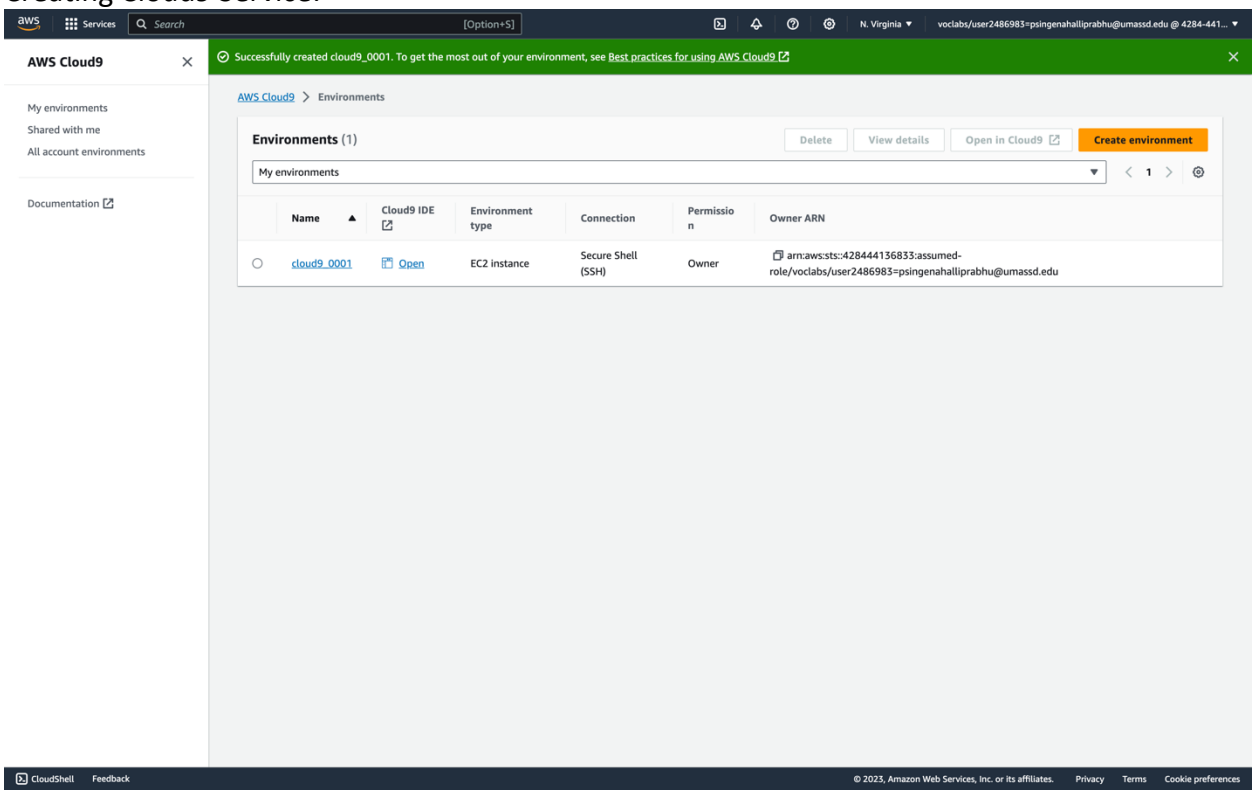
PRADYOTH SINGENAHALLI PRABHU
02071847
psingenahalliprabhu@umassd.edu

CIS 602: Big Data – Homework 1

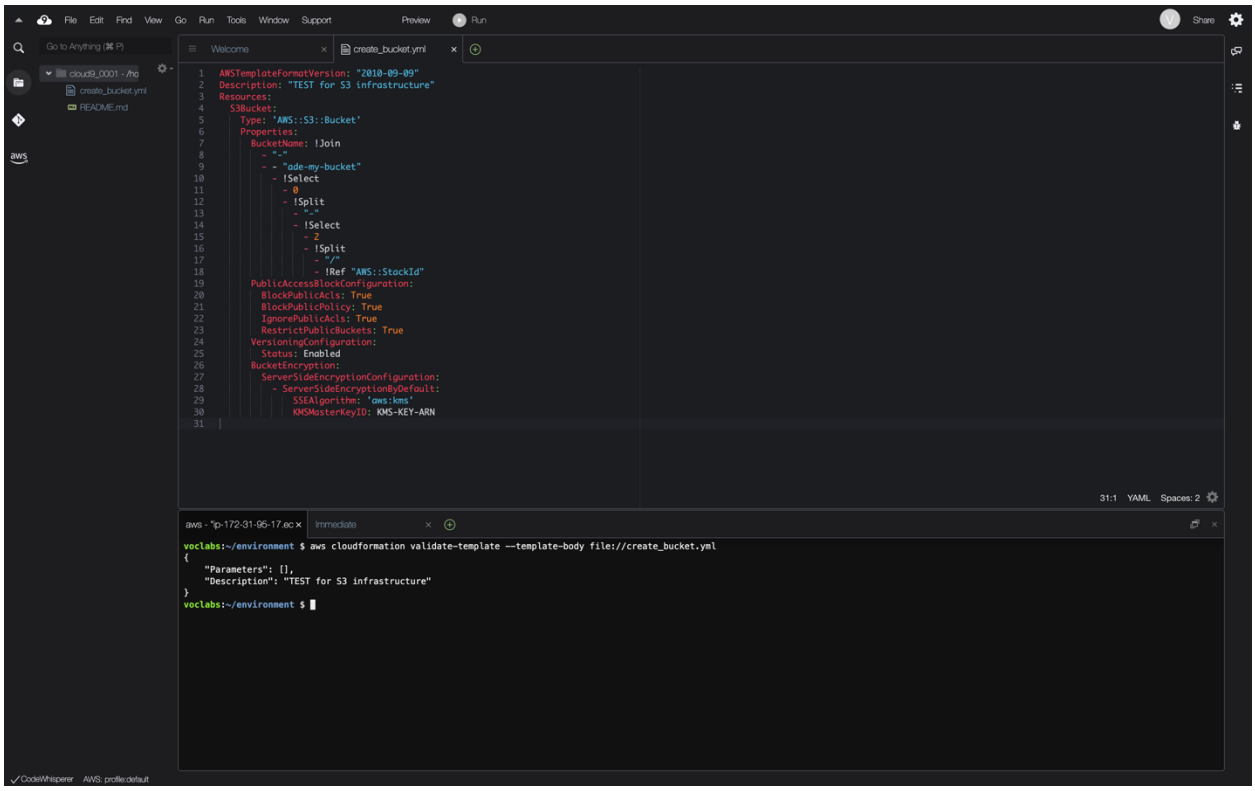
Task 1: Creating a CloudFormation template and stack

