PART-B

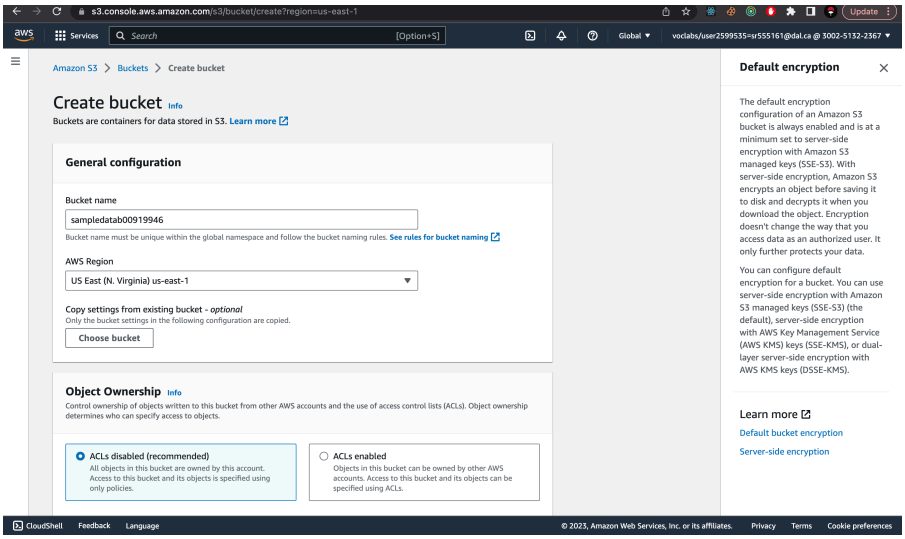

Fig. 1: Create a sampledatab00919946 bucket as mentioned in docs.

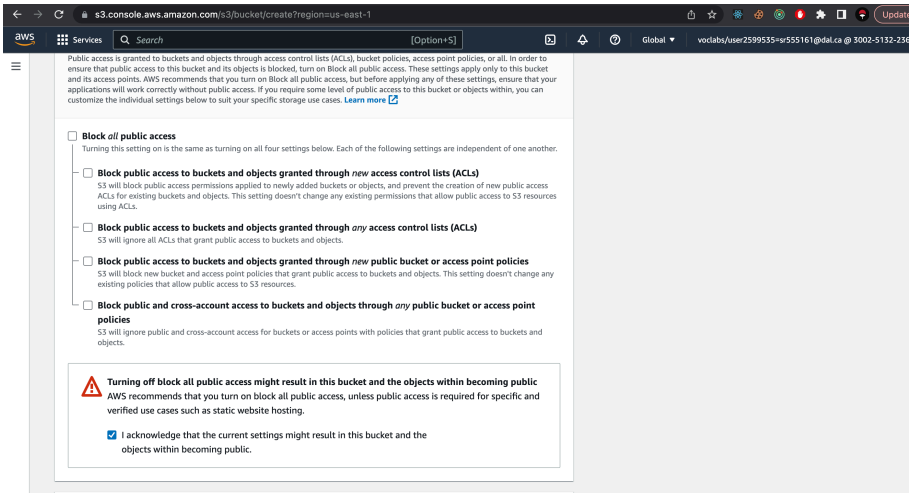

Fig. 2: Removing the public access block.

