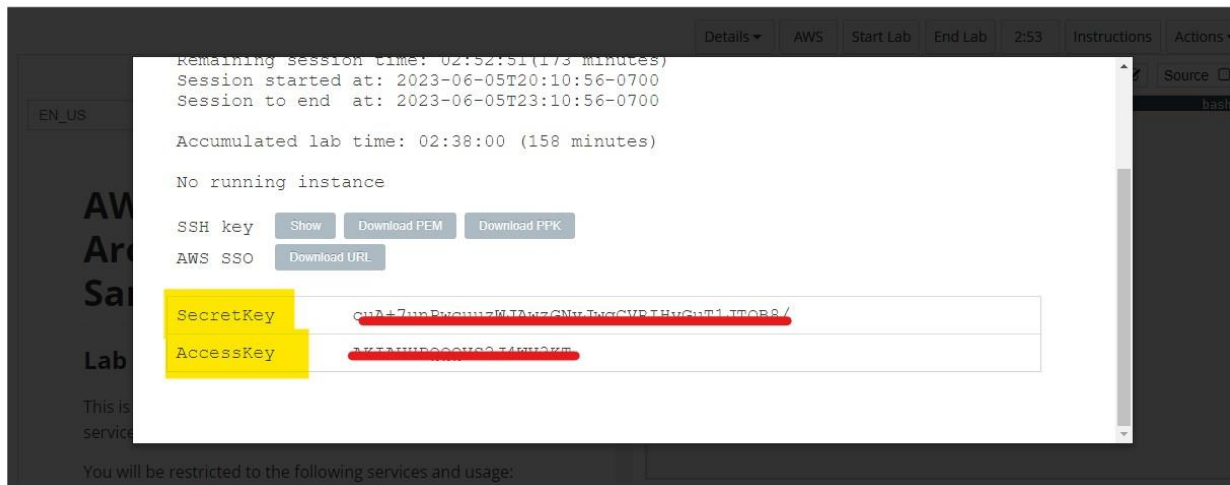
Figure 5: Click details in Sandbox

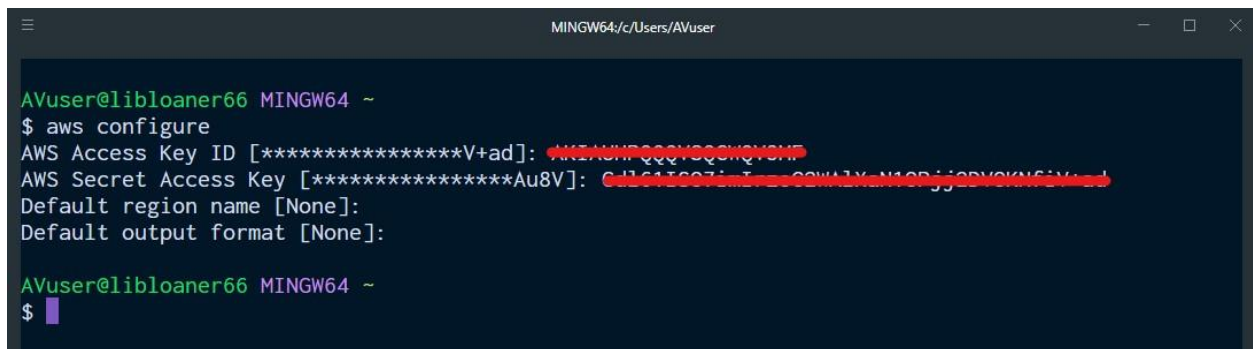*Figure 6: AWS Secret key and Access key for the session are available.*

Figure 7: AWS Credentials are set through cmd.

The code in the below **Figure 8**, utilizes the AWS SDK for Java to create an S3 bucket. It defines a createS3Client method that sets up an S3Client object with the specified region(US_EAST_1) and default credentials provider [8] (as provided in AWS CLI). The createBucket method takes an S3Client and a bucket name as input, creating a CreateBucketRequest and sending it to the S3 service.

If successful, it prints a confirmation message. In the main method, a bucket name and region are specified, and the createS3Client method is called to create the client. Finally, the createBucket method is invoked, handling any potential exceptions. Overall, this program creates an S3 bucket using the AWS SDK for Java.

Figure 8: Code to create S3 bucket.

The following **Figure 9,** shows the output of the program. The output of the provided code will be a message indicating whether the S3 bucket creation was successful or not.

Figure 9: Output for creating bucket

Figure 10 depicts the creation of the s3 bucket was successful. The s3 bucket we created through code is now visible in the AWS console.

