CSCI 5410 Serverless Data Processing

Assignment 1 Report

Prepared by:
Vikram Venkatapathi - B00936916

Master of Applied Computer Science (Summer'23)
Faculty of Computer Science
Dalhousie University

GitLab Repo link : **https://git.cs.dal.ca/vikramv/csci5410-summer-23-b00936916/-/tree/A1**

Contents

Part I.

PART A

Provide summary

1. Introduction

The given paper titled as *Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach* authored by "P.Vahidinia, B.Farahani and F.S.Aliee", focuses on addressing the challenge of reducing cold start delays in serverless computing[1].

This report attempts to review the given literature and then provides a summary.

2. Presentation by the Author

The authors present a two-layer approach to address the challenge of reducing cold start delays in serverless computing. The first layer utilizes a reinforcement learning algorithm to discover the function invocation pattern and determine the idle container window. The second layer employs an LSTM (Long Short-Term Memory model) to predict the next invocation time and determine the required number of prewarmed containers.

3. Addressed Issue

The specific issue addressed in the paper is the reduction of cold start delays in serverless computing. The authors highlight that existing solutions, such as keeping containers warm, often result in fixed policies and memory waste. The proposed approach aims to dynamically determine the idle-container window and the number of prewarmed containers based on the invocation pattern, balancing the reduction of cold start latency with memory consumption.

4. Experiments or Studies

The authors conducted experiments to evaluate the performance of their approach. They used an I/O bound function that sends an HTTP request to a web page and receives a text file, with a response time of around 6 seconds. Two simulations were performed: one using the Openwhisk platform with default parameters and another using the proposed approach with new parameters derived from their method. The evaluations focused on comparing the idle-container window, number of cold start occurrences, memory consumption, and execution of invocations on prewarmed containers.

5. Analysis and Findings

The findings of the study demonstrate the effectiveness of the proposed approach in reducing cold start delays and optimizing resource utilization. The authors observed that by dynamically determining the idle-container window based on the invocation pattern, they were able to reduce the number of cold start occurrences and control memory consumption. The number of cold start delays in the proposed approach was comparable to the results of the Openwhisk platform, indicating its effectiveness in mitigating cold start issues. Furthermore, the approach showed improvements in memory consumption time and a 22.65% enhancement in the execution of invocations on prewarmed containers compared to the platform's default settings.

6. Conclusion

Overall, the paper presents a two-layer approach that effectively addresses the challenge of reducing cold start delays in serverless computing. The proposed method dynamically determines the idle-container window and the number of prewarmed containers, resulting in improved performance while controlling memory consumption. The experiments and evaluations validate the effectiveness of the approach in reducing cold start occurrences and improving the execution of invocations on prewarmed containers.

Part II.