1. Creating Cloud9 Service:



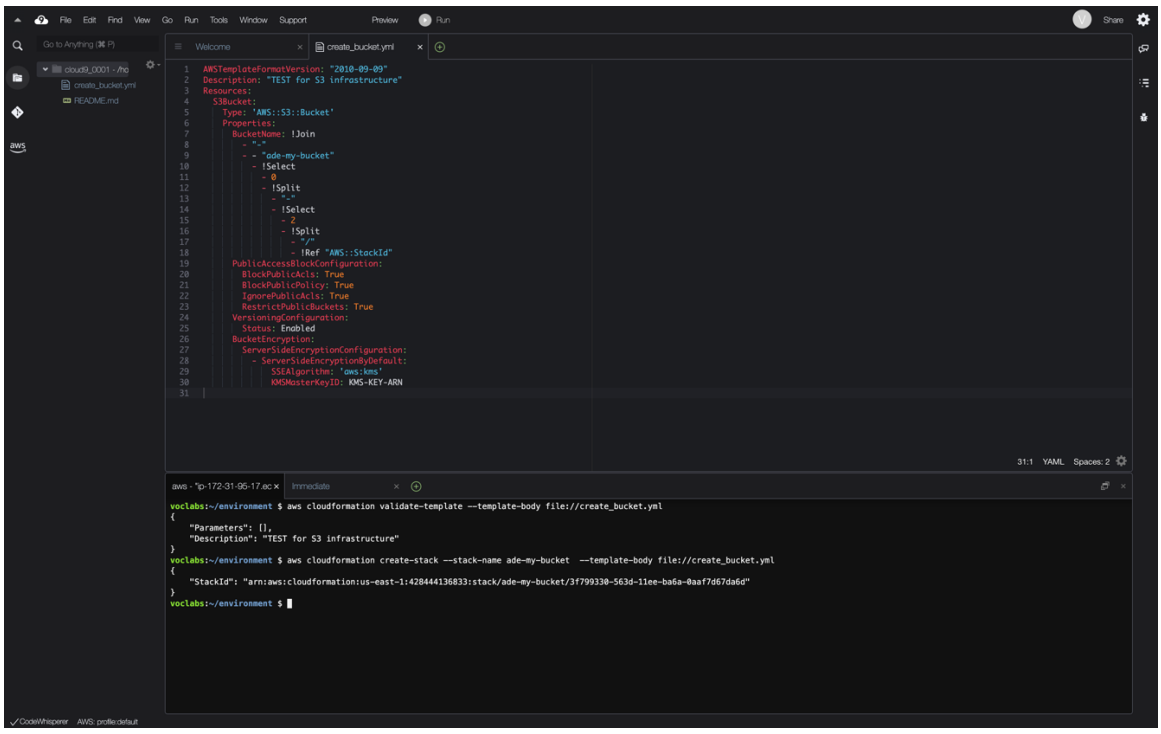
2. Validating the CloudFormation template:
   Used command:

```
aws cloudformation validate-template --template-body file://create_bucket.yml
```



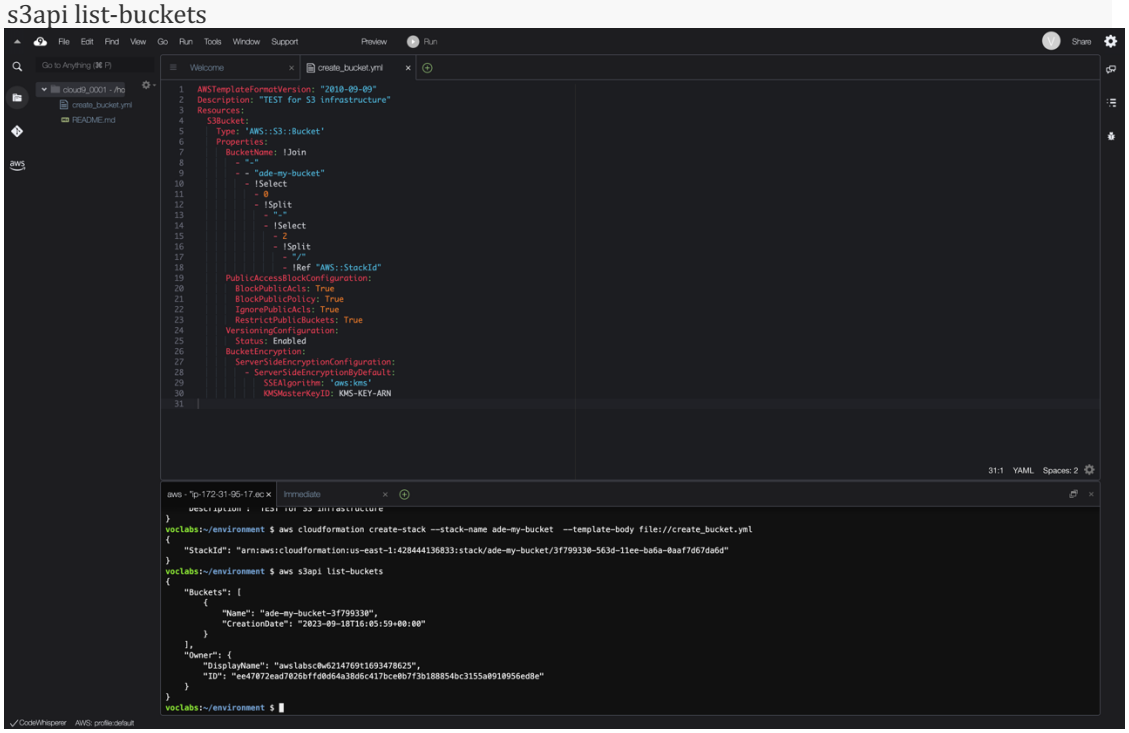
3. Creating the stack:
   Created stack using the following command:

```
aws cloudformation create-stack --stack-name ade-my-bucket --template-body file://create_bucket.yml
```