Figure 10: Bucket creation in AWS console.

Since we did not upload any content to the bucket it is empty as of now. **Figure 11** shows an empty bucket present in the AWS console.
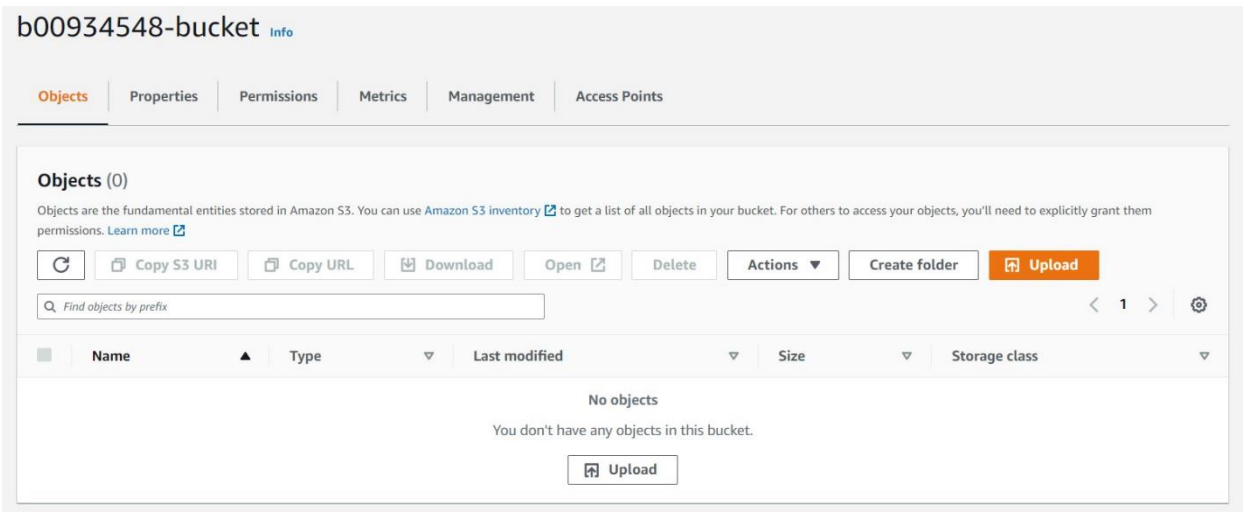
Figure 11: Empty bucket in AWS console