PART B

AWS S3 experiment

7. Procedure followed for the given experiment

7.1. Flowchart

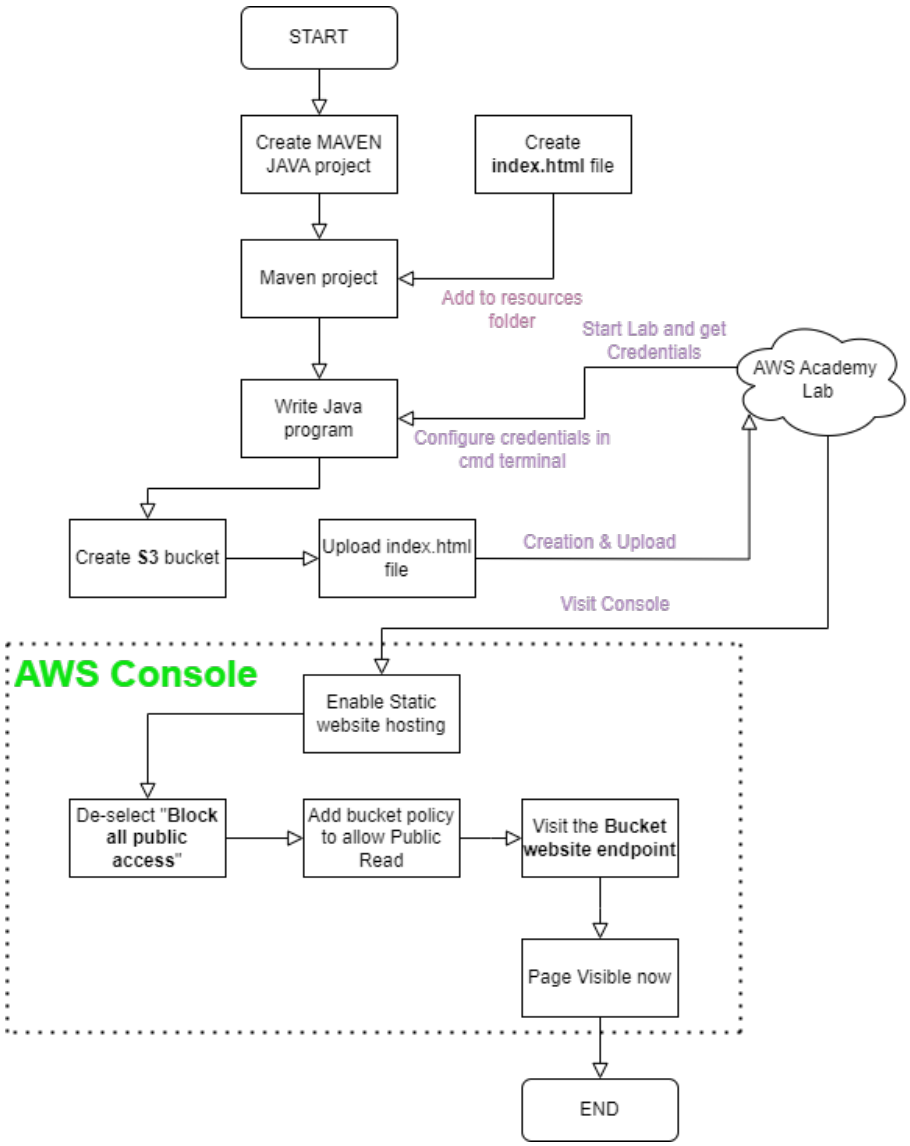

Figure 7.1.: ***Flowchart describing the operations that I have performed.***

7.2. My overall observation of the Java SDK

The Java SDK for AWS S3 provides developers with a comprehensive set of tools and APIs for interacting with S3 buckets. It offers extensive documentation and resources, making it easy to understand and utilize the SDK effectively. The SDK supports various operations such as creating, deleting, and managing S3 buckets, as well as uploading, downloading, and manipulating objects within the buckets. It integrates well with other AWS services, allowing developers to build complex applications and workflows. The SDK is optimized for performance, with features like multipart uploads for efficient handling of large files. Overall, the Java SDK for AWS S3 is a powerful and reliable tool for working with S3 buckets in Java applications.

7.3. Configurse session credentials

- Open a command terminal & Enter **aws configure**
- Enter the Access key ID & Secret access key (found in the details section of AWS academy, after Starting the lab)
- This will create and configure the credentials file inside the folder **`C:\Users\<your_name>\.aws`**

7.4. Maven project creation

- A maven Java project was created
- Following AWS dependencies were added to the *pom.xml* file:
  - SDK:

```
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.2.0</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
```

  - S3:

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
    <version>2.2.0</version>
</dependency>
```

7.5. Creation of Index.html file

- A *index.html* file was created inside the *resources* folder of the **maven project**

index.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Profile</title>
</head>
<body>
<h1>My Profile Information</h1>
<p>
    <strong>Name:</strong> Vikram Venkatapathi<br>
    <strong>Banner Number:</strong> B00936916<br>
    <strong>Email:</strong> vk485591@dal.ca<br>
</p>
<p>
    <em>This assignment is my own work; I did not take help from
       anyone.</em>
</p>
</body>
</html>
```

7.6. Java program to create S3 Bucket

- A Java program was written using the AWS SDK to
  - use the session specific ACCESS_KEY_ID & SECRET_ACCESS_KEY
  - create an S3 bucket following the naming conventions (Bucket name : b00936916-a1)
  - upload the index.html file to the created bucket.
- Comments were added following the Java Docs specification (Refer 7.2)

```
/**
 * Uploads a file to the specified S3 bucket.
 *
 * @param s3Client The S3 client
 * @throws URISyntaxException If the resource file URI is invalid
 */
1 usage  vikramv *
private static void uploadFiletoS3(S3Client s3Client) throws URISyntaxException {
    // Obtain the file name and path
    String fileName = "index.html";
    Path resourcePath = Paths.get(Main.class.getClassLoader().getResource(fileName).toURI());
```

Figure 7.2.: ***Comments added, based on Java Docs specification***

- Main.java

```java
package org.example;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;

public class Main {
    private static String bucketName = "b00936916-a1";
    private static String region = "us-east-1";

    /**
     * Main method to create an S3 bucket and upload a file to it.
     *
     * @param args Command-line arguments
     * @throws URISyntaxException If the file path is not a valid URI
     */
    public static void main(String[] args) throws URISyntaxException {
        // Create the credentials provider
        //Retrieve the AWS Access Key ID & AWS Secret Access Key from ./aws/
            credentials file
        ProfileCredentialsProvider credentialsProvider =
            ProfileCredentialsProvider.create();

        // Create the S3 client
        S3Client s3Client = S3Client.builder()
                .region(Region.of(region))
                .credentialsProvider(credentialsProvider)
                .build();

        // Call the method to create an S3 bucket
        createS3Bucket(s3Client);

        // Call the method to upload a file to S3
        uploadFiletoS3(s3Client);
    }

    /**
```

```
63          * Creates an S3 bucket with the specified bucket name.
64          *
65          * @param s3Client The S3 client
66          */
67         private static void createS3Bucket(S3Client s3Client) {
68             try {
69                 // Create a request to create an S3 bucket with the specified
                       bucket name
70                 CreateBucketRequest createBucketRequest = CreateBucketRequest.
                       builder()
71                         .bucket(bucketName)
72                         .build();
73
74                 // Send the create bucket request to the S3 service
75                 CreateBucketResponse createBucketResponse = s3Client.
                       createBucket(createBucketRequest);
76             } catch (S3Exception e) {
77                 if (e.awsErrorDetails().errorCode().equals("BucketAlreadyExists"
                       )) {
78                     System.err.print("Failed to create bucket. Bucket name
                           already exists");
79                 }
80             }
81         }
82
83         /**
84          * Uploads a file to the S3 bucket.
85          *
86          * @param s3Client The S3 client
87          * @throws URISyntaxException If the file path is not a valid URI
88          */
89         private static void uploadFiletoS3(S3Client s3Client) throws
               URISyntaxException {
90             // Obtain the file name and path
91             String fileName = "index.html";
92             Path resourcePath = Paths.get(Main.class.getClassLoader().
                   getResource(fileName).toURI());
93             String filePath = resourcePath.toString();
94
95             // Create a request to put the object in the S3 bucket
96             PutObjectRequest putObjectRequest = PutObjectRequest.builder()
97                     .bucket(bucketName)
98                     .key(filePath)
99                     .build();
100
101            // Upload the file to S3
102            s3Client.putObject(putObjectRequest, RequestBody.fromFile(new File(
                   filePath)));
103        }
```

```
104 }
105
106 /** References :
107 * Code snippets were referred from official AWS SDK documentation:
108
109 @see <a href="https://docs.aws.amazon.com/sdk-for-java/latest/developer-
        guide/examples-s3-buckets.html ">Create S3 bucket</a>
110 @see <a href="https://docs.aws.amazon.com/sdk-for-java/latest/developer-
        guide/examples-s3-objects.html">PUT object in S3 bucket</a>
111 */
```