4. Verifying stack:
   Verified stack created using the command:

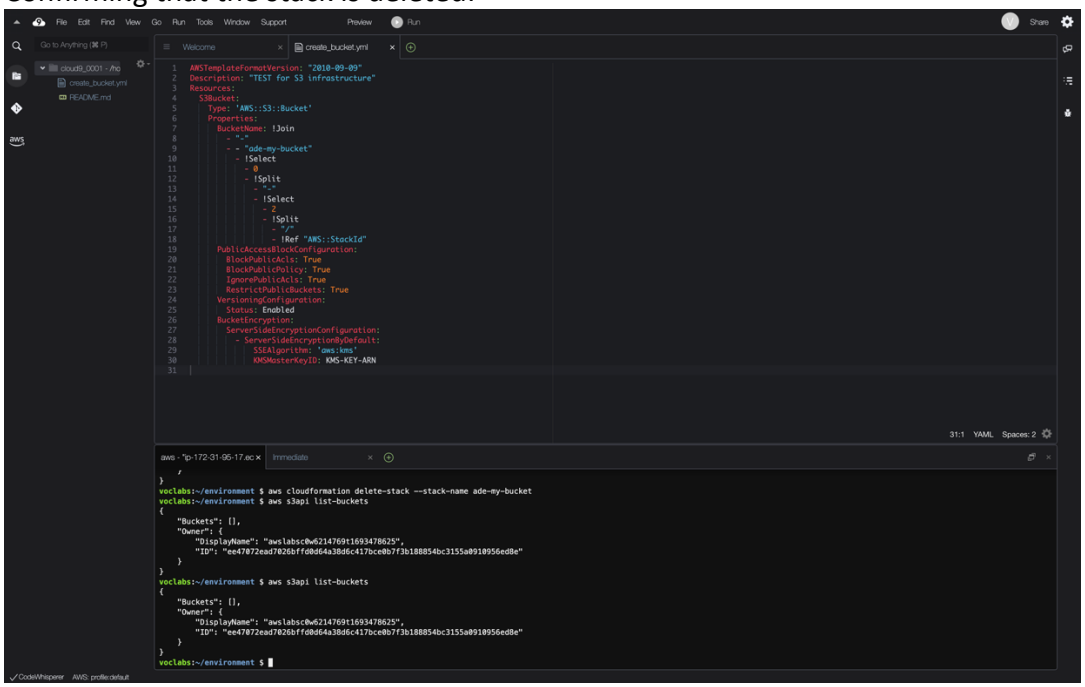
```
aws s3api list-buckets
```



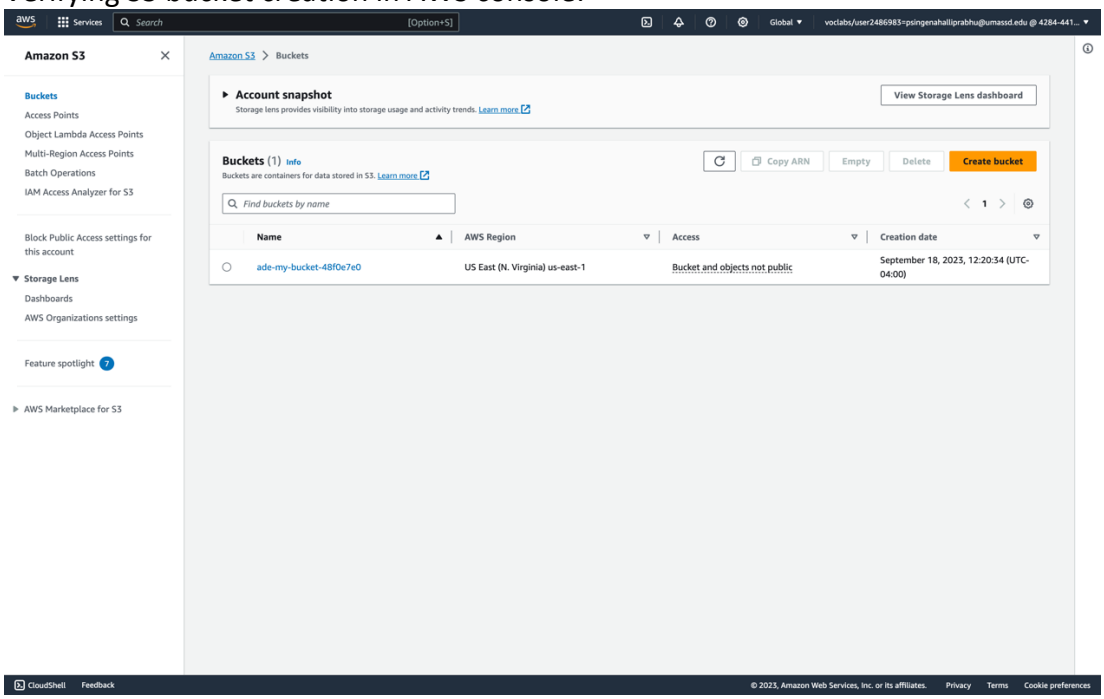
5. Deleting the stack
   We can delete the stack using the following the command:

```
aws cloudformation delete-stack --stack-name ade-my-bucket
```

   Confirming that the stack is deleted:

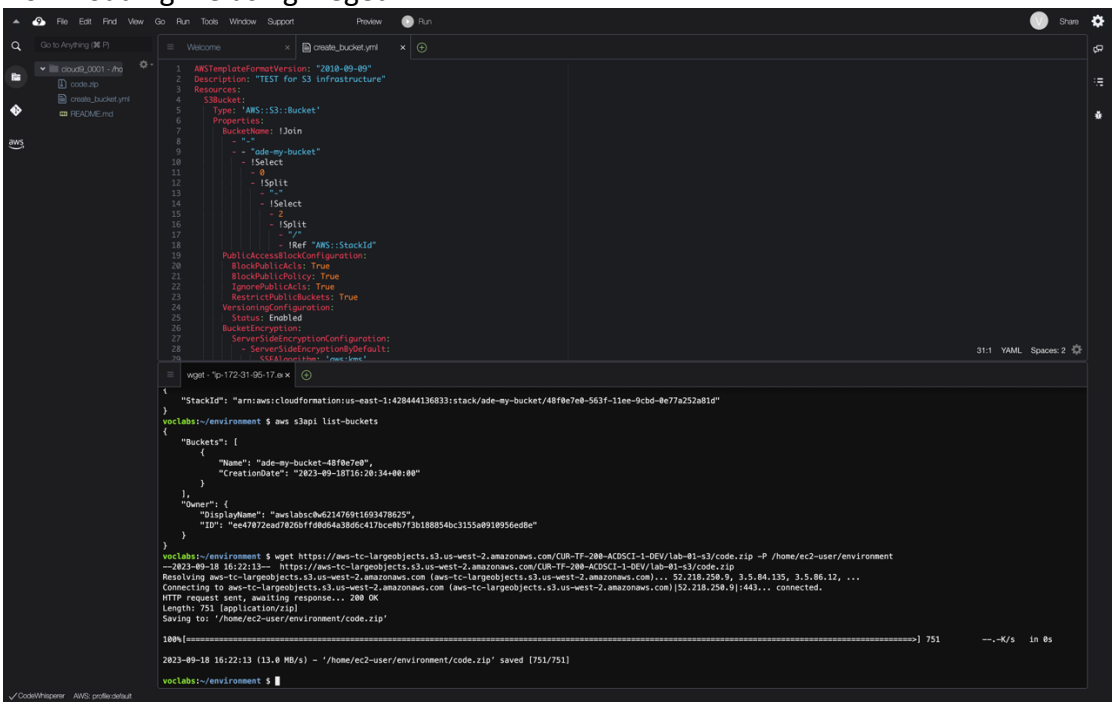


6. Verifying S3 bucket:
   Verifying S3 bucket creation in AWS console:

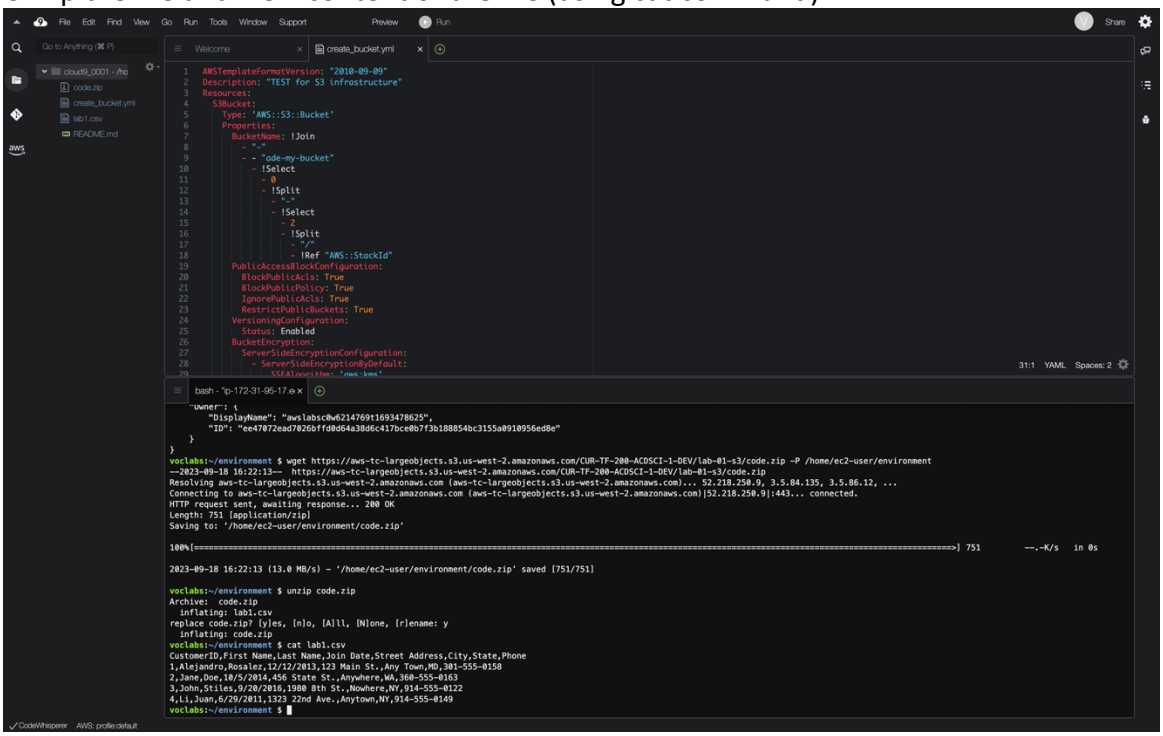


Task 2: Uploading sample data to an S3 bucket

1. Downloading file using weget:

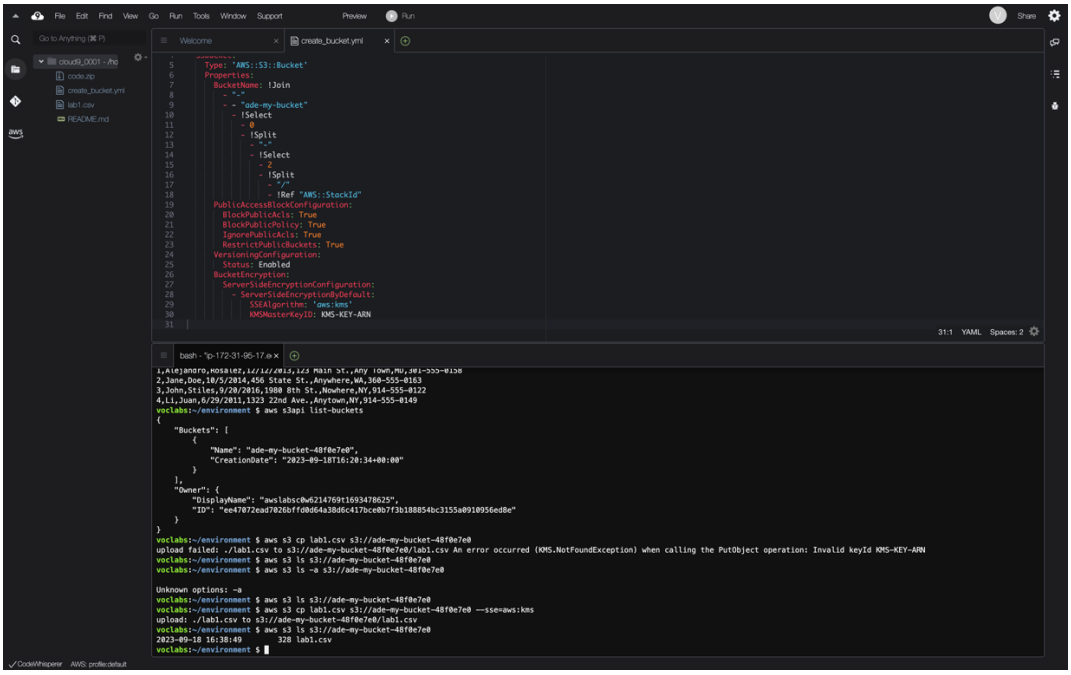


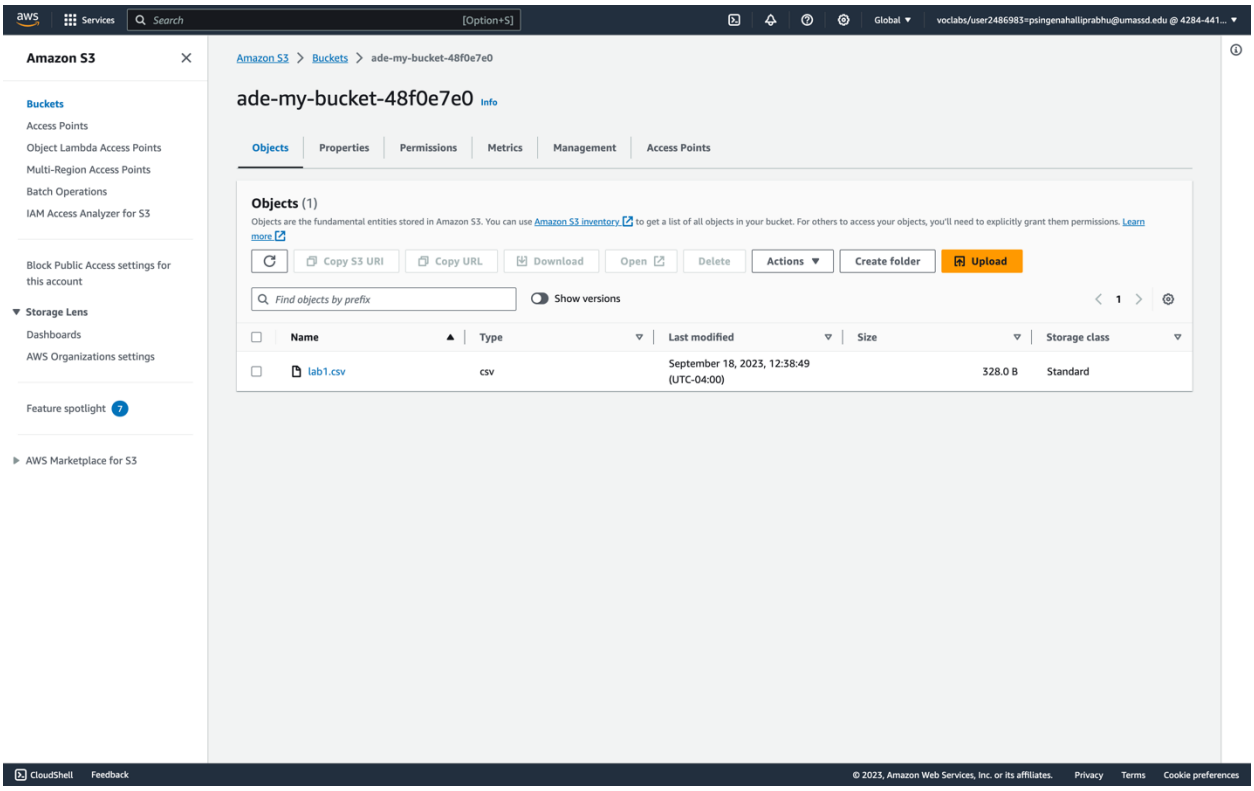