c. Java program to upload the file into a bucket

The code in the below **Figure 12** defines a method called uploadFile that utilizes the AWS SDK for Java to upload a file to an S3 bucket. This method takes an S3Client object, the name of the target bucket, the file path of the local file to be uploaded, and a key or name for the object in the S3 bucket. It creates a File object from the specified file path and constructs a PutObjectRequest with the bucket name and key. The putObject method is then invoked on the S3Client to upload the file contents to the S3 bucket. Finally, it prints a success message indicating that the file was uploaded successfully, including the key value. Overall, this code enables easy file uploads to an S3 bucket using the AWS SDK for Java.
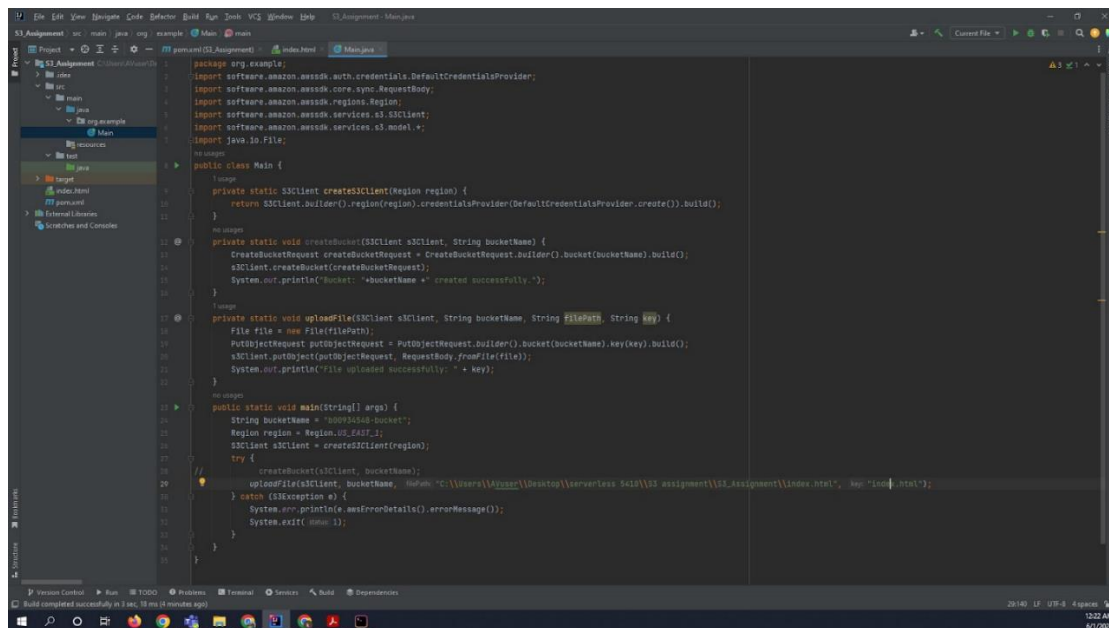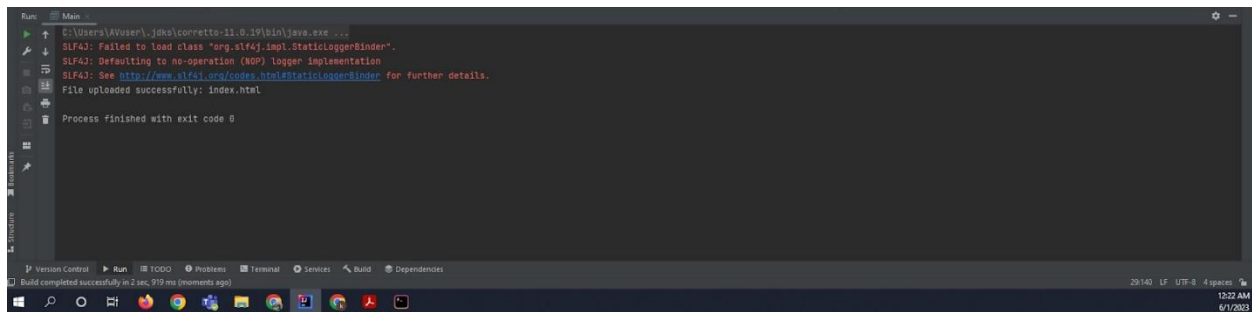
Figure 12:: Code for uploading the file into bucket

8

The following **Figure 13** shows the output of the program. The output of the provided code will be a message indicating whether the file is uploaded successfully or not.

Figure 13: Output for uploading HTML file successfully.

Now the file is uploaded to the s3 bucket **(b00934548-bucket)** we created earlier. **Figure 14** below shows the file which is uploaded to the AWS console.
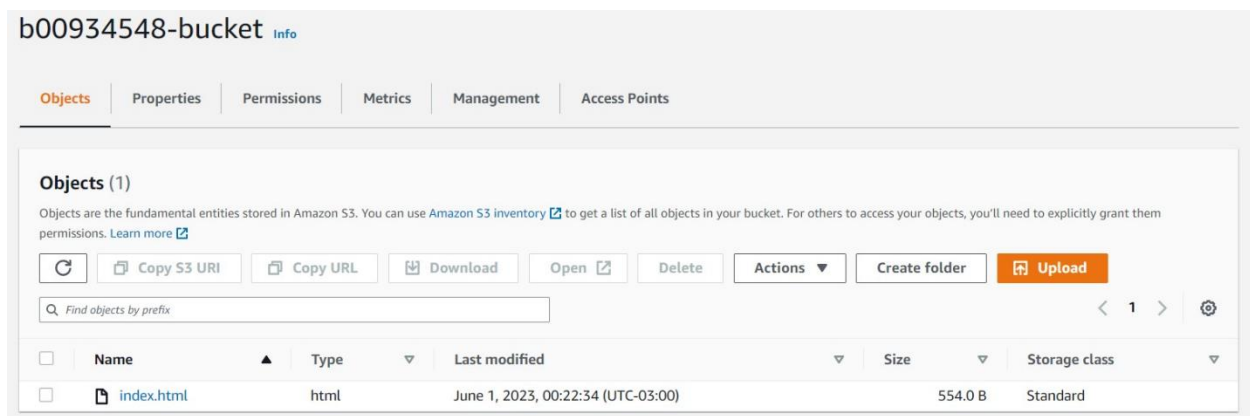
Figure 14: HTML file added to the bucket (b00934548-bucket).

d. Enabling hosting and changing policy.