7.7. Operations on the AWS console

1. Goto S3 Service
2. Select **Buckets**, in the left-hand side menu
3. Goto **Permissions** section
4. Scroll down, Edit **Static website hosting**
   a) Select **Enable**
   b) Specify the uploaded index file name in the **Index document** text box
   c) Click **Save changes**
5. Go to the created bucket
   a) Go to the **Properties** section
   b) Scroll to the end, till "Static website hosting".
   c) Visit the **Bucket website endpoint**
   d) **Access Denied** - because the bucket is **not Public**, and doesn't allow **Public Read Access**
6. Go to the created Bucket
   a) Go to **Permissions** section
   b) De-select **Block all public access**
   c) Go to the **Bucket policy** section, **Edit** it
   d) Add the following policy and **Save changes**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::b00936916-a1/*"
        }
    ]
}
```

7. Go visit the website endpoint, Hit Refresh
8. You can view the index.html file now.

NOTE: Refer to the following Screenshots for the steps mentioned above.

8. Screenshots

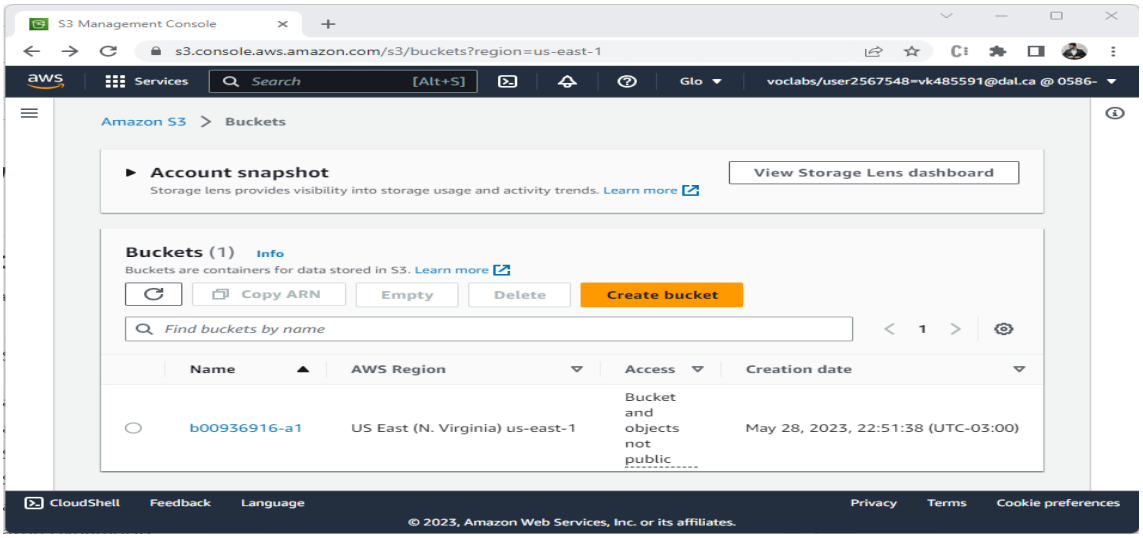

Figure 8.1.: ***Bucket created***

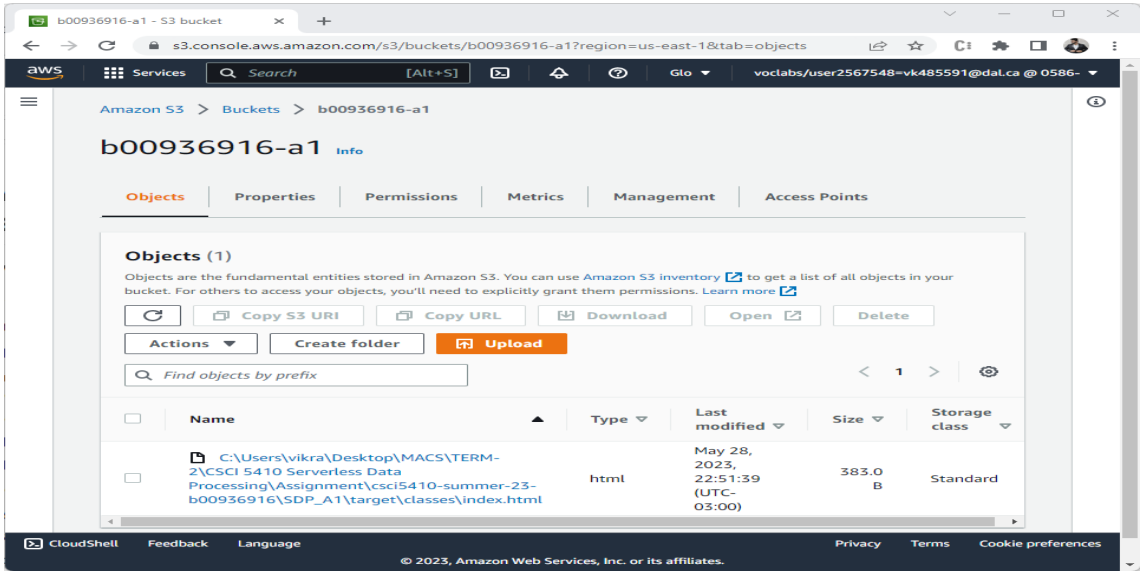

Figure 8.2.: ***Index.html file uploaded***

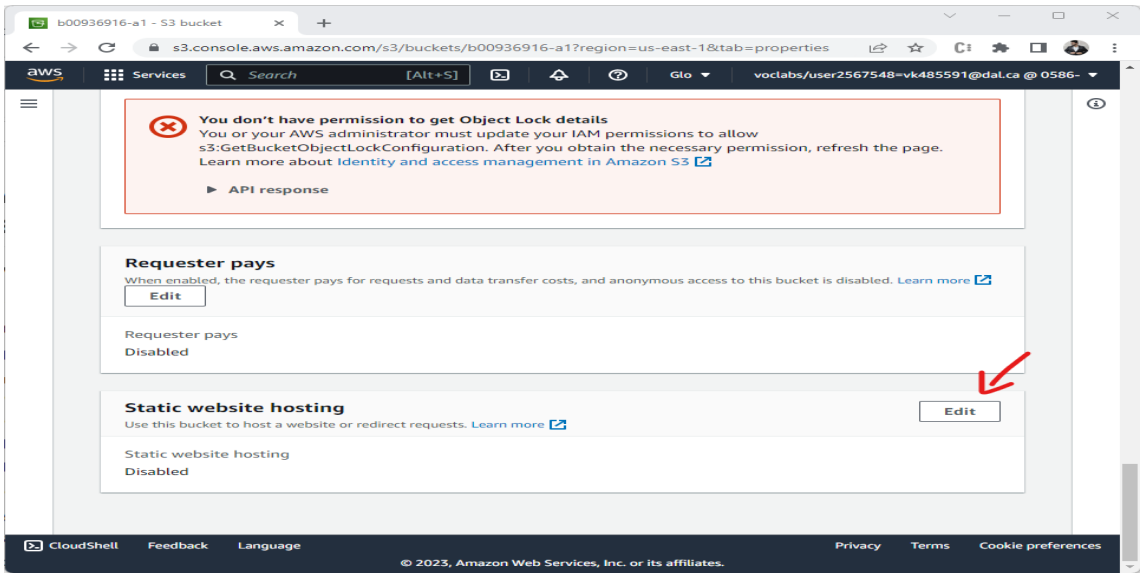

Figure 8.3.: ***Enable website hosting***

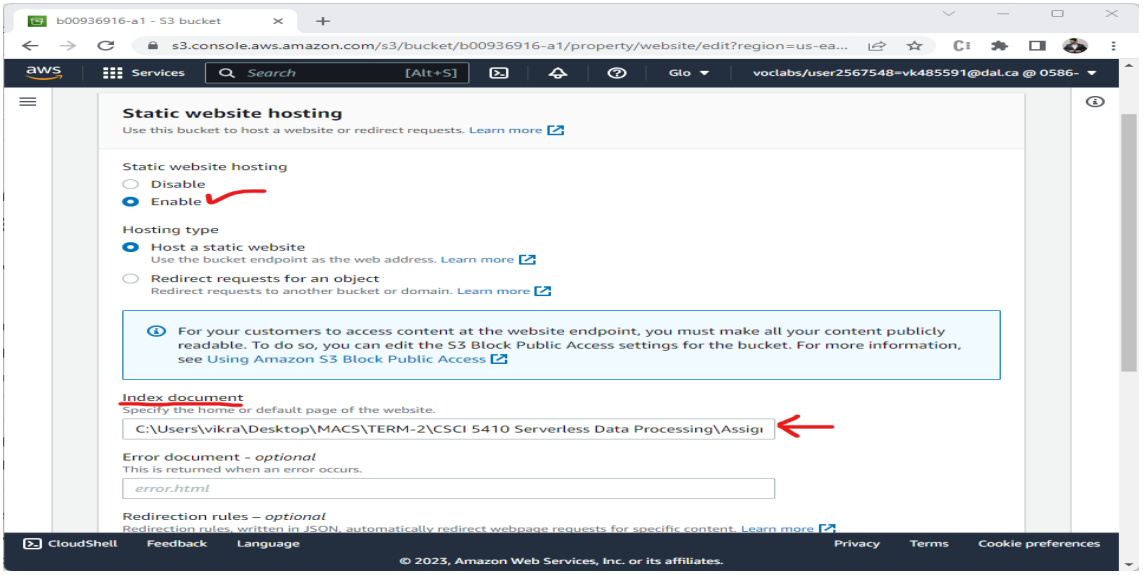

Figure 8.4.: ***Specify the index.html file name to host the website***