Unzip the file and view content of the file (using cat command):



2. Copy the data into the S3 bucket.
   Used command: aws s3 cp lab1.csv s3://ade-my-bucket-48f0e7e0

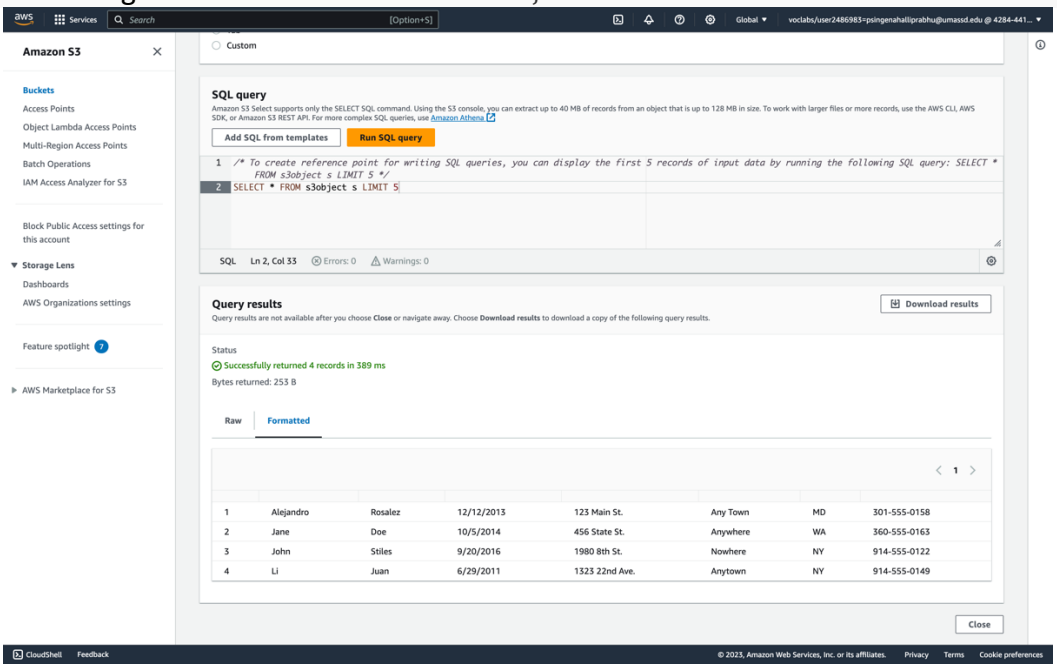
   Confirming if the file added to the S3 file:
   aws s3 ls s3://ade-my-bucket-48f0e7e0