Now the file is hosted in the s3 bucket but it is not open for public access since allowing public access to an S3 bucket without proper authorization can lead to potential security risks, data breaches, and unauthorized access to our data. Blocking public access helps mitigate these risks and ensures that we have explicit control over who can access our bucket and its contents. By default, public access is blocked as seen in the **Figure 15**.
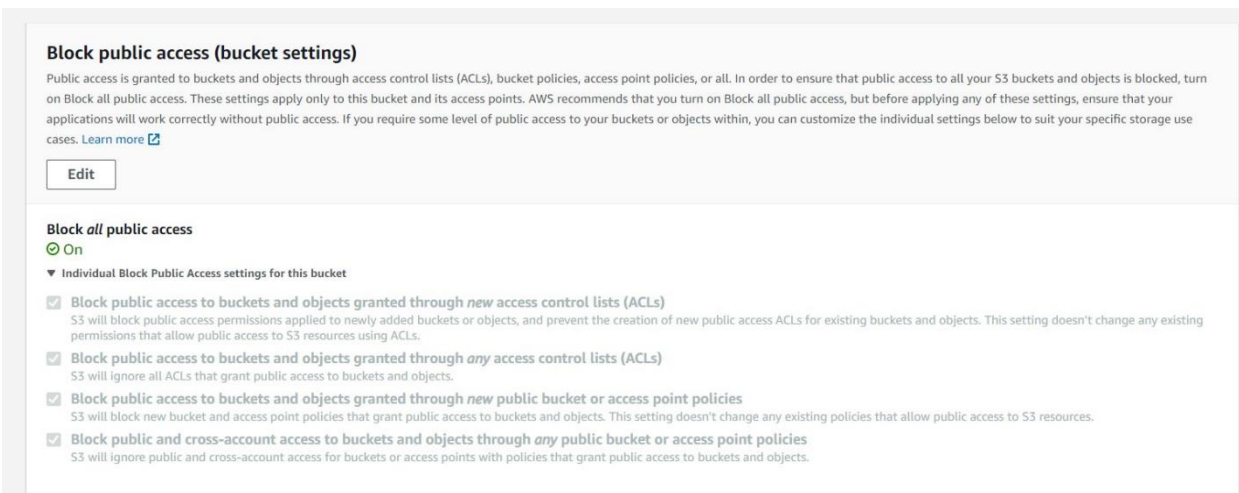
Figure 15: Public access is blocked.

However, it can be configured and disabled as per our needs as shown in **Figure 16** and **Figure 17**. Aws allows various configurations of Hosting a file in an s3 bucket.
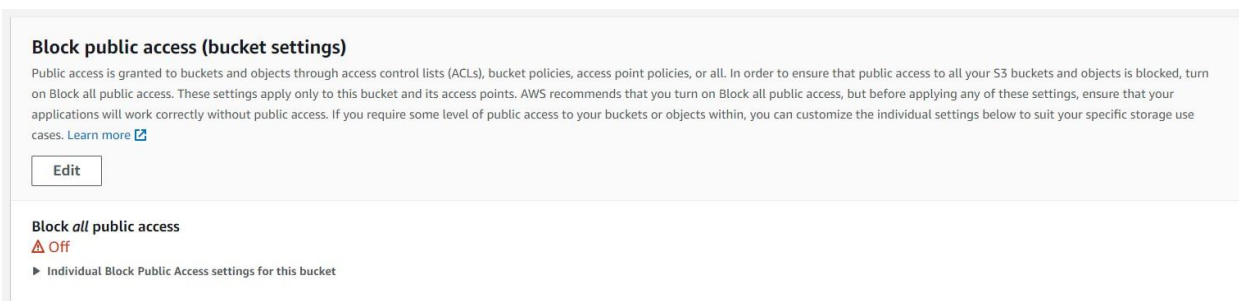
Figure 16: Public access is enabled.

