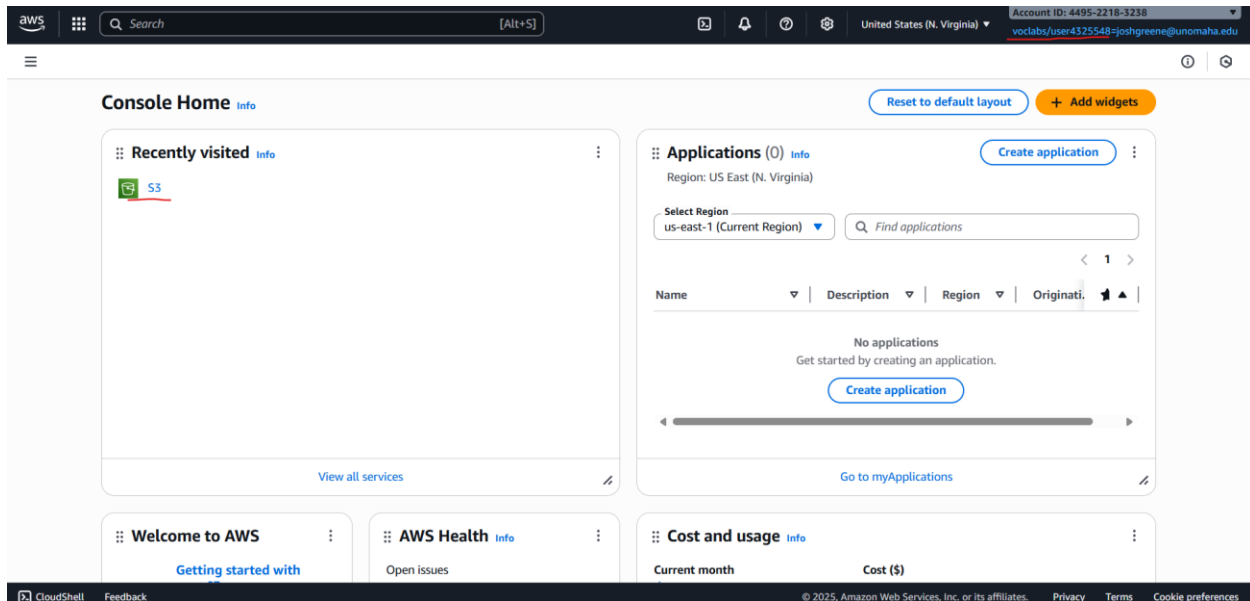
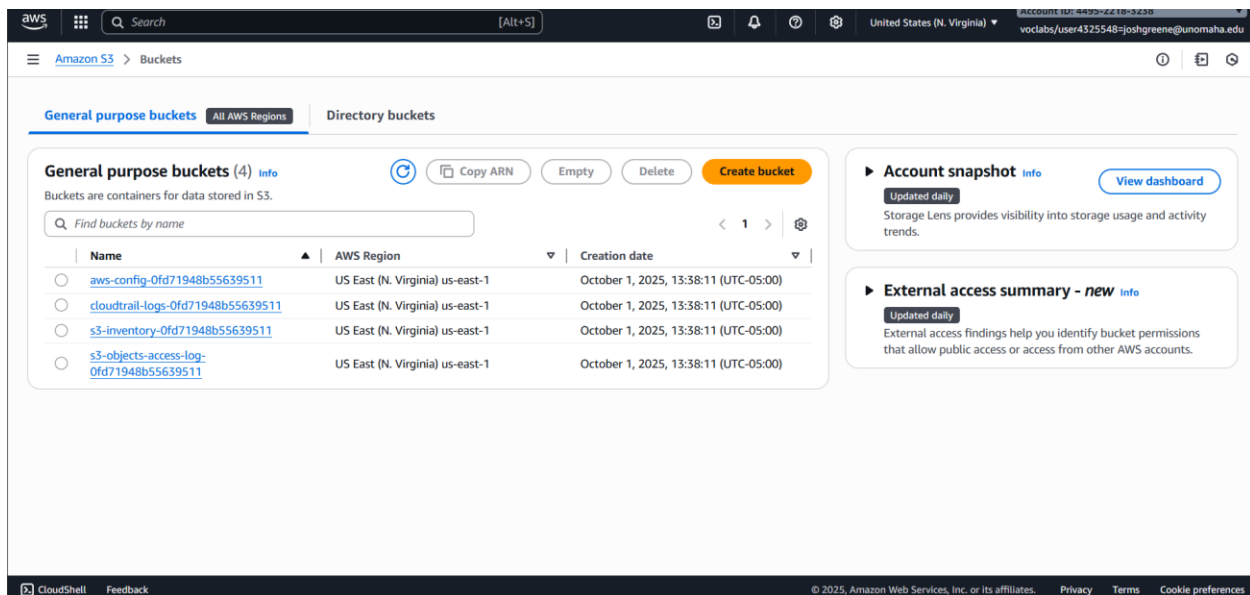


# Securing data in Amazon S3

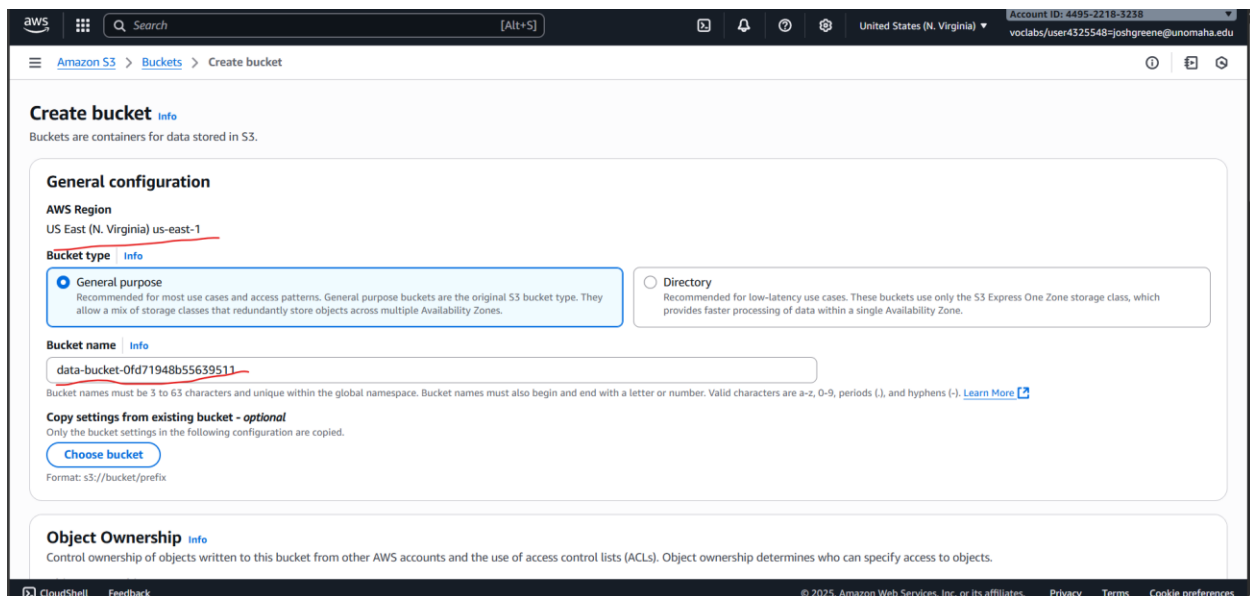
## Task 1.1: Create a bucket, apply a bucket policy, and test access



