

Assignment 1

Part B

Meet Patel (B00899516)

Dalhousie University

Subject

**CSCI 5410 (Serverless Data
Processing)**

Professor

Dr. Saurabh Dey

Link to the GitLab repository: https://git.cs.dal.ca/patel13/csci5410_b00899516_meet_patel.git

Link to the package: https://git.cs.dal.ca/patel13/csci5410_b00899516_meet_patel/-/tree/main/Assignment-1/a1_code/src/main/java/part_b_aws_s3

Flowchart showing the steps followed for performing the experiment

Figure 1 shows the steps to create **Amazon S3 bucket** and to **upload a txt file** to that bucket

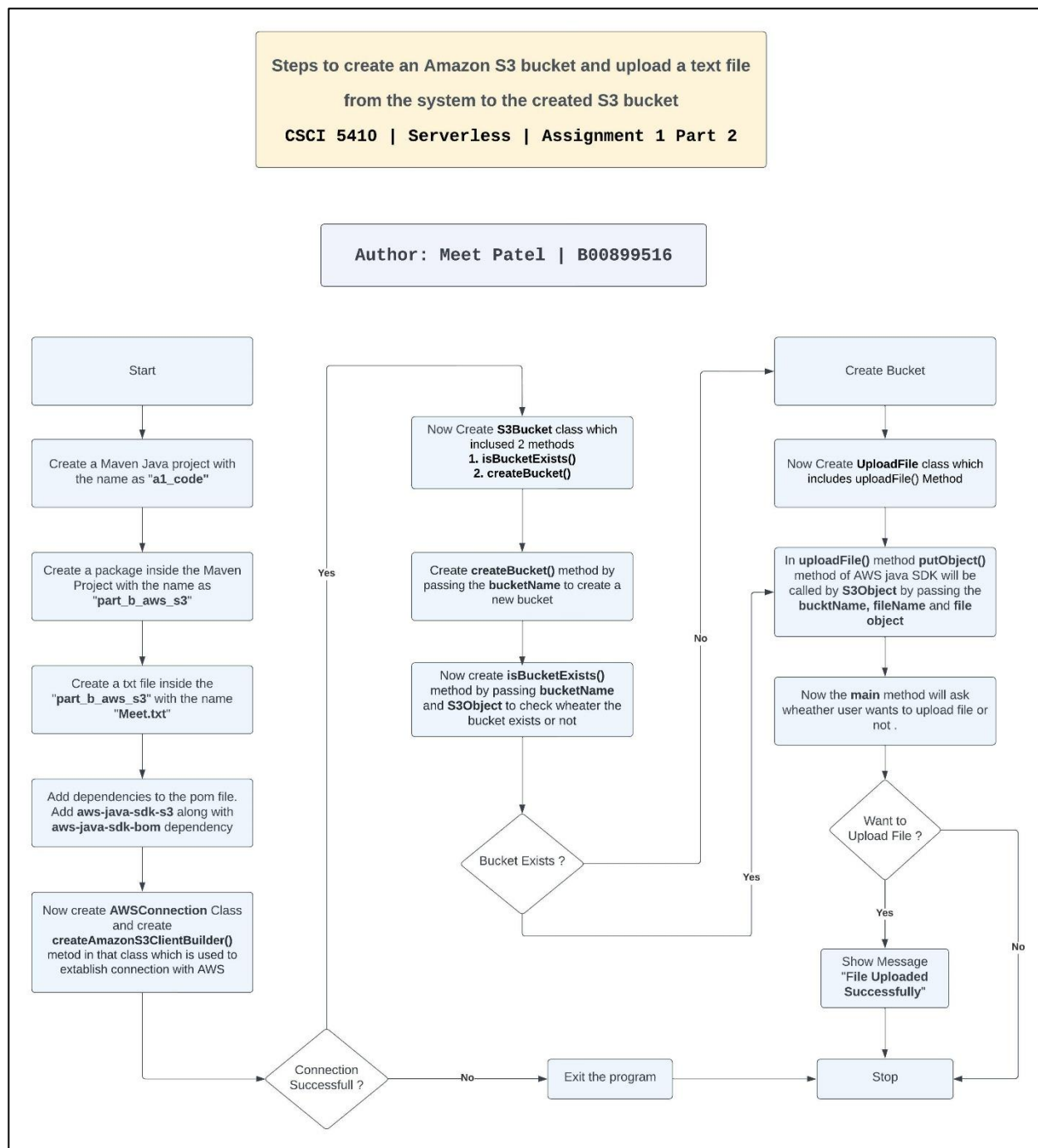


Figure 1: Steps followed for performing the experiment [1]