Policy

```
1 ▾ {
2       "Version": "2012-10-17",
3 ▾     "Statement": [
4 ▾         {
5               "Sid": "PublicReadGetObject",
6               "Effect": "Allow",
7               "Principal": "*",
8 ▾             "Action": [
9                   "s3:GetObject"
10              ],
11 ▾            "Resource": [
12                  "arn:aws:s3:::b00934548-bucket/*"
13              ]
14          }
15      ]
16  }
```
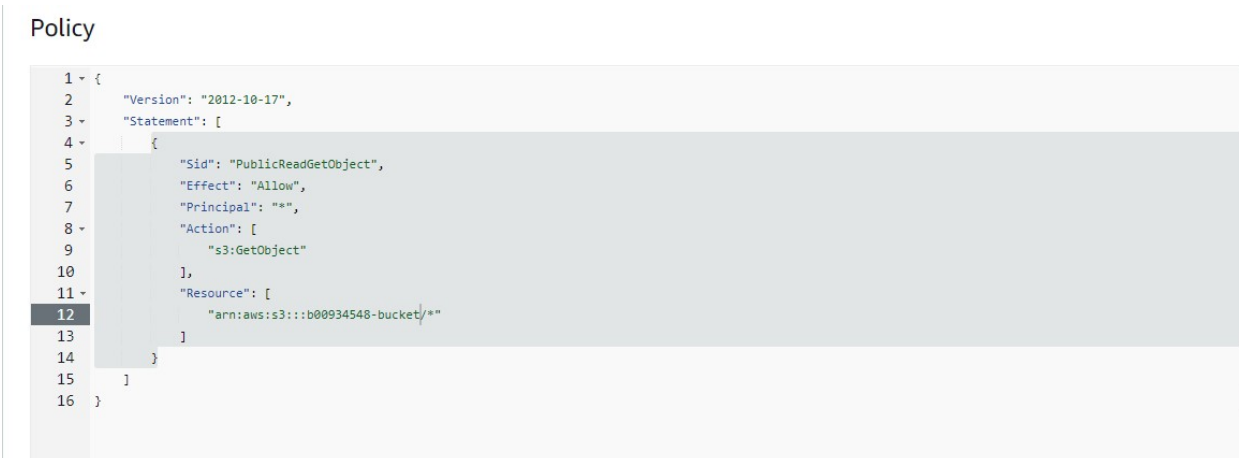
Figure 17: Updating the policy for public access.

After we edit S3 Block Public Access settings, we can add a bucket policy [9] to grant public read access to our bucket. When we grant public read access, anyone on the internet can access our bucket as seen in **Figure 18.**
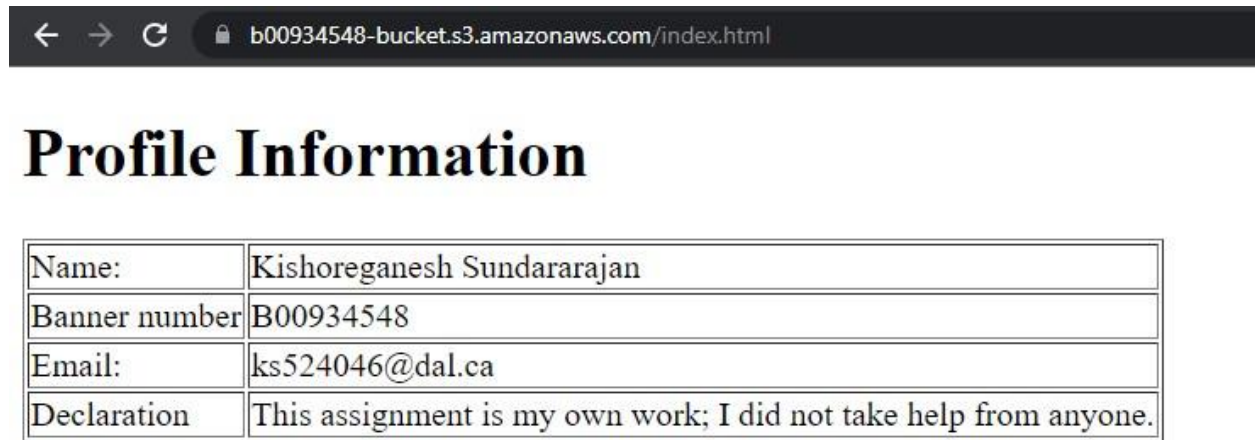
Figure 18: Profile Information is hosted on public access.

4. Challenges faced and solutions

During the course of this assignment, after creating the bucket and placing the file inside the bucket. It was regular practice for me to delete the file and bucket in the AWS console because running a bucket with a file may reduce my credits to some extent, but deleting the file and bucket was not taking much more time than expected because it prompts us multiple time to confirm that we really need to delete a bucket and file. This was initially a frustrating for me while I was having some issues with my index.html and some part of the code and want to delete my bucket and file regularly. So I figured out there is also DeleteObjectRequest class [10] which can delete the file from the bucket and also deletebucket method in S3client class to delete an bucket [11].