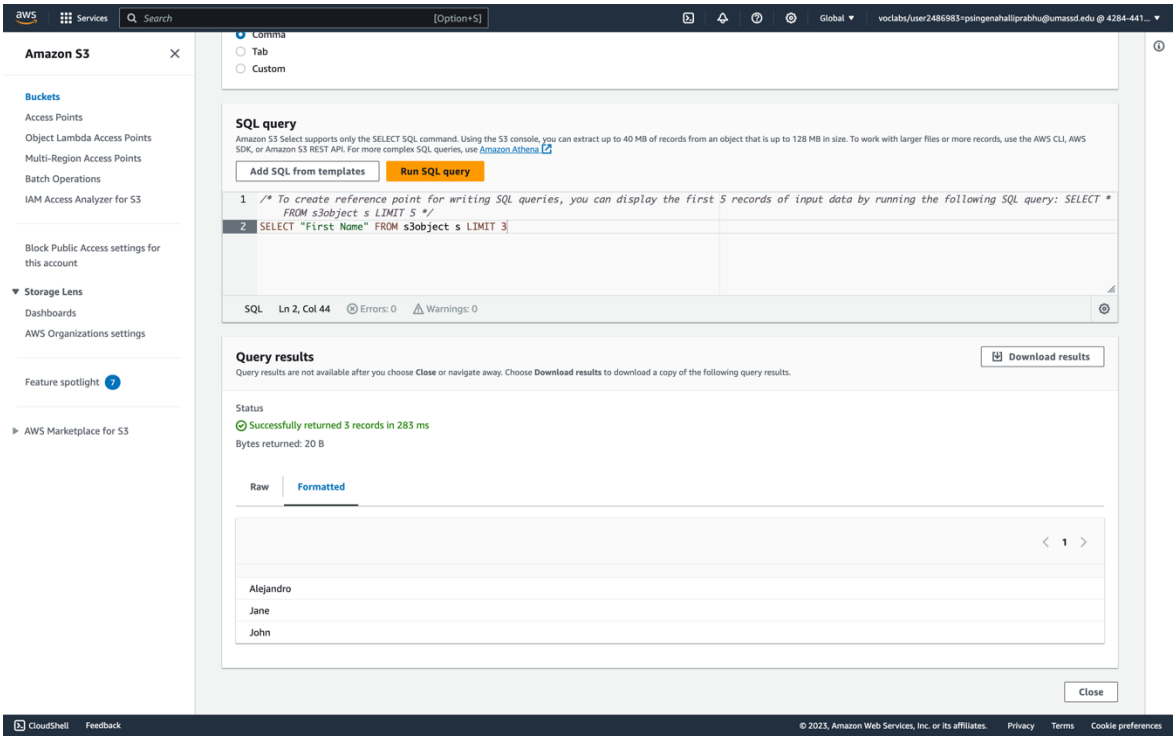


Task 3: Querying the data

1. Using the S3 Select query to run a SQL query on the uploaded data.
   Choosing Object actions > Query with S3 Select for the file lab.csv
   Running command: SELECT * FROM s3object s LIMIT 5