An observation of the AWS SDK for Java

AWS SDK for Java offers a variety of libraries that make it easy to make use of AWS services. It is open-source and highly intuitive. My observation of AWS SDK for java is that we can build a Java application that interacts with and executes numerous activities on Amazon S3, Amazon EC2, DynamoDB, and many other services in a couple of hours. There are only three requirements to use AWS SDK: proper JDE, AWS account and AWS credentials (Access key Id, Secret Access key and Session Token) to establish connection with the Amazon Web Services. A developer has access to a wide variety of inbuilt classes and methods for manipulating the services. However, the AWS SDK does not yet support all features. It's simple to utilize the built-in methods. For example, I used the `createBucket()` method, where I simply pass the bucket name and the bucket is created. Likewise, to upload a file to that bucket or any other existing bucket we can use build-in function named as `putObject()` which needs bucket name, file name and file as an argument. In addition, developer guide is also available which includes several examples for the usage of AWS services.

Screenshots of all the steps performed

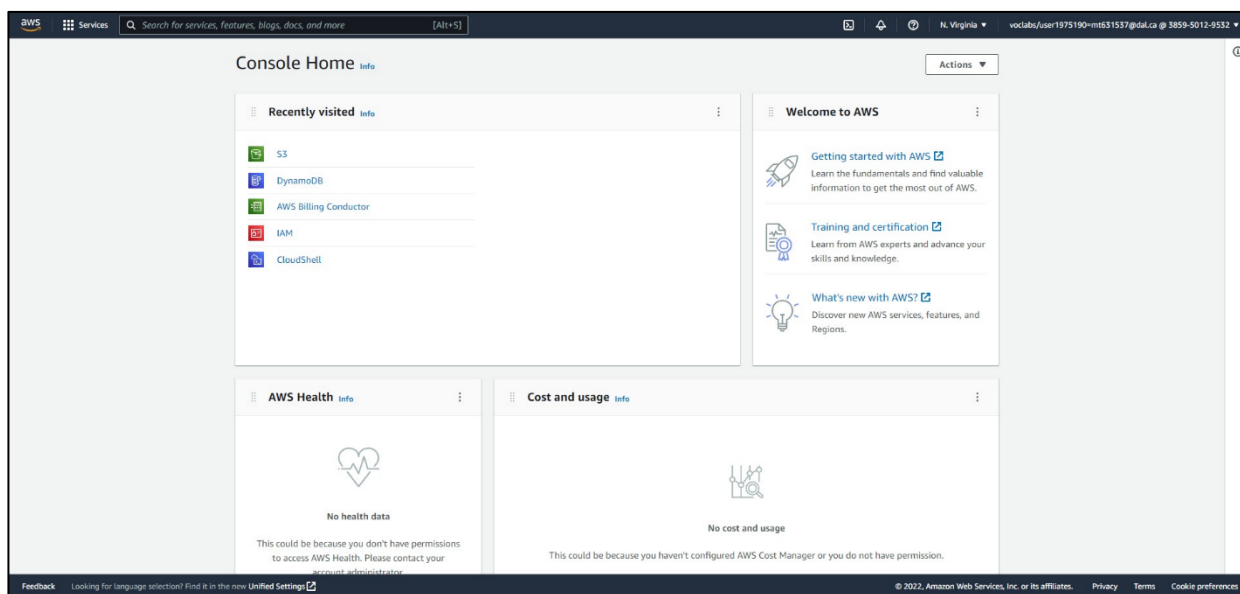


Figure 2: AWS Management Console

CSCI 5410: Assignment #1 (Part B) | Meet Patel B00899516

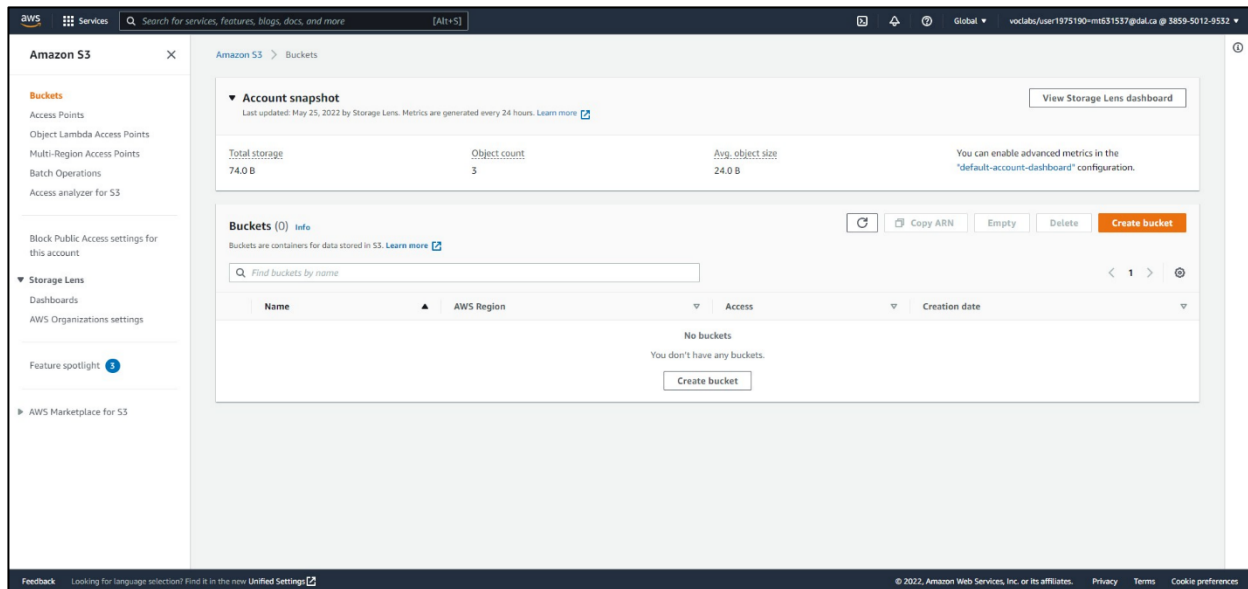


Figure 3: ScreenShot before creating Amazon S3 bucket

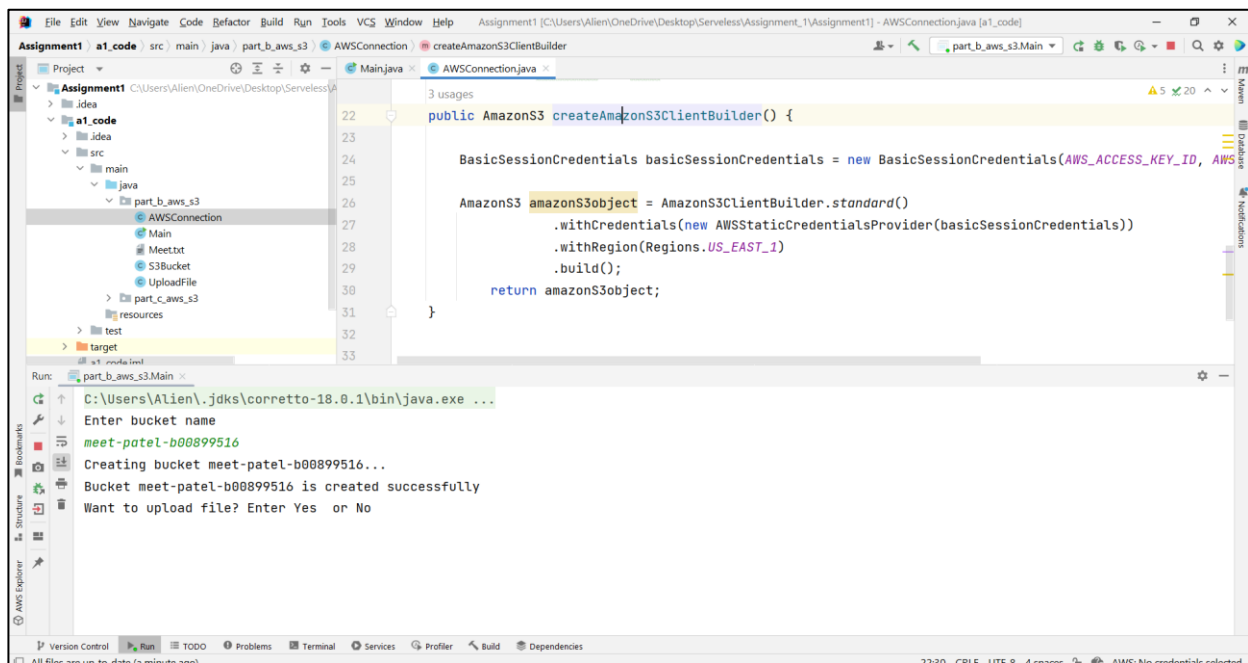


Figure 4: Screenshot of successful creation of the Amazon S3 bucket with name **meet-patel-b00899516**