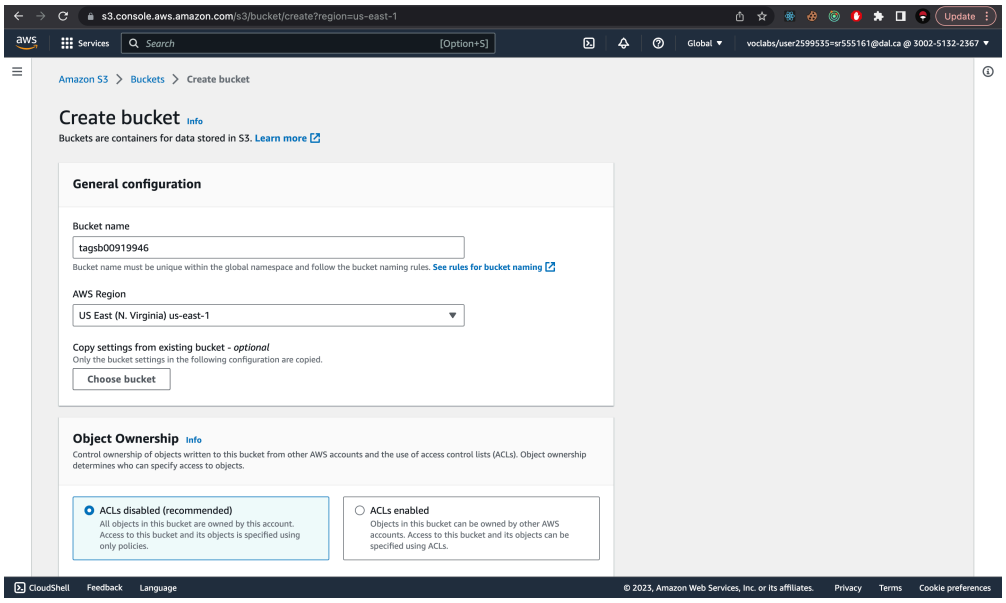

Fig. 3: Creating another bucket tagsb00919946.

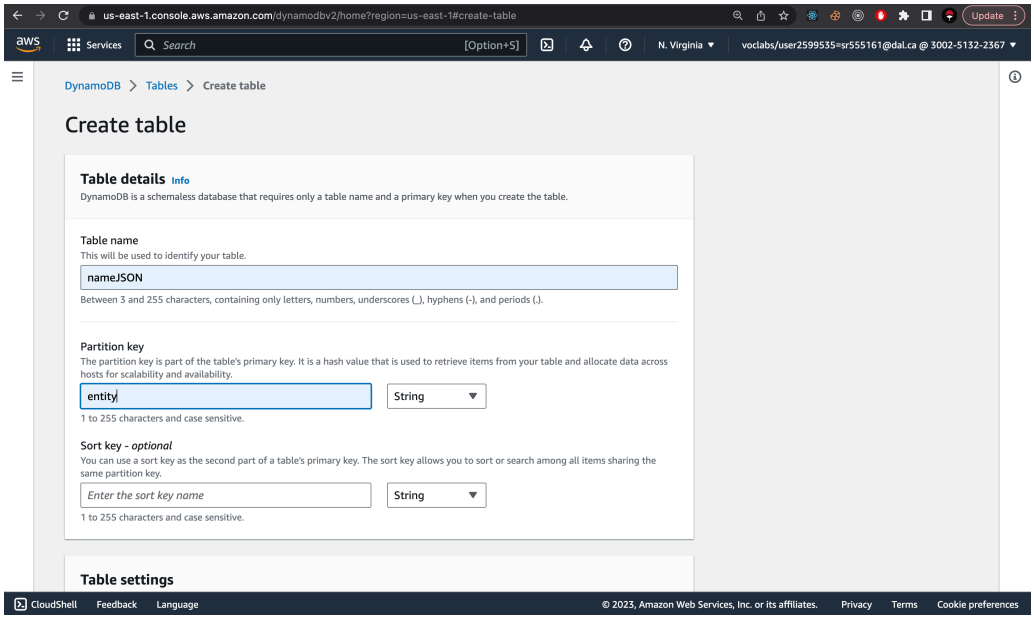

Fig. 4: Creating a DynamoDB table nameJSON to store the data from tags bucket..