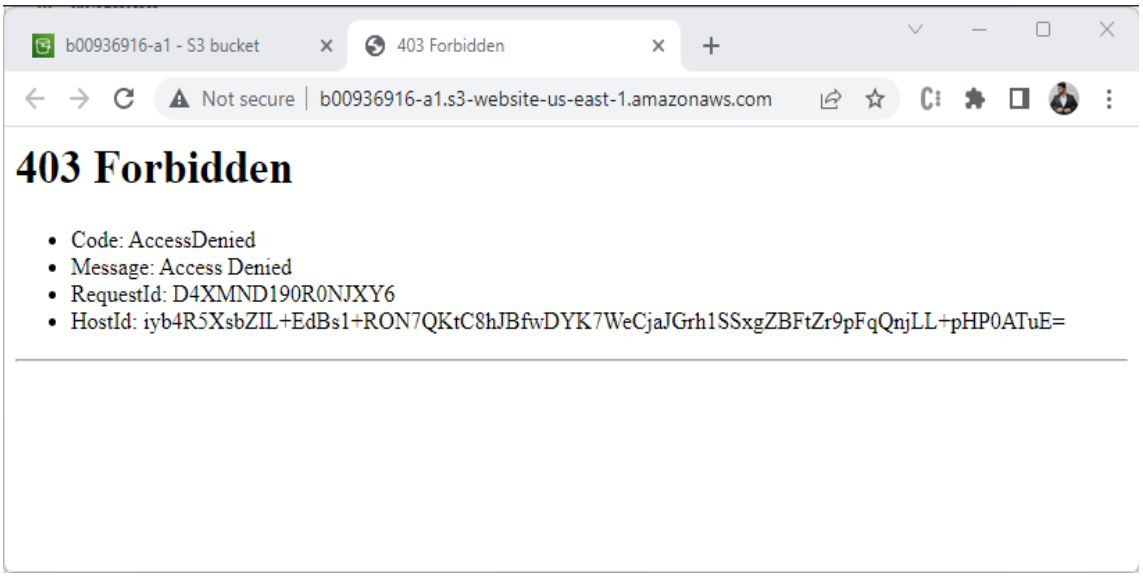

Figure 8.5.: ***Access is denied since bucket is not public and doesn't allow READ access***

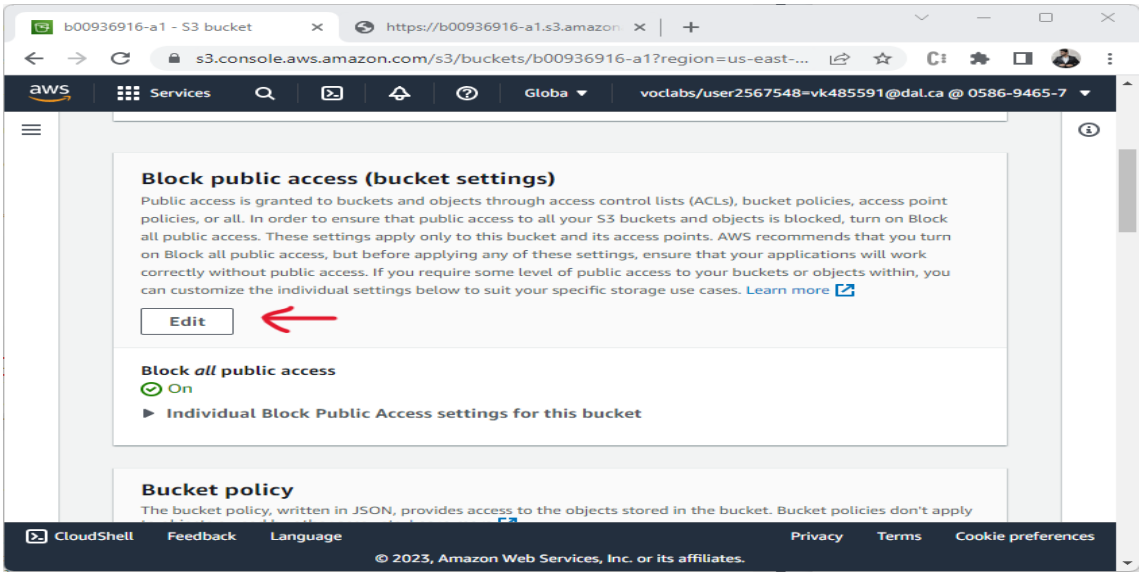

Figure 8.6.: ***Edit block all public access***

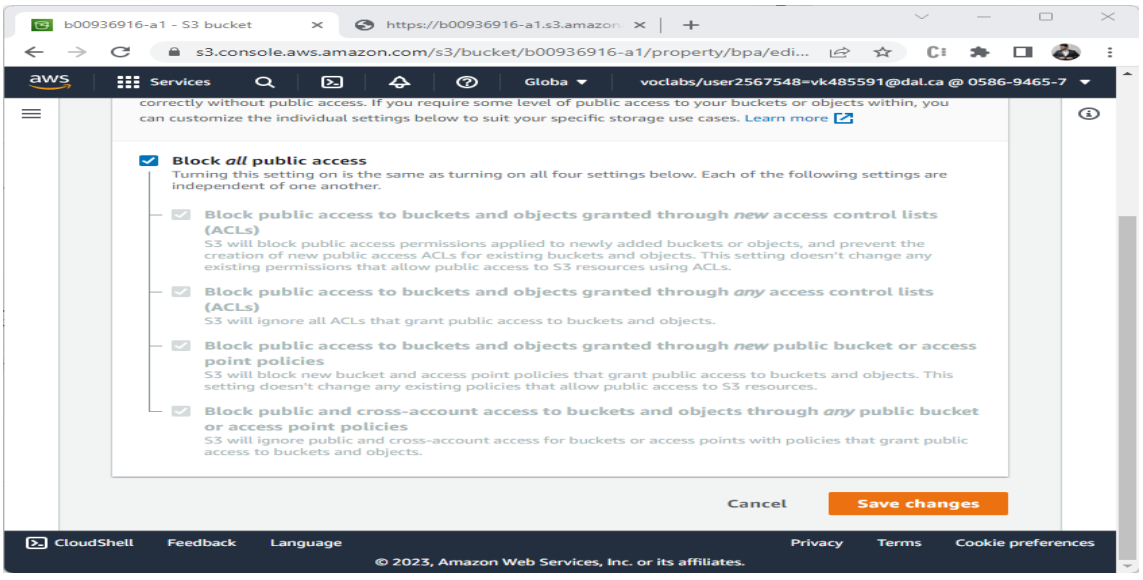

Figure 8.7.: ***De-select Bloack all public access***

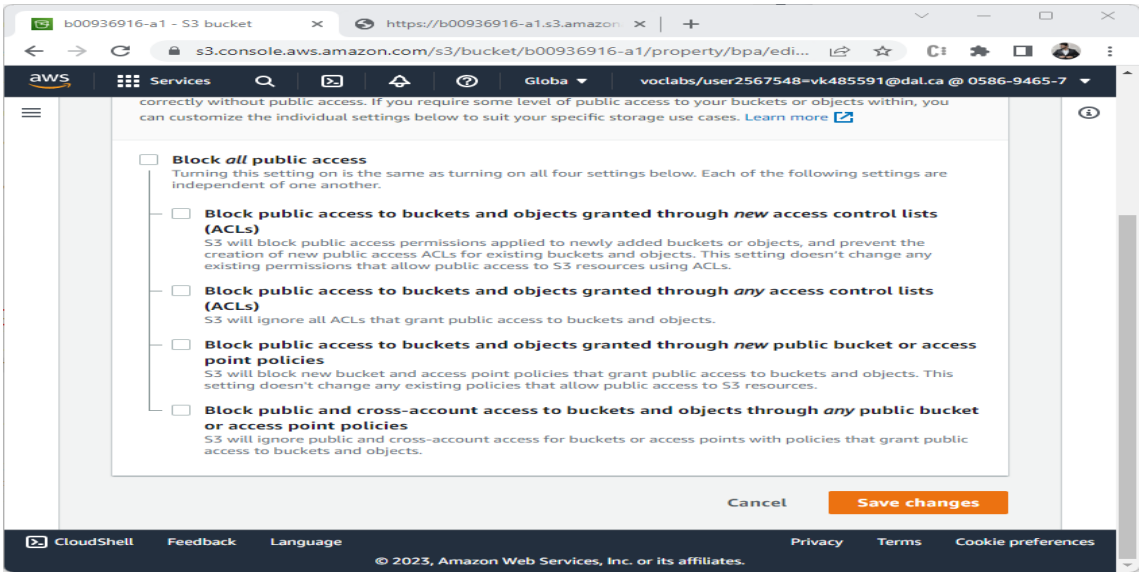

Figure 8.8.: ***De-selected***

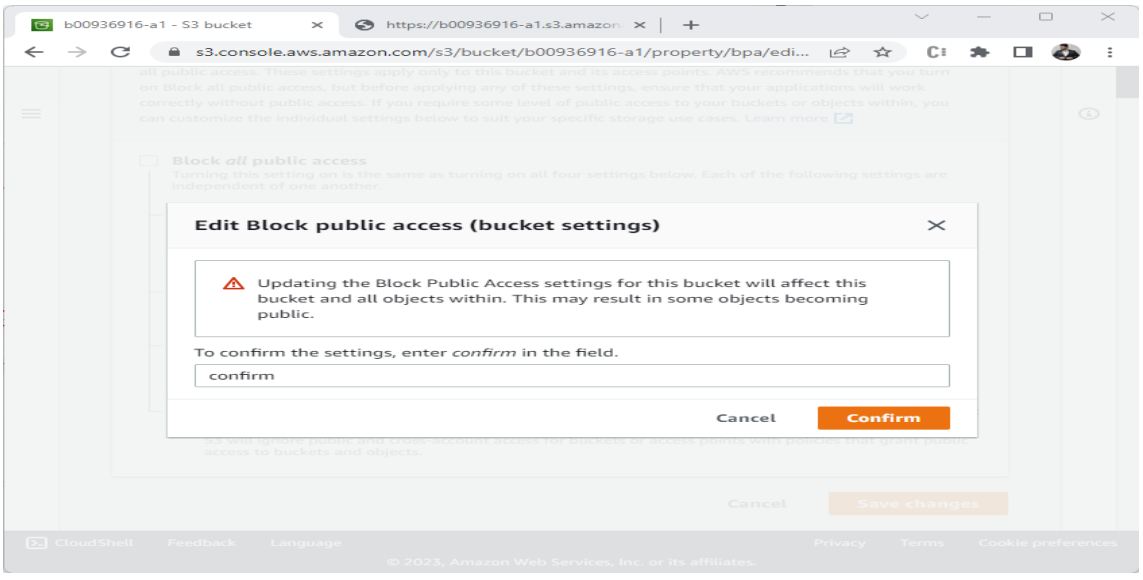

Figure 8.9.: ***Confirm the action***

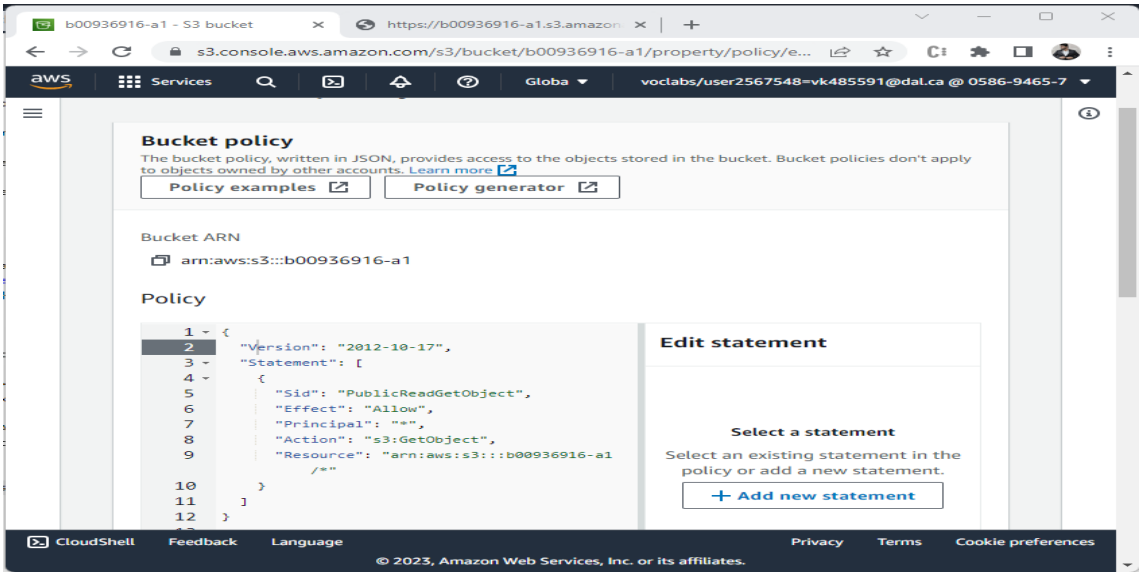

Figure 8.10.: ***Add bucket policy***

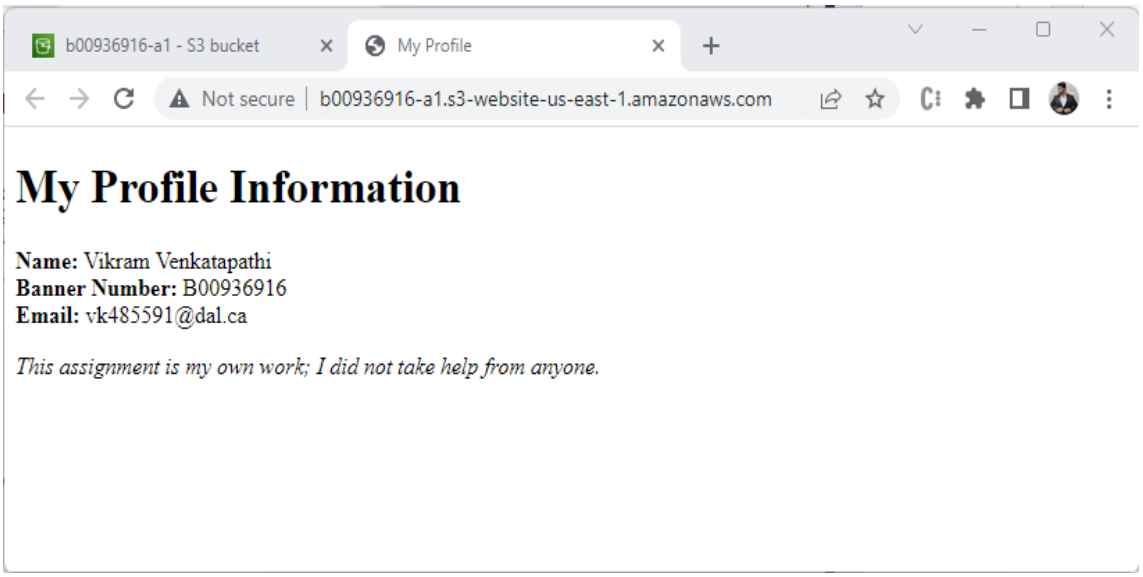

Figure 8.11.: ***Refresh the browser - Page is now visible***

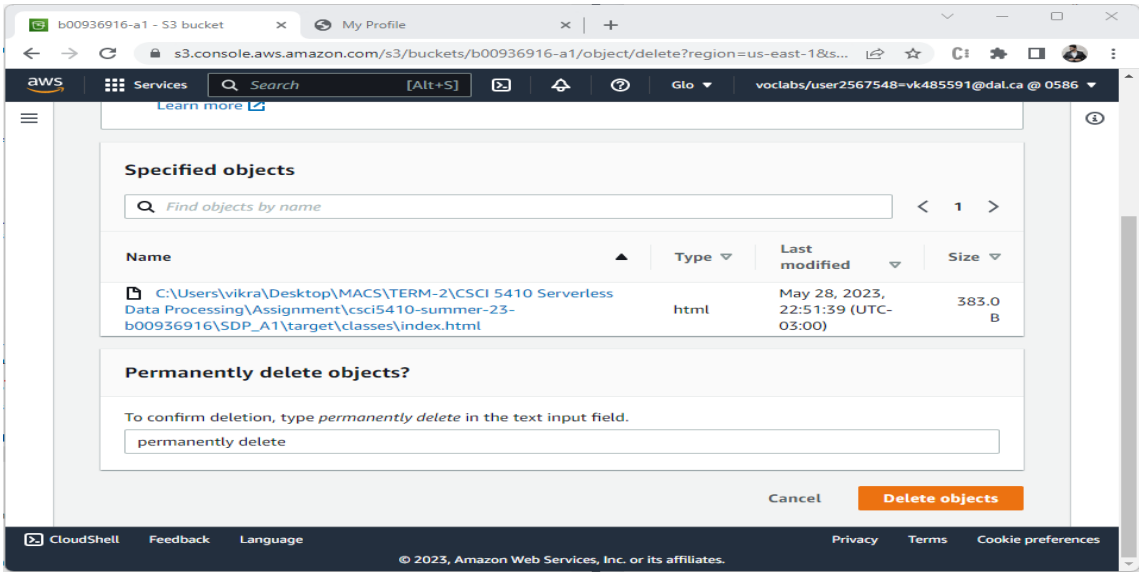