CSCI 5410: Assignment #1 (Part B) | Meet Patel B00899516

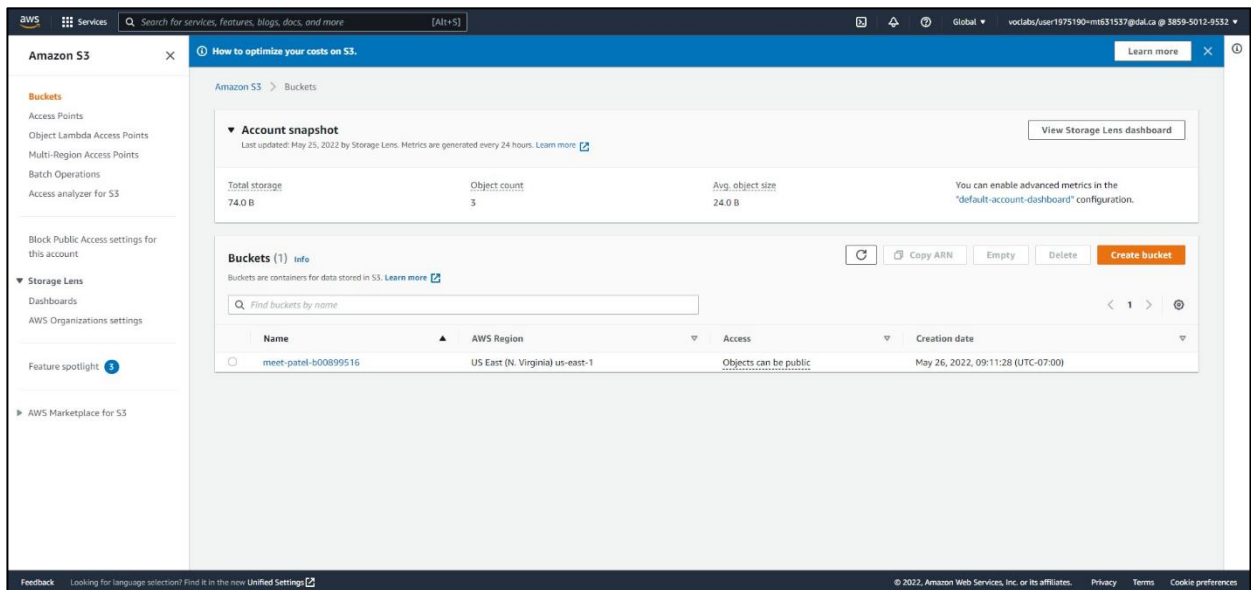


Figure 5: Screenshot shows