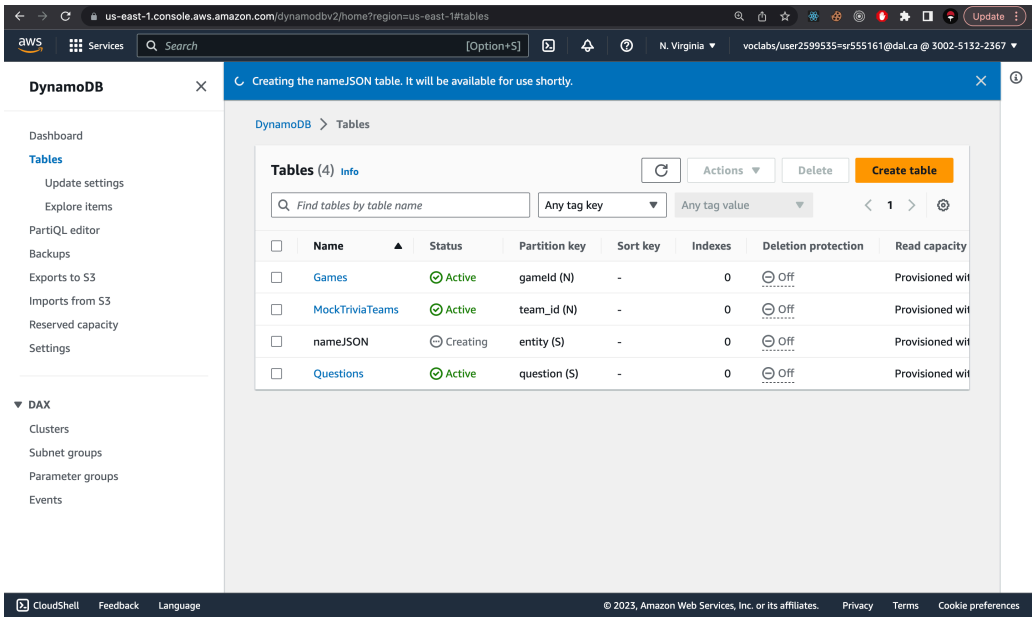

Fig. 5: nameJSON table created.

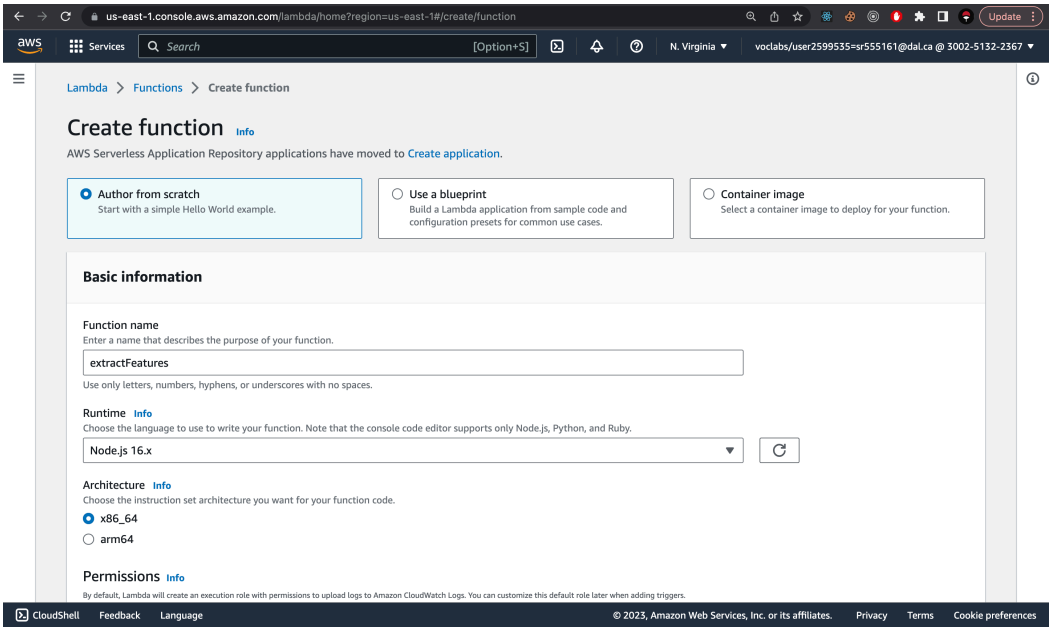

Fig. 6: extractFeatures Lambda function as per the requirement.