Figure 8.12.: ***Empty the bucket, after all tasks are done***

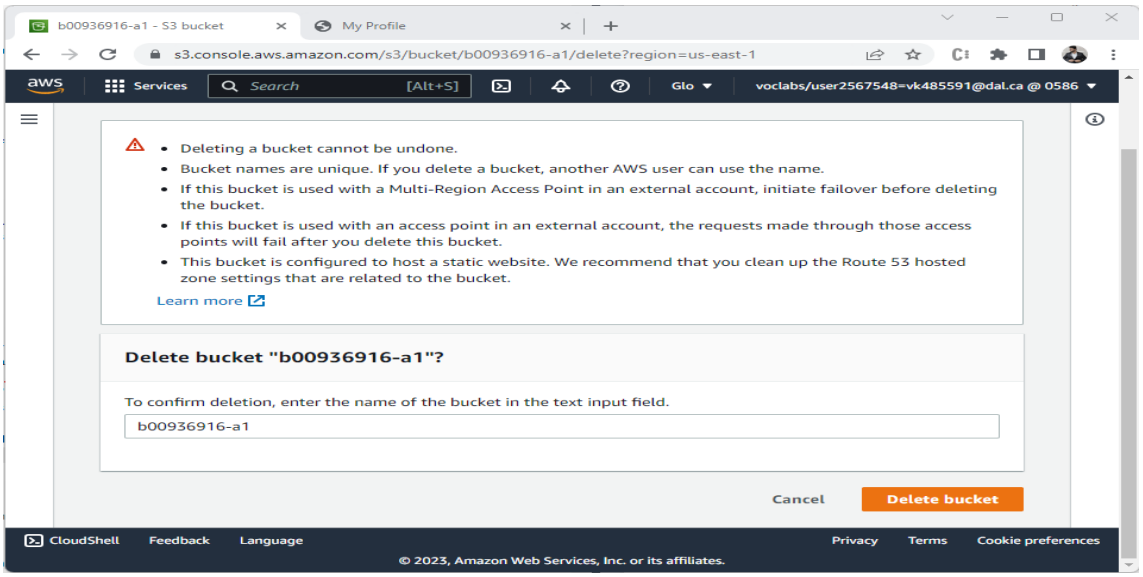

Figure 8.13.: ***Delete the bucket, Save resources and credits***

Part III.

REFERENCES

References

[1] P. Vahidinia, B. Farahani and F. S. Aliee, "Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach," *IEEE Internet of Things Journal*, doi: 10.1109/JIOT.2022.3165127.

[2] Eisenbarth, J.-P. (2010, July 3). "Inserting code in this LaTeX document with indentation". *Stack Overflow*. Available: https://stackoverflow.com/questions/3175105/inserting-code-in-this-latex-document-with-indentation. [Accessed 07 June 2023].