successful creation of Amazon S3 bucket on Amazon S3

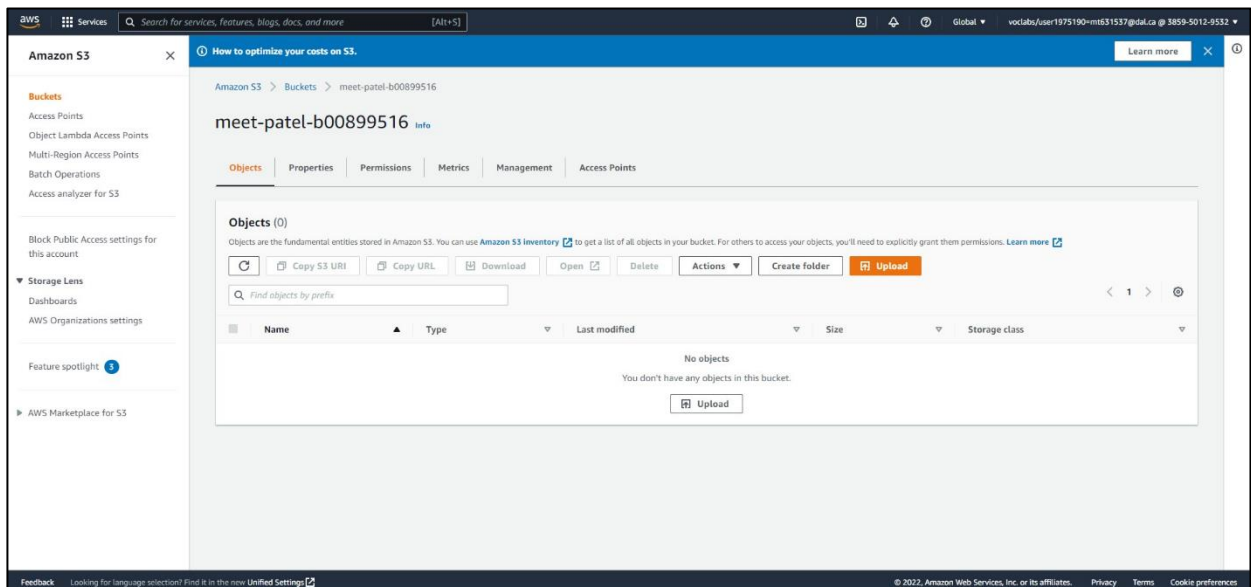
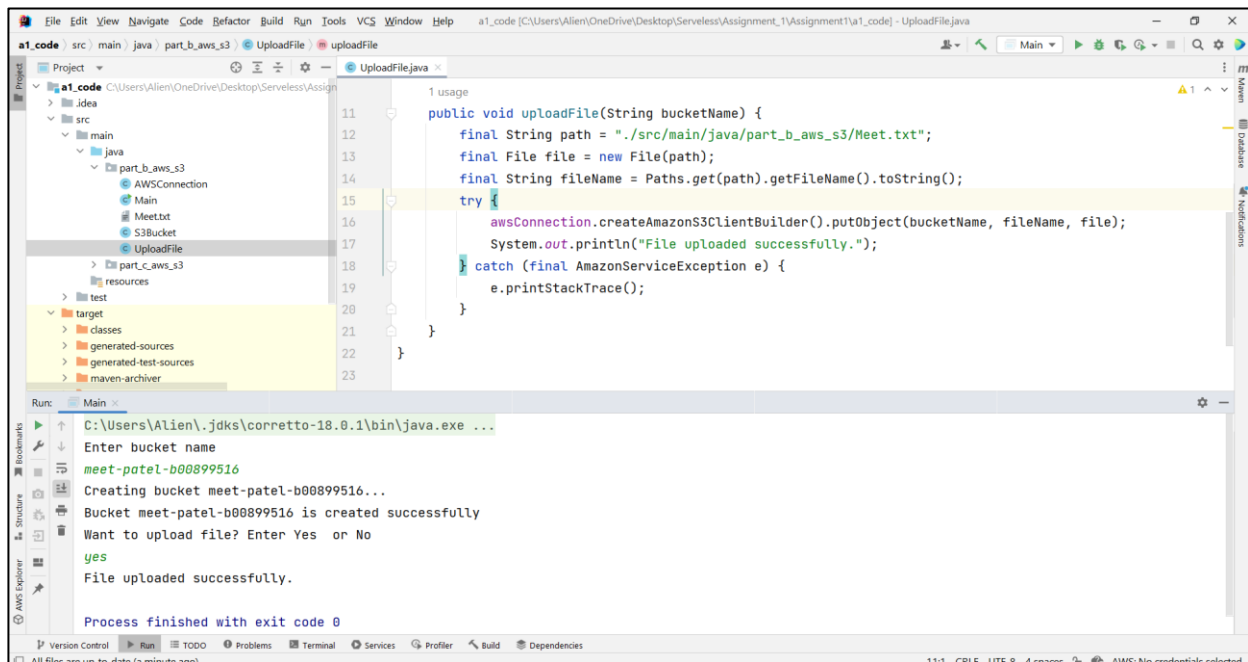