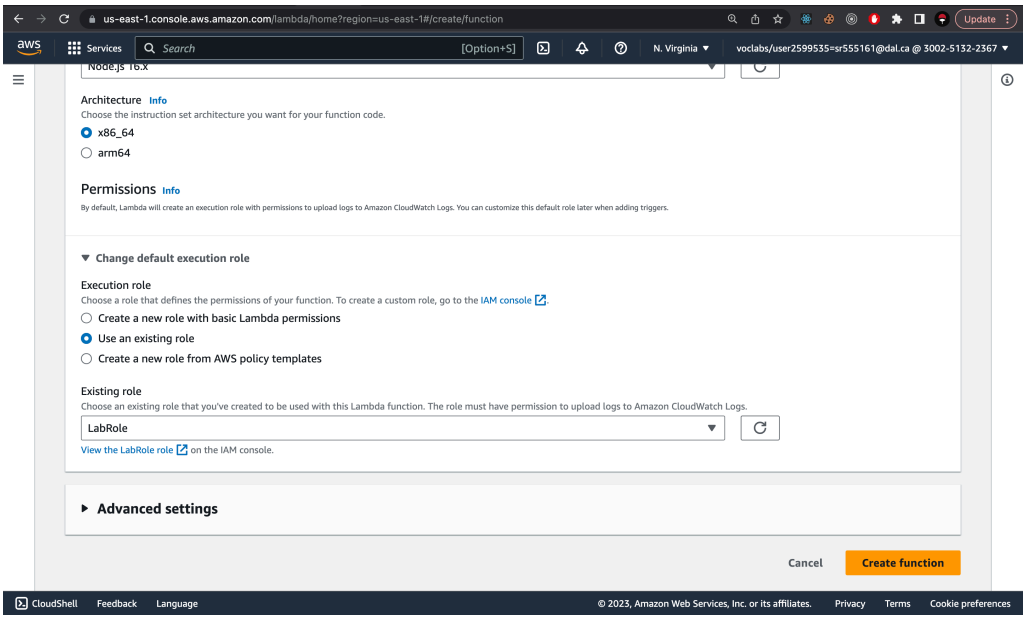

Fig. 7: selecting the IAM role as LabRole.