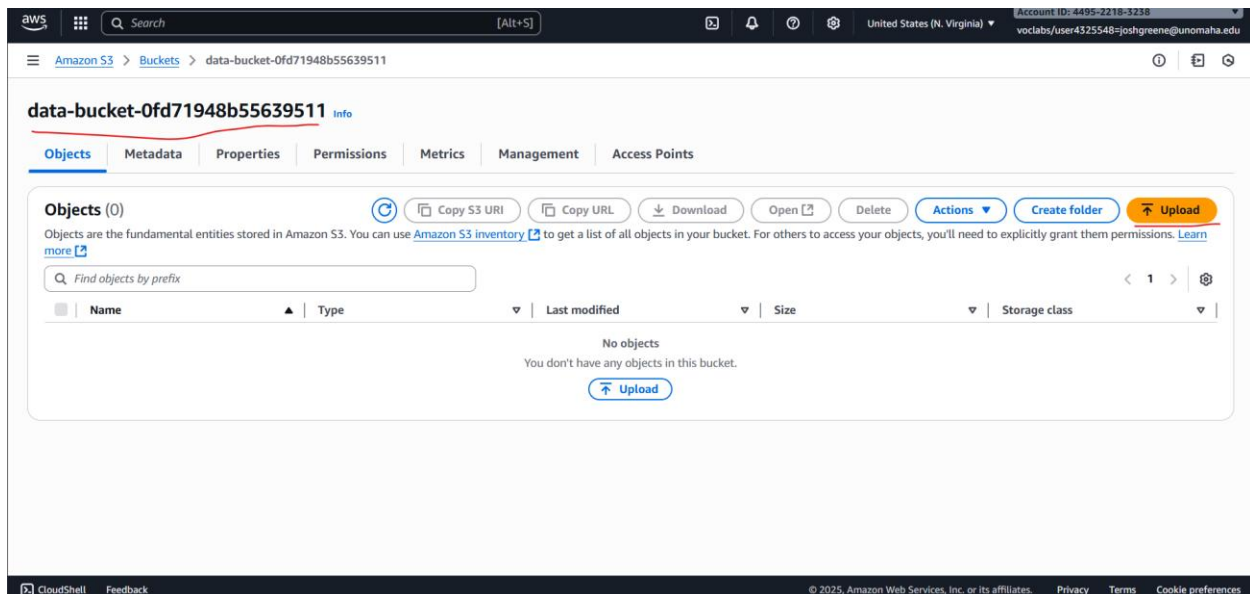
The first thing I did for this task was open the Amazon S3 service in the AWS Management Console.



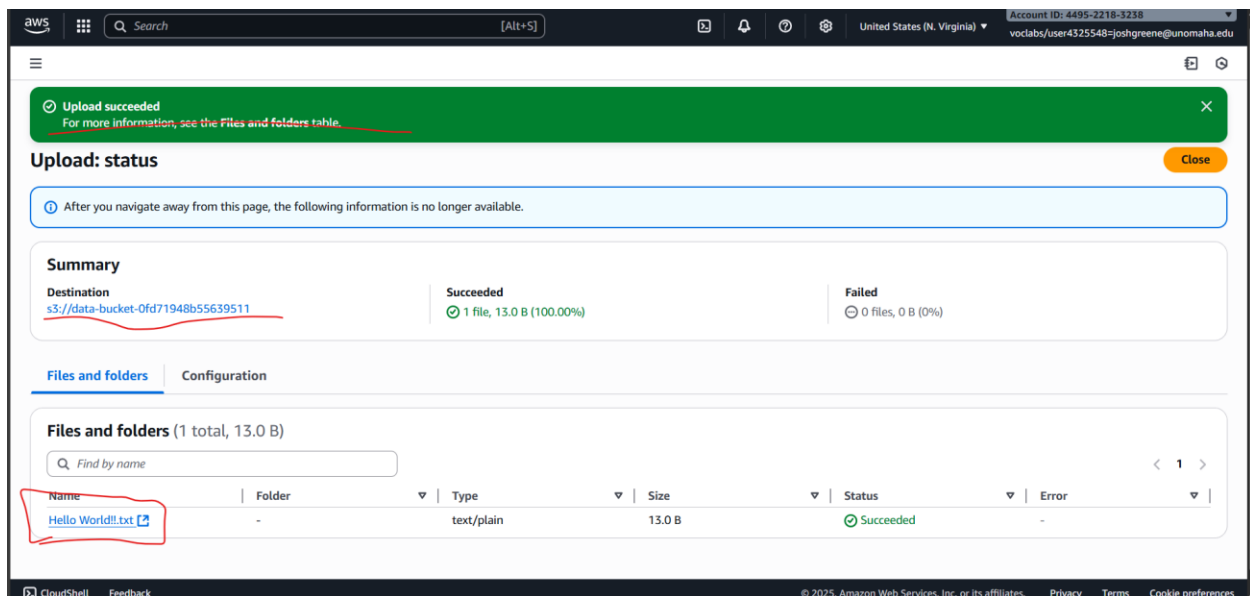
Once in the S3 console, I saw four buckets listed by default. From there, I selected the option to create a new bucket.