Figure 6: Screenshot shows the empty Amazon S3 bucket (Before uploading the text file in the bucket)

CSCI 5410: Assignment #1 (Part B) | Meet Patel B00899516



The screenshot shows an IDE window with a project named 'a1_code'. The 'UploadFile.java' file is open, showing the following code:

```
11 public void uploadFile(String bucketName) {
12     final String path = "./src/main/java/part_b_aws_s3/Meet.txt";
13     final File file = new File(path);
14     final String fileName = Paths.get(path).getFileName().toString();
15     try {
16         awsConnection.createAmazonS3ClientBuilder().putObject(bucketName, fileName, file);
17         System.out.println("File uploaded successfully.");
18     } catch (final AmazonServiceException e) {
19         e.printStackTrace();
20     }
21 }
```

The 'Run' console at the bottom shows the following output:

```
Enter bucket name
meet-patel-b00899516
Creating bucket meet-patel-b00899516...
Bucket meet-patel-b00899516 is created successfully
Want to upload file? Enter Yes or No
yes
File uploaded successfully.
Process finished with exit code 0
```

Figure 7: Screenshot shows the successful upload of the text file with the name **Meet.txt**

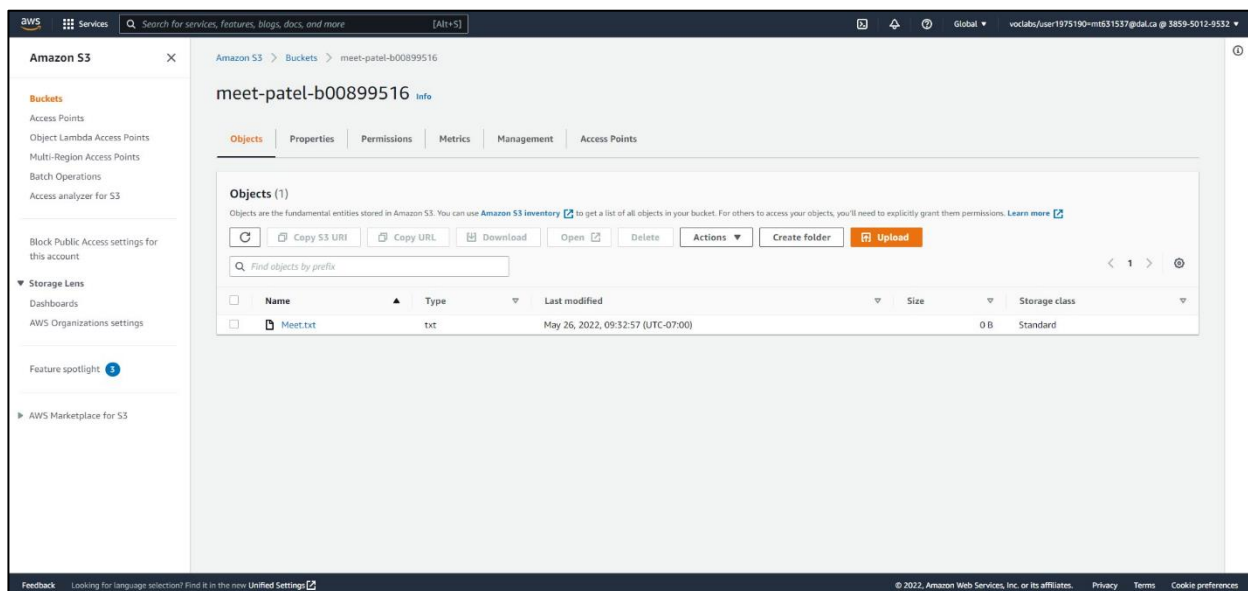


Figure 8: Screenshot shows the successful upload of the **Meet.txt** text file to the **meet-patel-b00899516** bucket