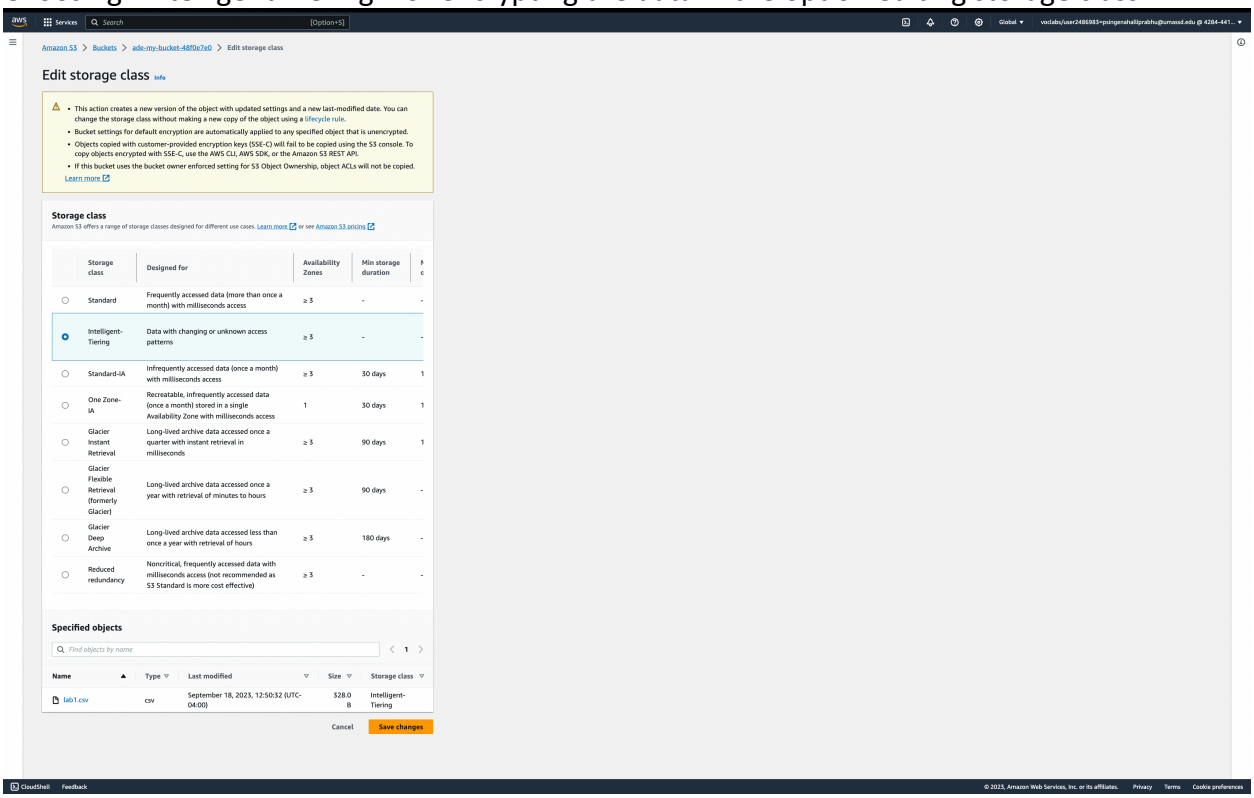


Running command: SELECT "First Name" FROM s3object s LIMIT 3



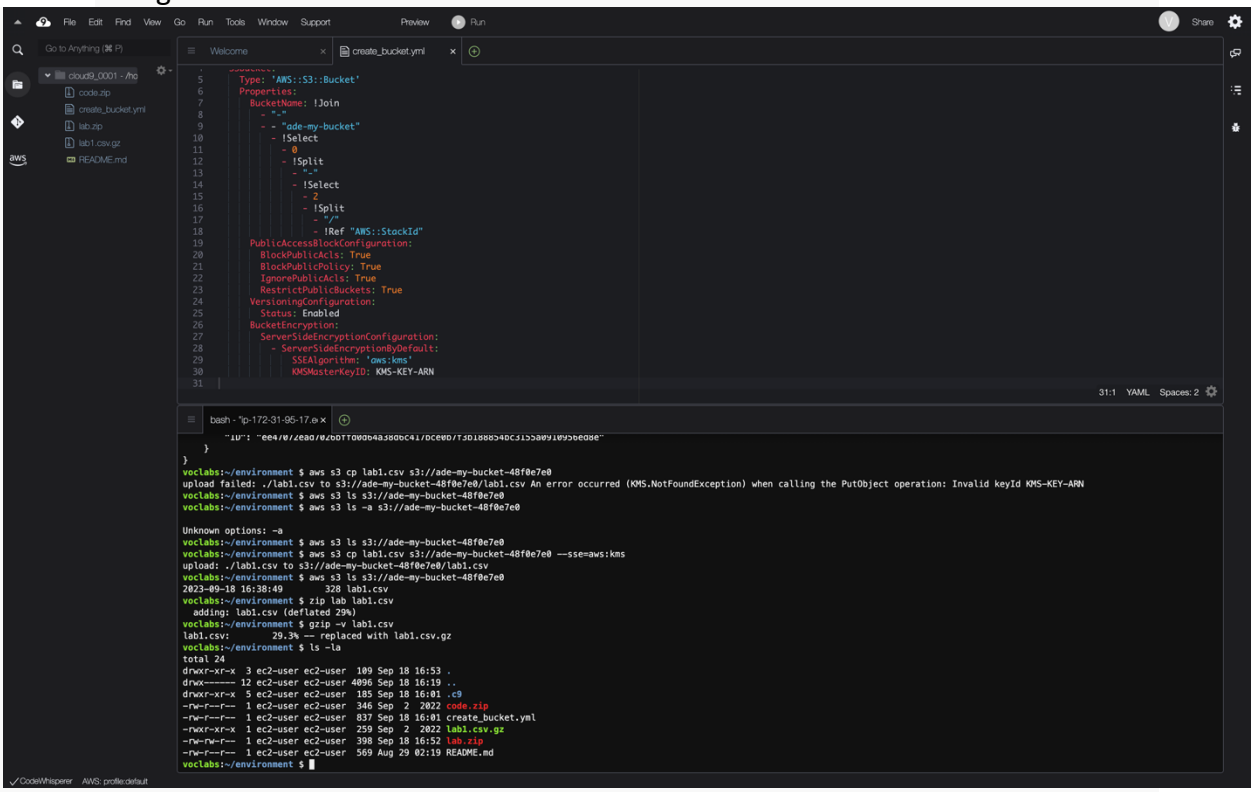
Task 4: Modifying an object's encryption properties and storage type

Choosing “Intelligent-Tiering” for encrypting the data in the option editing storage class.