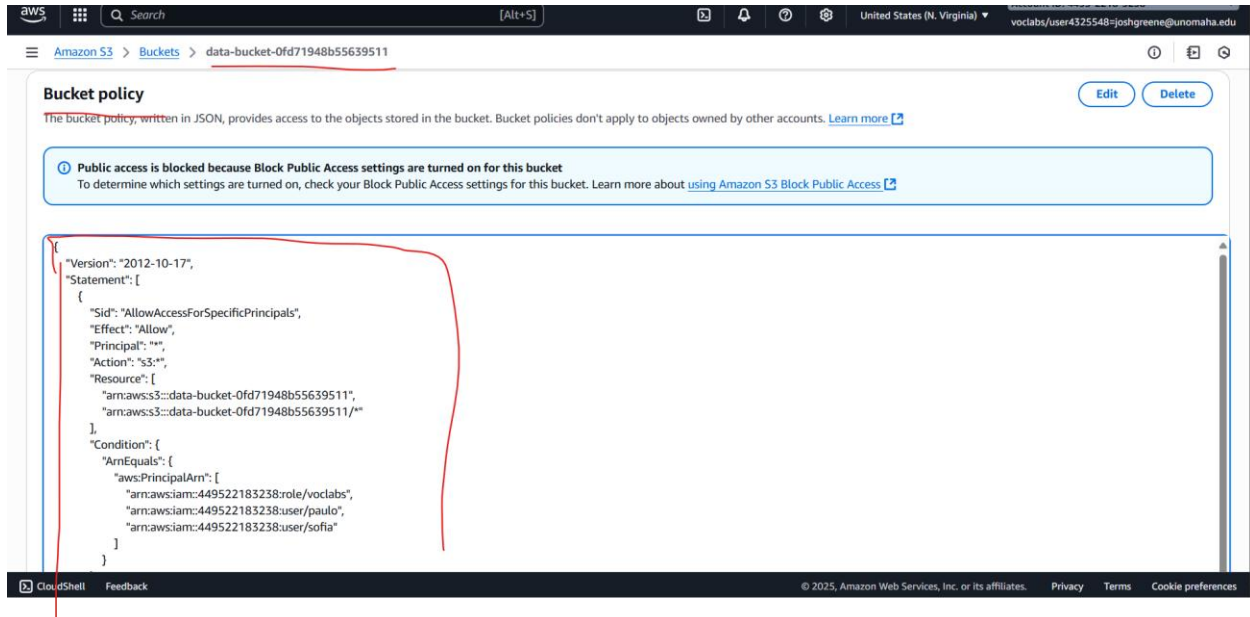
After clicking “Create bucket,” I made sure the AWS region was set to US East (N. Virginia). I named the bucket data-bucket-0fd71948b55639511 and then clicked Create bucket to finish the setup.



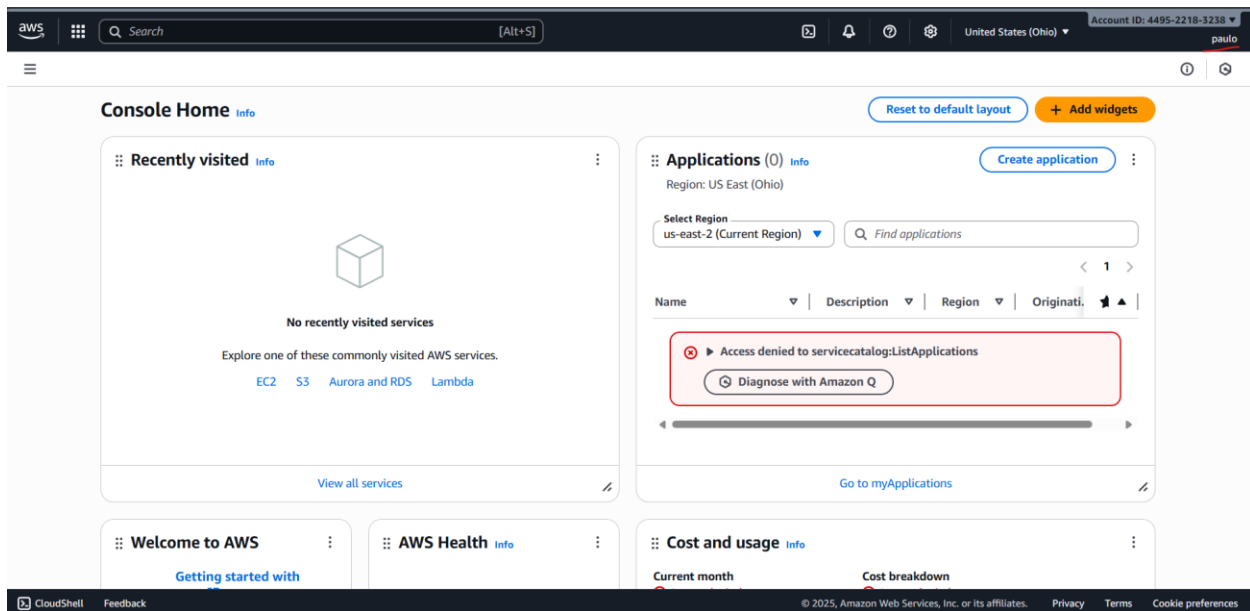
Here you can see that I’ve successfully created the new bucket. Next, I’m going to upload an object into it. To do that, I open my data bucket and click the orange Upload button to begin adding a file.



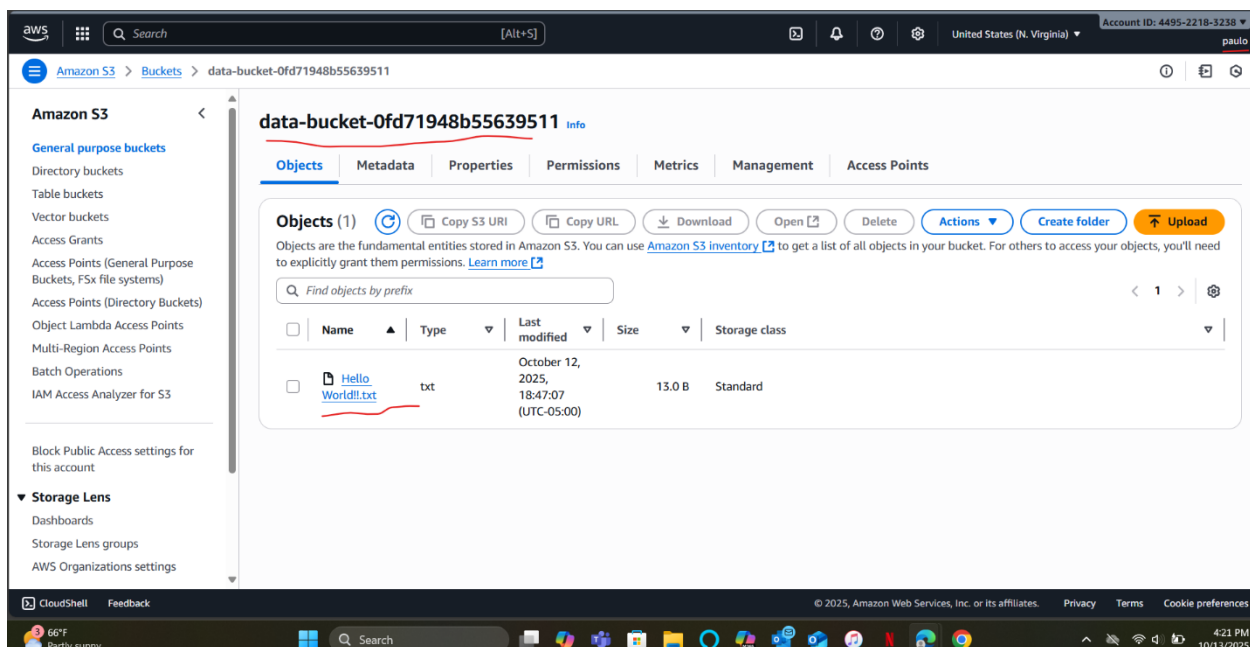
In this screenshot, you can see that my upload was successful. The uploaded object, Hello World!!txt, appears at the bottom of the screen, confirming that the file was added to the bucket correctly.