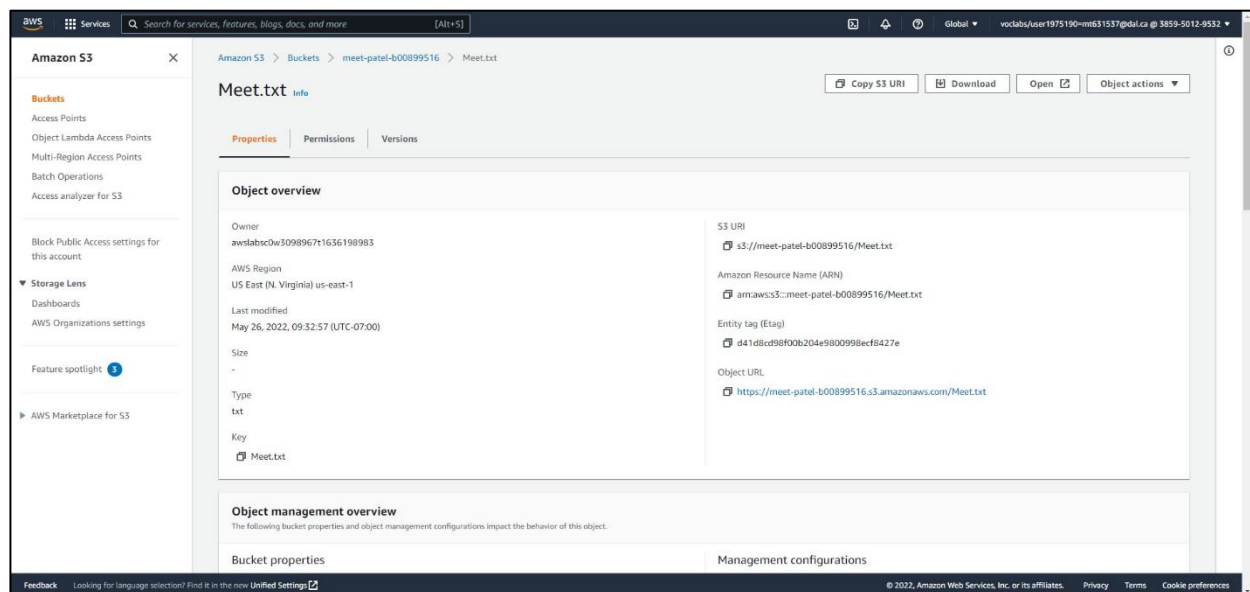


Figure 9: Screenshot shows the properties of the text file **Meet.txt** which is uploaded to the **meet-patel-b00899516** bucket

Program Script

AWSConnection.java

This java class is used to establish connection with the **Amazon S3 service**

```
package part_b_aws_s3;
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 AWSConnection {
    private static final String AWS_ACCESS_KEY_ID = "ASIAVTXDLTV6F6HW5576";
    private static final String AWS_SECRET_ACCESS_KEY = "mose7hvZ6Pnr6B4VKJDPZJH56PJkXZij7F0WEBNb";
    private static final String AWS_SESSION_TOKEN = "FwoGZXIvYXZdENH////////wEaDFM6oYlUicxLePbaSCLAASO/" +
        "nvZcYU8Rwrd38cOvDsFsiRa4TPYckIopzwHjddq7A+9RwgLaTC/a2edG/K8RBOMdGlexzbPSkxJ2hDXTBHDWCumORszjrrf" +
        "ACpRNQty/bn3sR6h6B/42OW/Ie7R/fPUMevOWklqiKf4x/dTEXgDKfyqSCgq3jCgSkgpMDEHiVekpsz1WWFjD5IJ6oa1TcnbB" +
        "xFgVsTBrw+D0GXrEAGDCEDFrq448xhwgupiMqVI1EScG7/mUo01TuLZiAJ6XSjgx76UBjItV9IrC2T50fMfEFcMtpS290AnEewg"
+
        "79IWDEhdbbhwWELeScIggKNoORjmr9II";
    public AmazonS3 createAmazonS3ClientBuilder() {

        BasicSessionCredentials basicSessionCredentials = new BasicSessionCredentials(AWS_ACCESS_KEY_ID,
AWS_SECRET_ACCESS_KEY, AWS_SESSION_TOKEN);

        AmazonS3 amazonS3Object = AmazonS3ClientBuilder.standard()
            .withCredentials(new AWSStaticCredentialsProvider(basicSessionCredentials))
            .withRegion(Regions.US_EAST_1)
            .build();
        return amazonS3Object;
    }
}
```