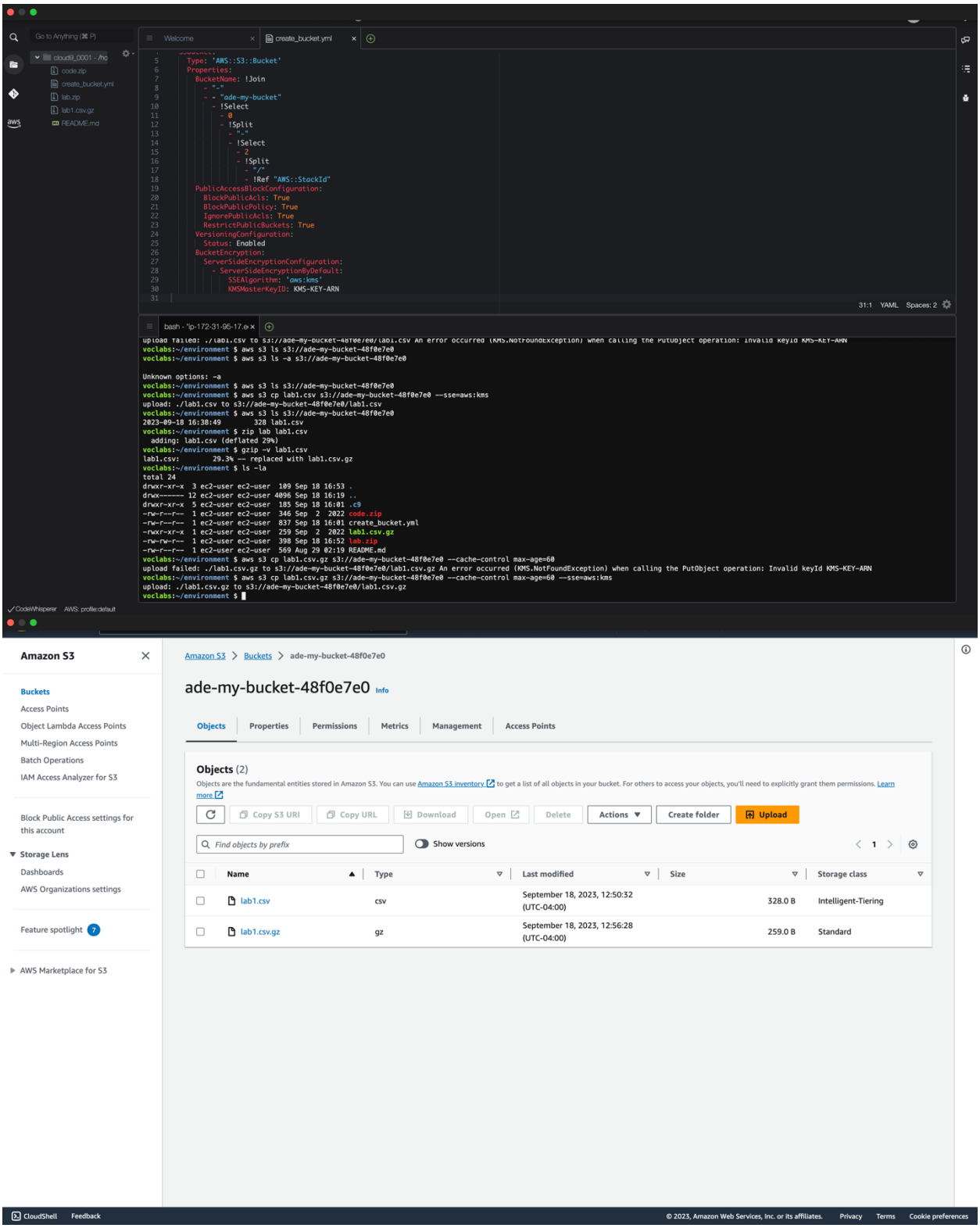


Task 5: Compressing and querying the dataset

1. Compressing the file
    a. To compress the file, converting into zip format:
    Using the command: zip lab lab1.csv
    b. Compressing the file with the GZIP format:
    Using the command: gzip -v lab1.csv
    c. Listing the object in the directory:
    Using the command: ls -la

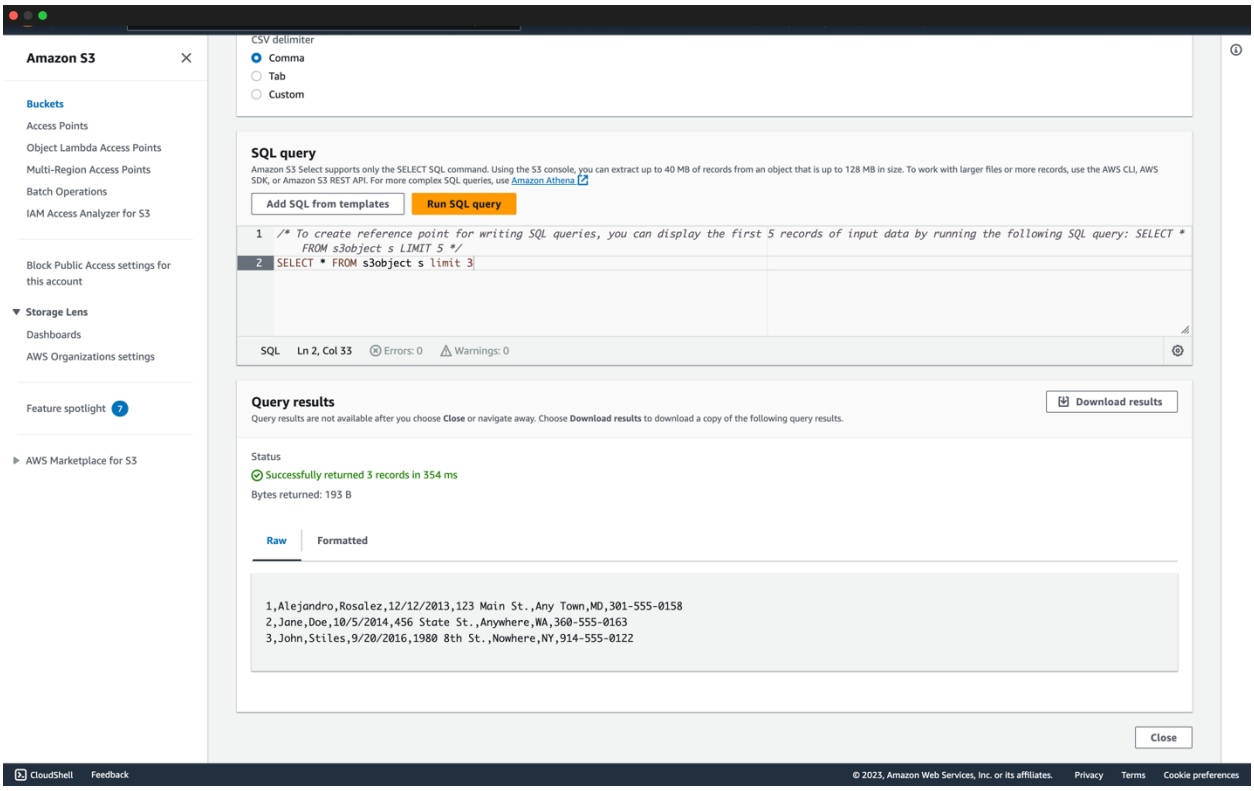


2. Uploading the GZIP version file to S3 bucket:
    Using the command:
    aws s3 cp lab1.csv.gz s3://ade-my-bucket-48f0e7e0 --cache-control max-age=60 --sse=aws:kms



3. Running the S3 Query on the compressed file:

   Running the command: `SELECT * FROM s3object s limit 3`



Conclusion Remarks:

In conclusion, this report provides a summary of our project's key achievements, highlighting the important lessons learned and tasks completed. Firstly, we became proficient in accessing AWS Cloud9, a valuable skill for modern technology environments. We automated S3 bucket creation using YAML scripts and AWS CLI commands, enhancing the efficiency of our infrastructure setup. We ensured stack reliability by verifying its status through AWS CLI.

We also learnt how to delete the stack using AWS CLI. With the ability to remove stacks through the command-line interface, we not only ensured the elimination of unnecessary resources but also demonstrated our proficiency in managing cloud-based systems effectively. We acquired essential data acquisition skills by downloading files using wget. Subsequently, we smoothly uploaded these files to S3 buckets, ensuring data accessibility and integrity.

Our proficiency with AWS CLI allowed us to list files in S3 buckets efficiently, simplifying data management. Complex data queries were executed using S3 Select, including tasks like retrieving specific data fields and limiting results.

Data security was addressed through encryption using 'Intelligent-Tiering'. This proactive measure played a pivotal role in safeguarding sensitive information and ensuring the confidentiality and integrity of our data assets. By employing this encryption method, we not only protected our data from unauthorized access but also optimized our data storage costs. The intelligent tiering system dynamically moves data between different storage classes, such as

Standard and Glacier, depending on its access patterns. This means that frequently accessed data is kept in more expensive but faster storage, while less frequently accessed data is moved to more cost-effective storage options.

Furthermore, we optimized data storage and transfer by compressing files with 'gzip' and successfully uploaded compressed gzip files to S3 buckets. Lastly, we demonstrated versatility by running S3 queries on compressed data.