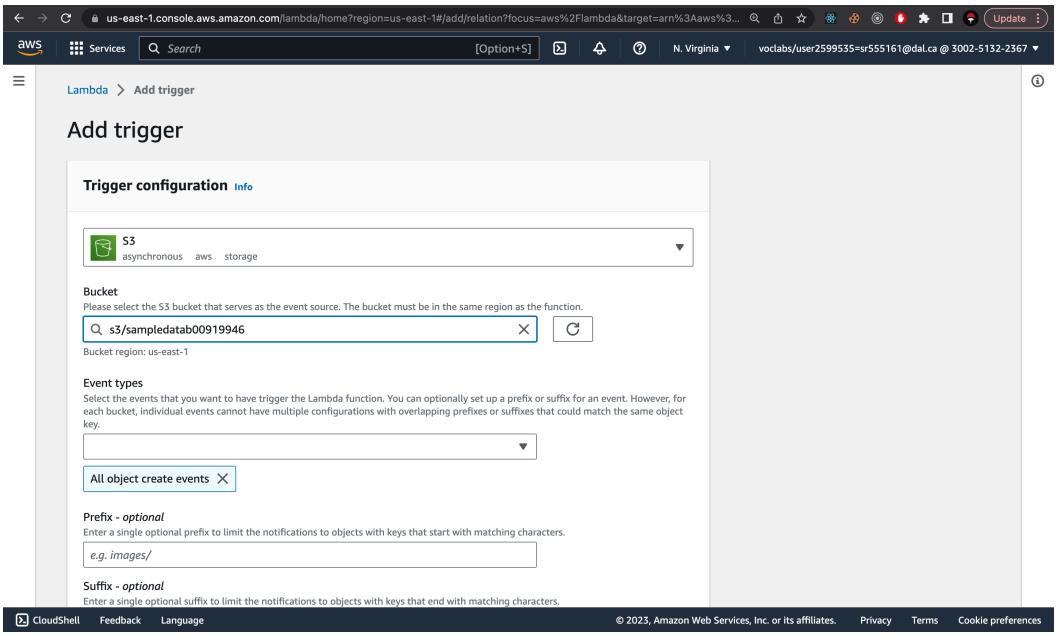

Fig. 8: Adding trigger as S3 to the extractFeatures Lambda function as per the requirement.

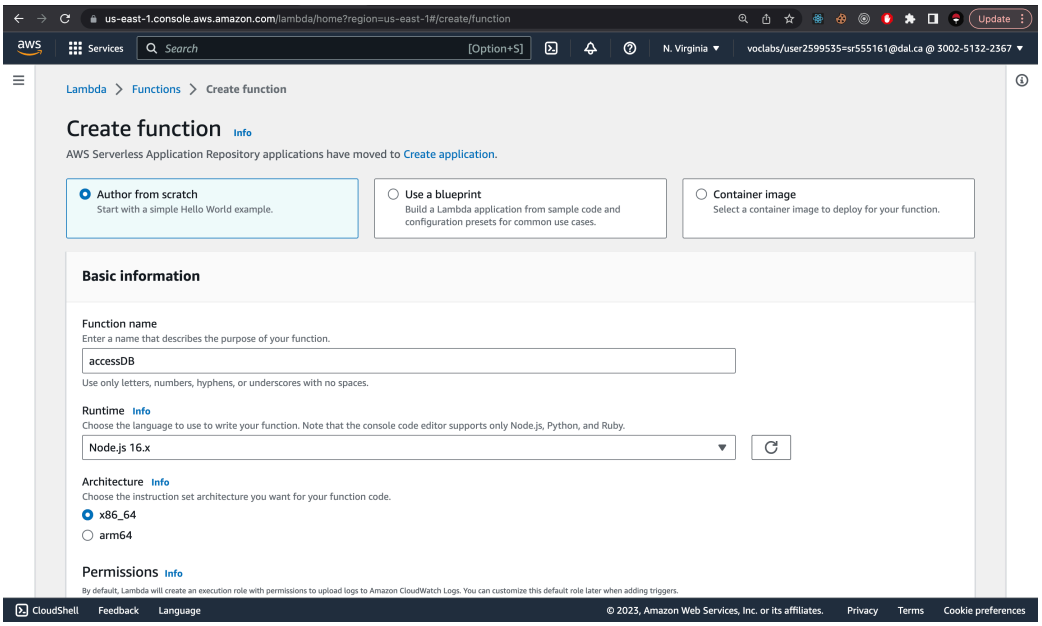

Fig. 9: accessDB Lambda function as per the requirement.