S3Bucket.java

This java class is used to create **Amazon S3 bucket** using **createBucket ()** built method. Before creating bucket this also checks whether bucket already exists or not.

```
package part_b_aws_s3;

import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.Bucket;

import java.util.List;

public class S3Bucket {
    public boolean isBucketExists(String bucketName, AmazonS3 amazonS3ClientBuilder){
        List<Bucket> bucketList = amazonS3ClientBuilder.listBuckets();
        for (Bucket bucket: bucketList){
            if(bucket.getName().equals(bucketName)){
                return true;
            }
        }
        return false;
    }

    public void createBucket(String bucketName){
        AWSConnection awsConnection = new AWSConnection();
        if (isBucketExists(bucketName, awsConnection.createAmazonS3ClientBuilder())){
            System.out.println("Bucket "+bucketName+ " already exists....");
        }else {
            try {
                System.out.println("Creating bucket " + bucketName + "...");
                awsConnection.createAmazonS3ClientBuilder().createBucket(bucketName);
                System.out.println("Bucket "+bucketName+ " is created successfully");
            } catch (AmazonS3Exception e){
                System.err.println(e.getErrorMessage());
            }
        }
    }
}
```


UploadFile.java

This java class is used to upload the text file **Meet.txt** to the **Amazon S3 bucket** using **putObject ()** method

```
package part_b_aws_s3;

import com.amazonaws.AmazonServiceException;

import java.io.File;
import java.nio.file.Paths;

public class UploadFile {
    AWSConnection awsConnection = new AWSConnection();
    public void uploadFile(String bucketName) {
        final String path = "./src/main/java/part_b_aws_s3/Meet.txt";
        final File file = new File(path);
        final String fileName = Paths.get(path).getFileName().toString();
        try {
            awsConnection.createAmazonS3ClientBuilder().putObject(bucketName, fileName, file);
            System.out.println("File uploaded successfully.");
        } catch (final AmazonServiceException e) {
            e.printStackTrace();
        }
    }
}
```

Main.java

This is the main method is used to which takes the bucket name from the user and then pass it to the **create bucket** method. It also asks user whether you user wants to upload file or not. If user wants to add file to the bucket, then it calls the **upload file** method to upload the file

```
package part_b_aws_s3;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String bucketName = null;
        while (isNull(bucketName) || isEmpty(bucketName)) {
            System.out.println("Enter bucket name");
            bucketName = scanner.nextLine();
        }
        S3Bucket s3Bucket = new S3Bucket();
        s3Bucket.createBucket(bucketName);

        String input;
        System.out.println("Want to upload file? Enter Yes or No");
        input = scanner.nextLine();
        if (input.equalsIgnoreCase("Yes")) {
            UploadFile uploadFile = new UploadFile();
            uploadFile.uploadFile(bucketName);
        } else if (input.equalsIgnoreCase("No")) {
            System.exit(0);
        }
    }

    private static boolean isEmpty(String bucketName) {
        return bucketName.trim().isEmpty();
    }

    private static boolean isNull(String bucketName) {
        return bucketName == null;
    }
}
```

References

- [1] "Intelligent Diagramming | Lucidchart," *Lucidchart*, 2022. [Online]. Available: <https://www.lucidchart.com/> [Accessed: May 26, 2022].
- [2] A. W. Services, "AWS SDK for Java Documentation," *Amazon*, 2022. [Online]. Available: <https://docs.aws.amazon.com/sdk-for-java/index.html> [Accessed: May 26, 2022].