I then updated the bucket policy to control access for specific users. To do this, I opened the Permissions tab for my bucket, selected Bucket Policy, and pasted my custom JSON script that allows access for the voclabs, Paulo, and Sofia users while denying access to everyone else. After reviewing the policy, I clicked Save changes to apply it successfully.

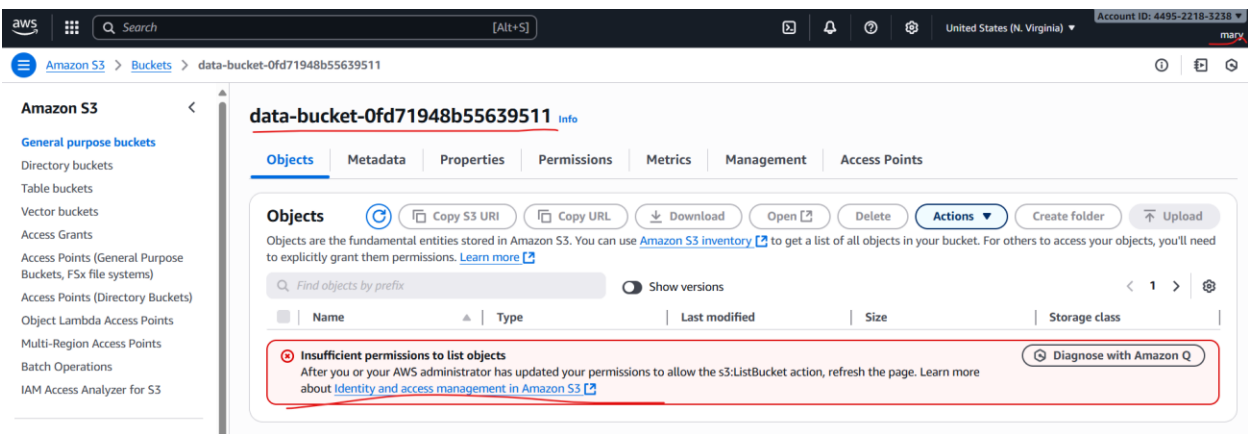


In the top-right corner of this screenshot, you can see that I'm logged in as Paulo. Here, I'm going to test whether he has access to the object I uploaded earlier in my data-bucket. I'll open the S3 console, navigate to the bucket, and try to view or download the Hello World!!txt file to confirm that his access permissions are working correctly.

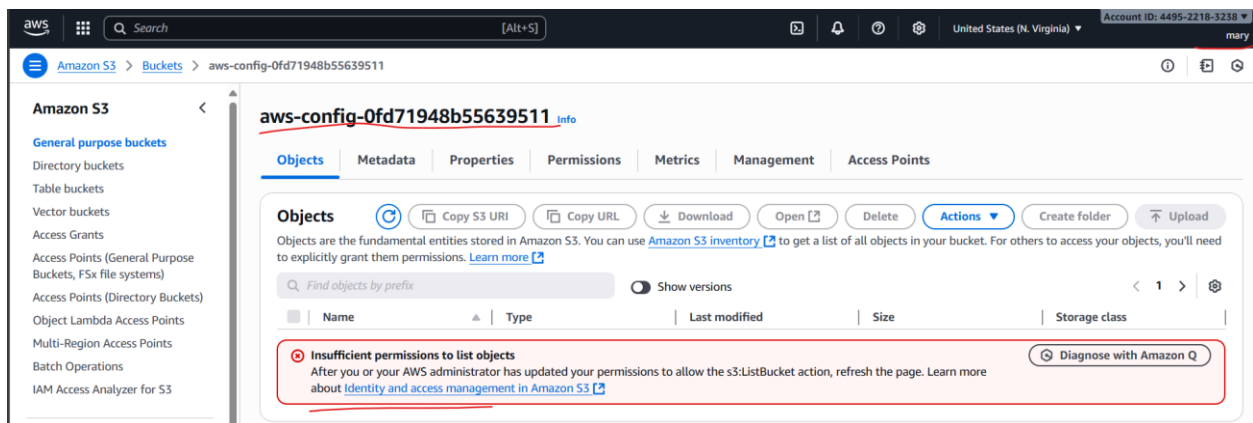


After opening S3 and navigating to my data-bucket, I confirmed that Paulo has access to my uploaded object. In the screenshot, you can clearly see the file Hello World!!txt listed inside the bucket, showing that Paulo's permissions were applied correctly and he can view the object without any access errors.