5. Program script

package org.example;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;

import software.amazon.awssdk.core.sync.RequestBody;

import software.amazon.awssdk.regions.Region;

import software.amazon.awssdk.services.s3.S3Client;

```java
import software.amazon.awssdk.services.s3.model.*;

import java.io.File;

import java.security.Policy;

import java.sql.Statement;

public class Main {

    public static S3Client createS3Client(Region region) {

        return
S3Client.builder().region(region).credentialsProvider(DefaultCredentialsProvider.create()).build();

    }

    public static void createBucket(S3Client s3Client, String bucketName) {

        CreateBucketRequest createBucketRequest =
CreateBucketRequest.builder().bucket(bucketName).build();

        s3Client.createBucket(createBucketRequest);

        System.out.println("bucket created");

    }


    public static void uploadFile(S3Client s3Client, String bucketName, String filePath, String key) {

        File file = new File(filePath);

        PutObjectRequest putObjectRequest =
PutObjectRequest.builder().bucket(bucketName).key(key).build();

        s3Client.putObject(putObjectRequest, RequestBody.fromFile(file));

        System.out.println("file uploaded");

    }

    public static void deleteFile(S3Client s3Client, String bucketName, String key){

        DeleteObjectRequest request=
DeleteObjectRequest.builder().bucket(bucketName).key(key).build();

        s3Client.deleteObject(request);
```

```java
        System.out.println("file dleted");

    }

    public static void deleteBucket(S3Client s3Client, String bucketName){

        DeleteBucketRequest
deleteBucketRequest=DeleteBucketRequest.builder().bucket(bucketName).build();

        s3Client.deleteBucket(deleteBucketRequest);

        System.out.println("bucket deleted");

    }

    public static void main(String[] args) {

        String bucketName = "b00934548-bucket";

        Region region = Region.US_EAST_1;

        S3Client s3Client = createS3Client(region);

        try {

//          createBucket(s3Client, bucketName);

//          uploadFile(s3Client, bucketName, "C:\\Users\\AVuser\\Desktop\\serverless 5410\\S3
assignment\\S3_Assignment\\index.html", "index.html");

            deleteFile(s3Client,bucketName,"index.html");

            deleteBucket(s3Client,bucketName);

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());

            System.exit(1);

        }

    }
}
```

Gitlab Repository: https://git.cs.dal.ca/kishoregan/csci5410-summer-23-b00934548.git

References

[1]   Amazon Web Services, Inc., "Amazon Web Services (AWS) - Cloud Computing Services," 2023.
      [Online]. Available: https://aws.amazon.com/. [Accessed 1 June 2023].

[2]   Amazon Web Services, Inc., "Amazon S3," 2023. [Online]. Available: https://aws.amazon.com/s3/.
      [Accessed 01 June 2023].

[3]   Amazon Web Services, Inc., "Interface S3Client," 2023. [Online]. Available:
      https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/S3Client.html.
      [Accessed 3 June 2023].

[4]   Amazon Web Services, Inc., "Class CreateBucketRequest," 2023. [Online]. Available:
      https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/model/Create
      BucketRequest.html. [Accessed 03 June 2023].

[5]   Amazon Web Services, Inc., "Class PutObjectRequest," 2023. [Online]. Available:
      https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/services/s3/model/PutObj
      ectRequest.html. [Accessed 03 June 2023].

[6]   MvnRepository, "AWS Java SDK :: AWS Core » 2.20.75," 2023. [Online]. Available:
      https://mvnrepository.com/artifact/software.amazon.awssdk/aws-core/2.20.75. [Accessed 3 June
      2023].

[7]   Amazon Web Services, Inc., "AWS Command Line Interface," 2023. [Online]. Available:
      https://aws.amazon.com/cli/. [Accessed 03 June 2023].

[8]   Amazon Web Services, Inc., "Class DefaultCredentialsProvider," 2023. [Online]. Available:
      https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/auth/credentials/DefaultC
      redentialsProvider.html. [Accessed 03 June 2023].

[9]   Amazon Web Services, Inc., "Setting permissions for website access," 2023. [Online]. Available:
      https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteAccessPermissionsReqd.html.
      [Accessed 3 June 2023].

[10] CodeJava.net, "AWS Java SDK S3 Delete Objects Examples," 2023. [Online]. Available:
      https://www.codejava.net/aws/delete-s3-objects-examples. [Accessed 03 June 2023].

[11] CodeJava.net, "AWS Java SDK S3 Delete Buckets Examples," 2023. [Online]. Available:
      https://www.codejava.net/aws/delete-buckets-examples. [Accessed 03 June 2023].