[3] "Getting two pictures to appear on the same page," *TeX Stack Exchange*, Available: https://tex.stackexchange.com/questions/134162/getting-two-pictures-to-appear-on-the-same-page. [Accessed 07 June 2023].

[4] "LaTeX Listings package JSON formatting," *TeX Stack Exchange*, Available: https://tex.stackexchange.com/questions/560830/latex-listings-package-json-formatting. [Accessed 07 June 2023].

[5] V. Venkatapathi, "B00936916_VikramVenkatapathi_A1_Report.pdf," *Dalhousie University*, Jan. 2023. [Online]. Available: https://git.cs.dal.ca/vikramv/csci5408_w23_b00936916_vikram_-venkatapathi/-/blob/main/Assignment_1/B00936916_VikramVenkatapathi_A1_Report.pdf. [Accessed 07 June 2023].

[6] Shanmuganathan, Vishakan. "OOAD-Project." *GitHub*, 2021, https://github.com/svishakan/OOAD-Project. [Accessed 07 June 2023].

[7] AWS SDK for Java Developer Guide., "Working with Amazon S3 Buckets".*Amazon Web Services, Inc., 2022* [Online]. Available: https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/examples-s3-buckets.html. [Accessed 07 June 2023].

[8] AWS SDK for Java Developer Guide., "Working with Amazon S3 Objects". *Amazon Web Services, Inc., 2022* [Online]. Available: https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/examples-s3-objects.html. [Accessed 07 June 2023].

[9] AWS Command Line Interface User Guide., "Configuration and Credential Files". *Amazon Web Services, Inc., 2022* [Online]. Available: https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html. [Accessed 07 June 2023].