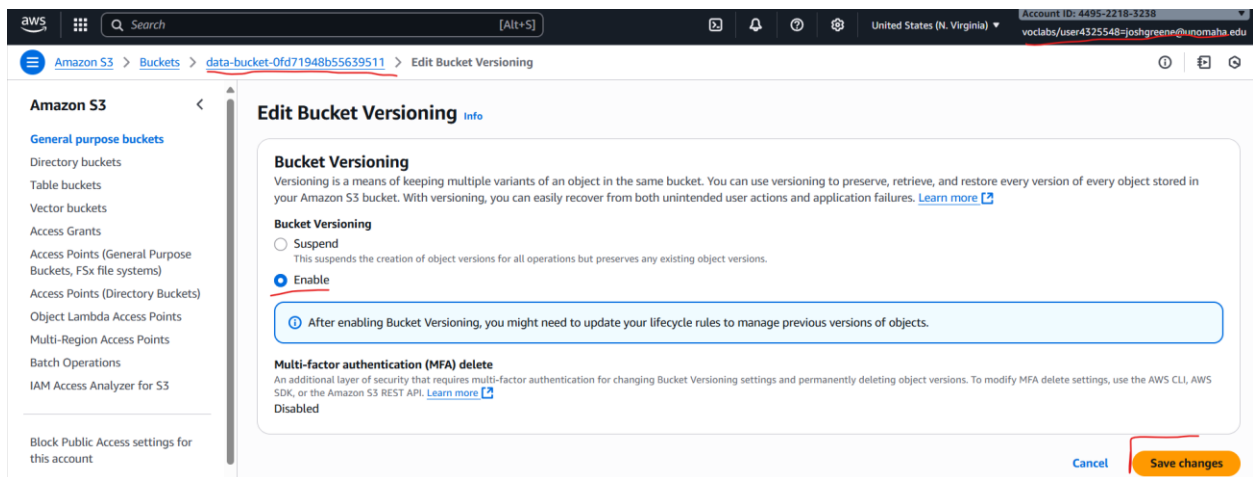


When Mary opens the data-bucket, she gets an “Insufficient permissions” message and can’t see any objects inside. This shows her access is correctly blocked, which is the expected result.



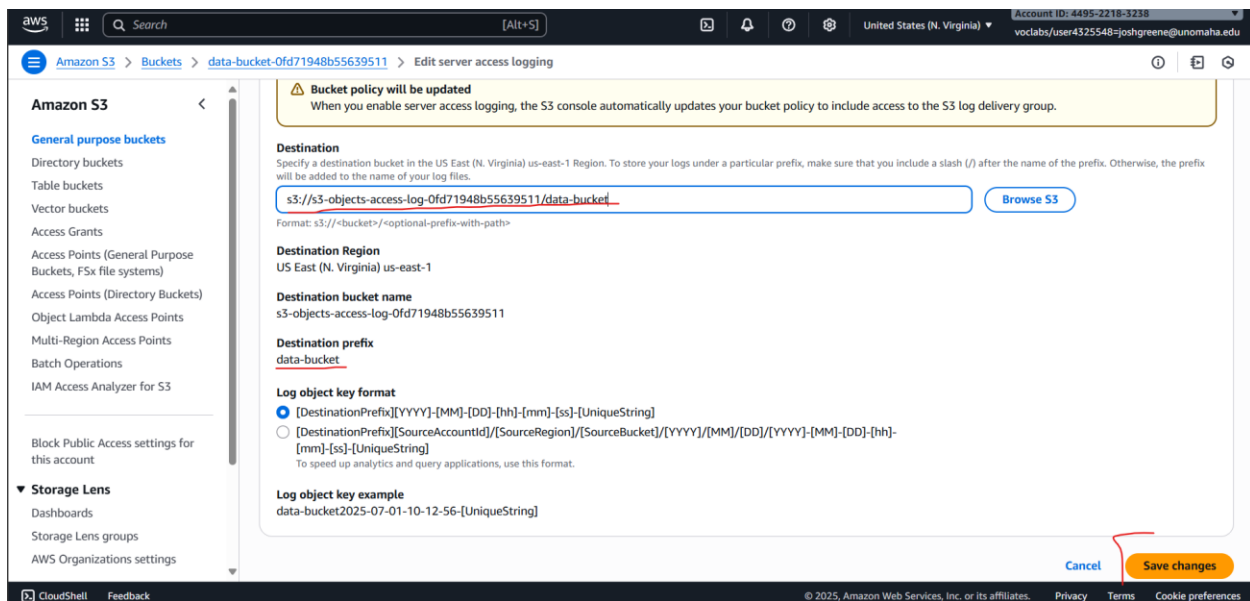
Here, I tested other buckets, and Mary again doesn’t have permission to access them. This confirms her restrictions are working as expected.

## Task 1.2: Enable versioning and object-level logging on a bucket

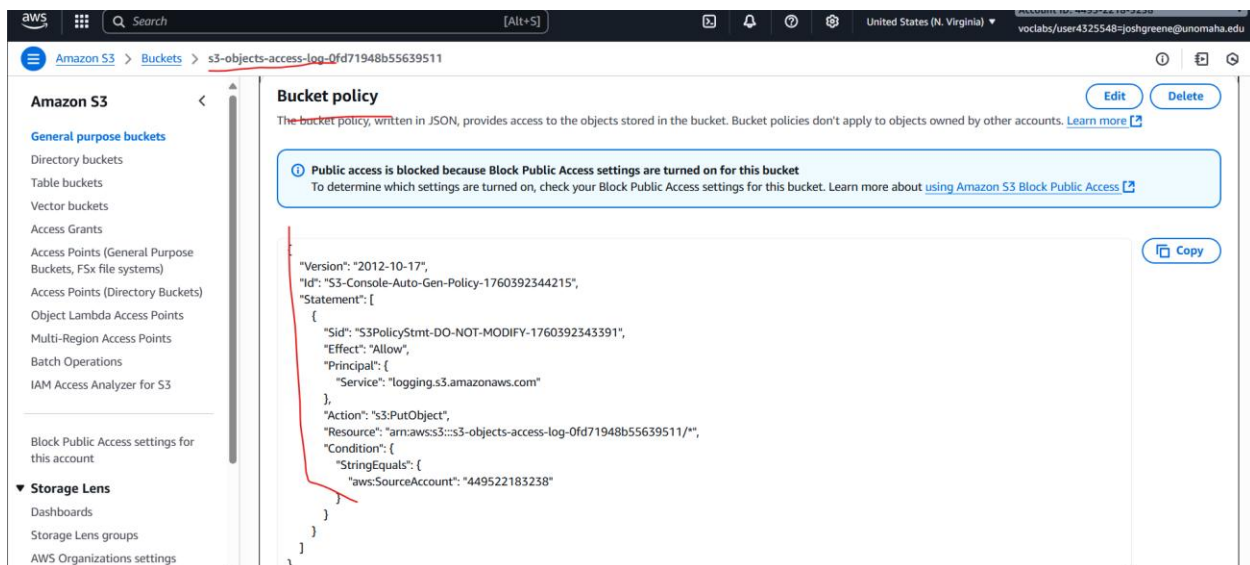


Here, I went back to my voclabs account and opened my data-bucket. Inside the bucket settings, I selected Edit bucket versioning, clicked Enable, and then saved the changes.

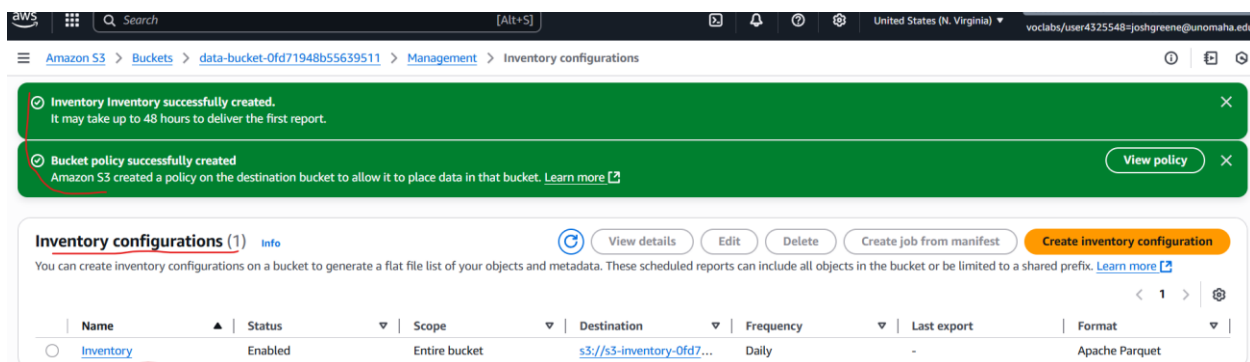




Next, I edited the server access logging settings for my bucket. I set the destination to my objects access log bucket, added the data-bucket prefix, and then saved the changes.

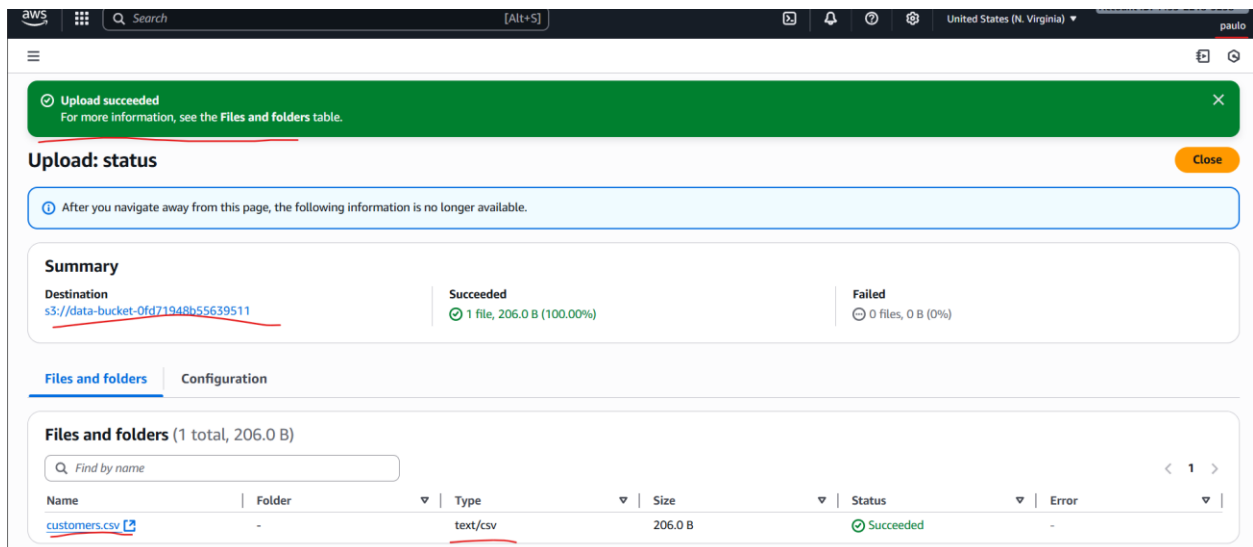


### Task 1.3: Implement the S3 Inventory feature on a bucket



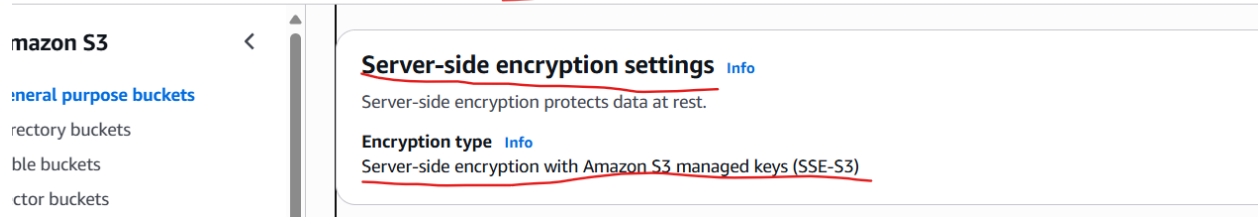
Here, I went to my data-bucket, opened the Management tab, and selected Inventory configuration. From there, I created a new configuration and named it Inventory.

## Task 1.4: Confirm that versioning works as intended

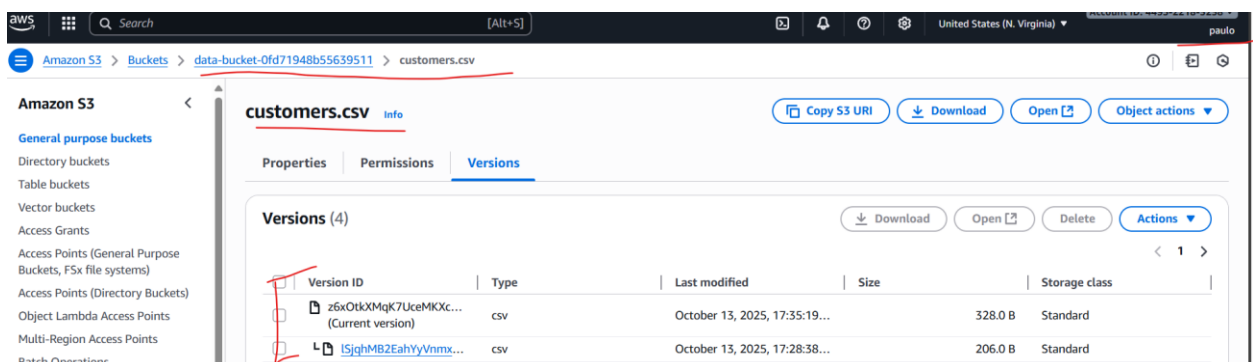


At the top right, you can see I'm logged in as Paulo. I uploaded the customers.csv file into the data-bucket to test versioning and access permissions.

Amazon S3 > Buckets > data-bucket-0fd71948b55639511 > customers.csv

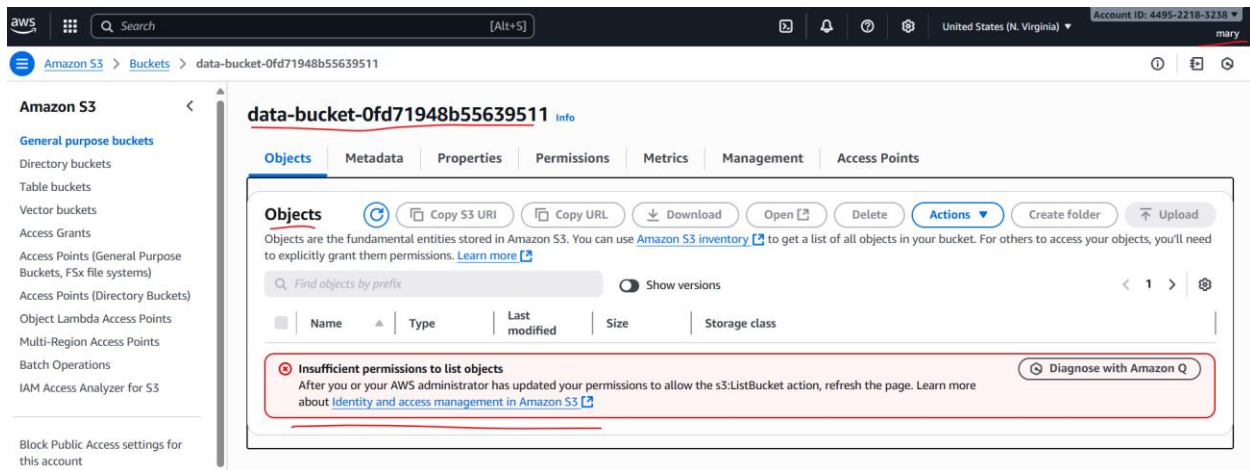


After uploading the file, I opened customers.csv to check its encryption settings. The object details show SSE-S3, which means the file was automatically encrypted using Amazon S3 managed keys.



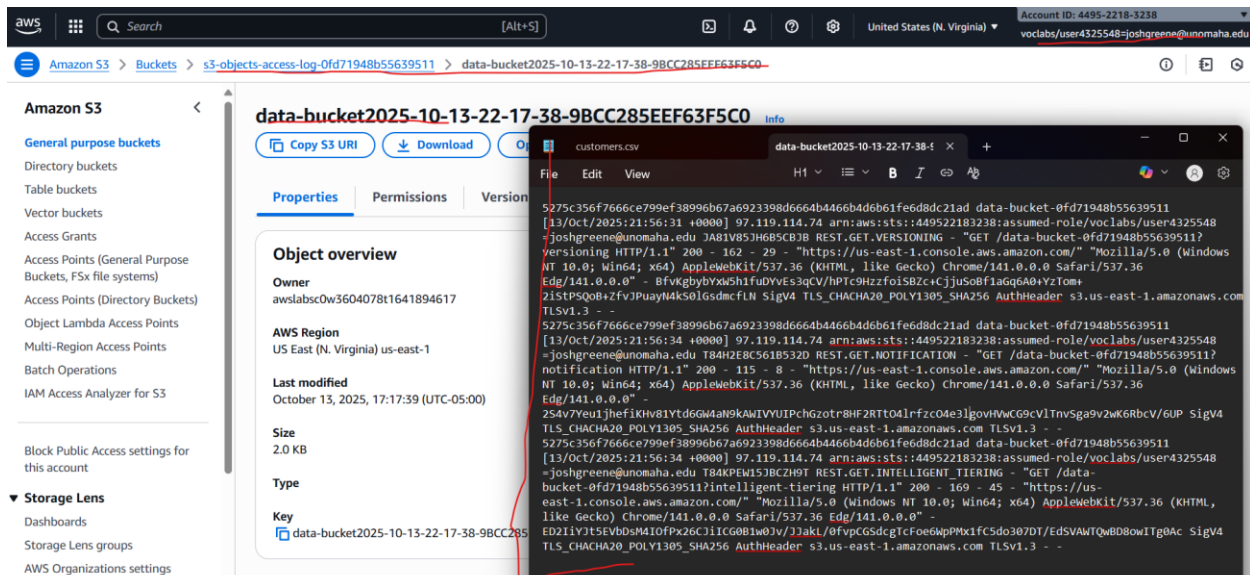
Here, I uploaded an updated version of customers.csv. The new version shows four records at the top, while the previous version only contains two. This confirms that versioning is enabled and tracking changes correctly.



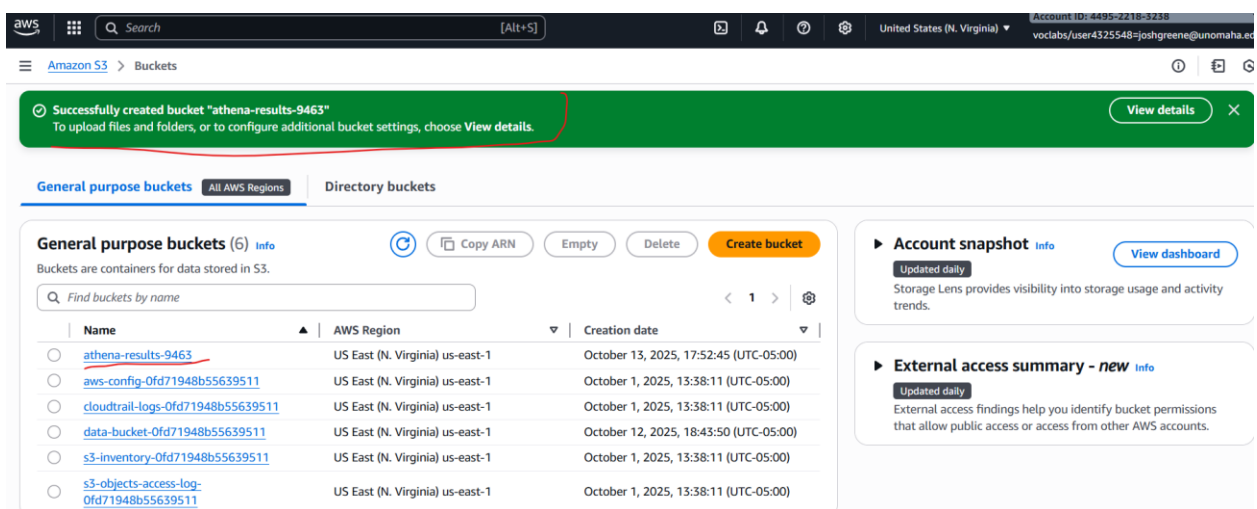


I logged in as Mary to check if she could see Paulo's uploaded object in the data-bucket, but she doesn't have permission. This confirms that her access is restricted as intended.

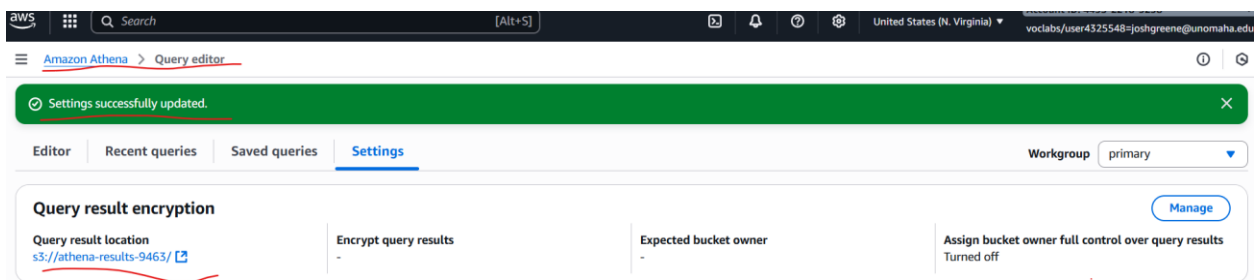
## Task 1.5: Confirm object-level logging and query the access logs by using Athena



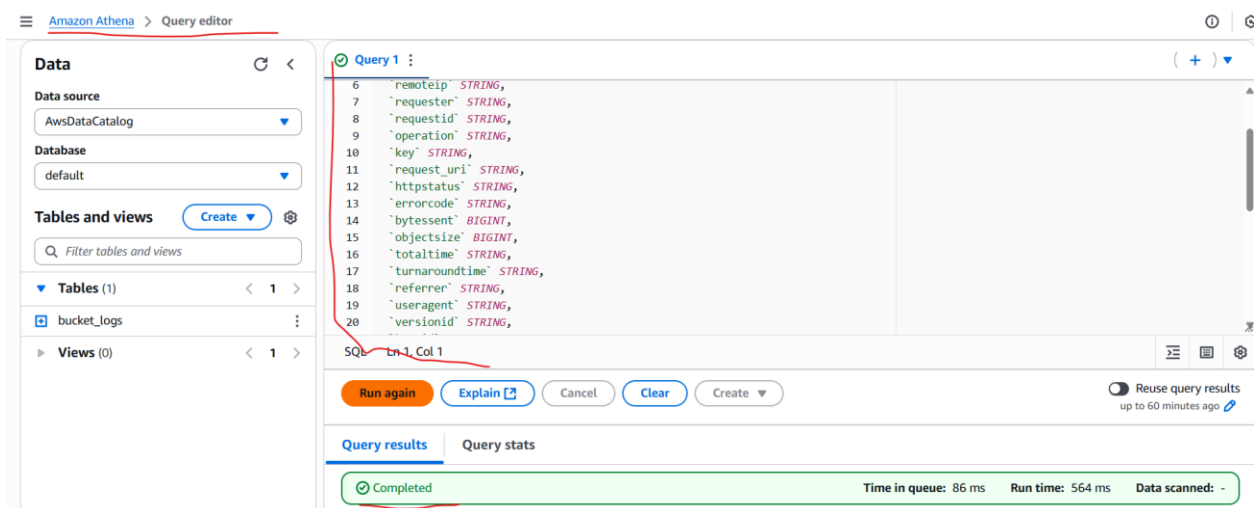
Here, I logged into my voclabs account and opened my list of S3 buckets. I selected the objects access log bucket, which displayed several log files for the data bucket. I downloaded one of the files, shown in the screenshot above. The file contains detailed lines showing who accessed the bucket, along with timestamps and status codes, confirming that object-level logging is working correctly.




Here, I created a new bucket named athena-results-9643. This bucket will be used to store the results from any queries I run in Amazon Athena, keeping my analysis data organized and separate from the main S3 buckets.



Here, I created a query in Amazon Athena to analyze my S3 access logs and view user activity results stored in my athena-results-9643 bucket.



Here, I pasted the query into the Athena Query Editor and ran it successfully, confirming that the table was created and connected to my S3 logs without any errors.

Query results		Query stats		
Completed		Time in queue: 103 ms	Run time: 924 ms	Data scanned: 14.28 KB
Results (4)		<a href="#">Copy</a> <a href="#">Download results CSV</a>		
<input type="text" value="Search rows"/>		< 1 > 		
#	requester	operation	key	httpstatus
1	arn:aws:iam::449522183238:user/paulo	REST.GET.VERSIONING	-	200
2	arn:aws:iam::449522183238:user/paulo	REST.GET.OBJECT_LOCK_CONFIGURATION	-	404
3	arn:aws:iam::449522183238:user/paulo	REST.GET.VERSIONING	-	200
4	arn:aws:iam::449522183238:user/paulo	REST.GET.OBJECT_TAGGING	customers.csv	200

Here, I ran the second query, which displayed Paulo’s activity in the results. The 200 status codes show successful requests, while the 404 indicates a request for something that didn’t exist.

## Project Summary

In this project, I created and secured an Amazon S3 environment by setting up buckets with the proper permissions, logging, and monitoring. I started by creating a data-bucket and uploading a test file, then added a bucket policy that allowed access for voclabs, Paulo, and Sofia, while denying others like Mary. I confirmed that the policy worked by testing access under each user.

Next, I enabled versioning, server access logging, and S3 Inventory to track changes and monitor activity. I verified that encryption with SSE-S3 was active and that multiple versions of files were stored correctly. Finally, I used Amazon Athena to analyze the S3 access logs, which showed successful requests (200) from Paulo and denied ones (403 or 404) from other users. This phase showed how to use S3 features such as policies, encryption, versioning, and logging to protect data, monitor access, and maintain a secure storage environment.