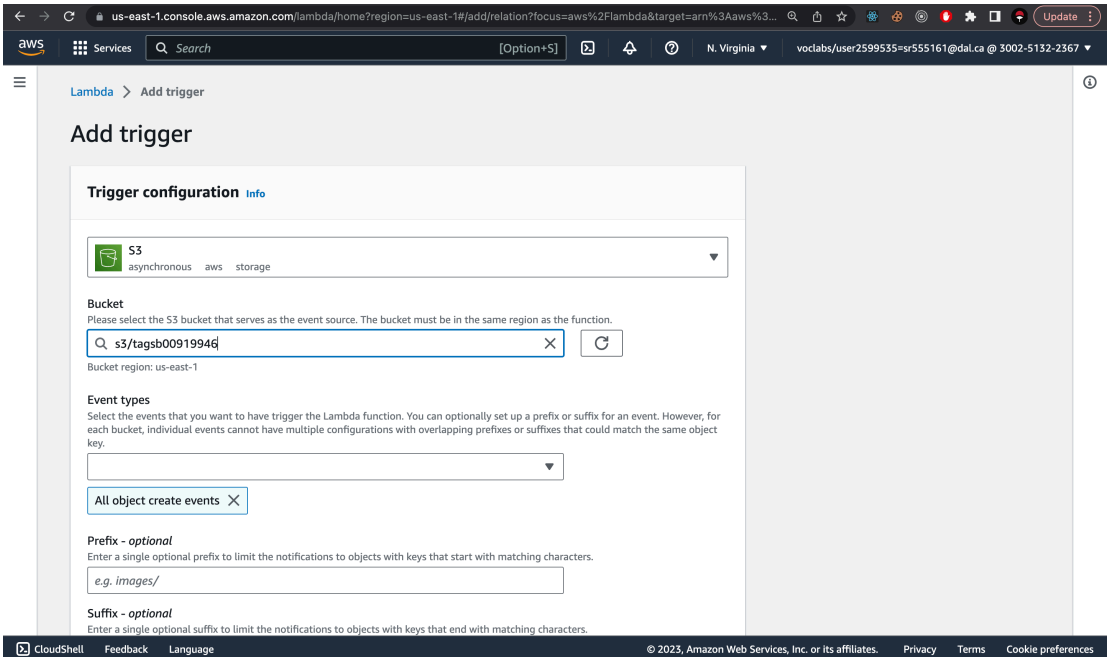

Fig. 10: Adding tags bucket trigger to the accessDB Lambda function as per the requirement.

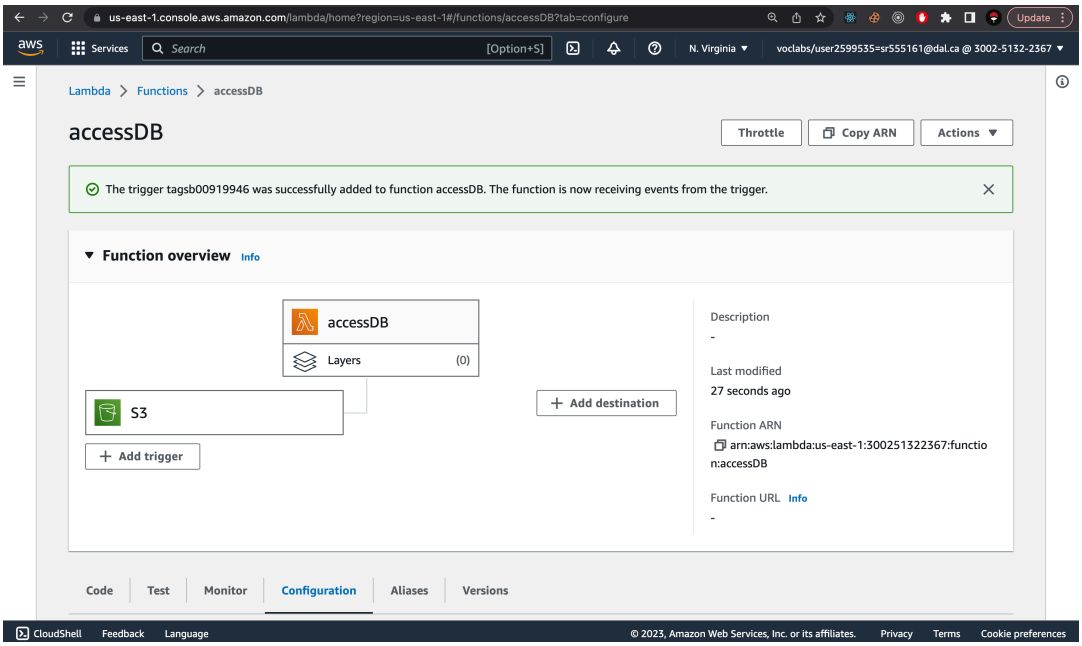

Fig. 11: accessDB S3 trigger.

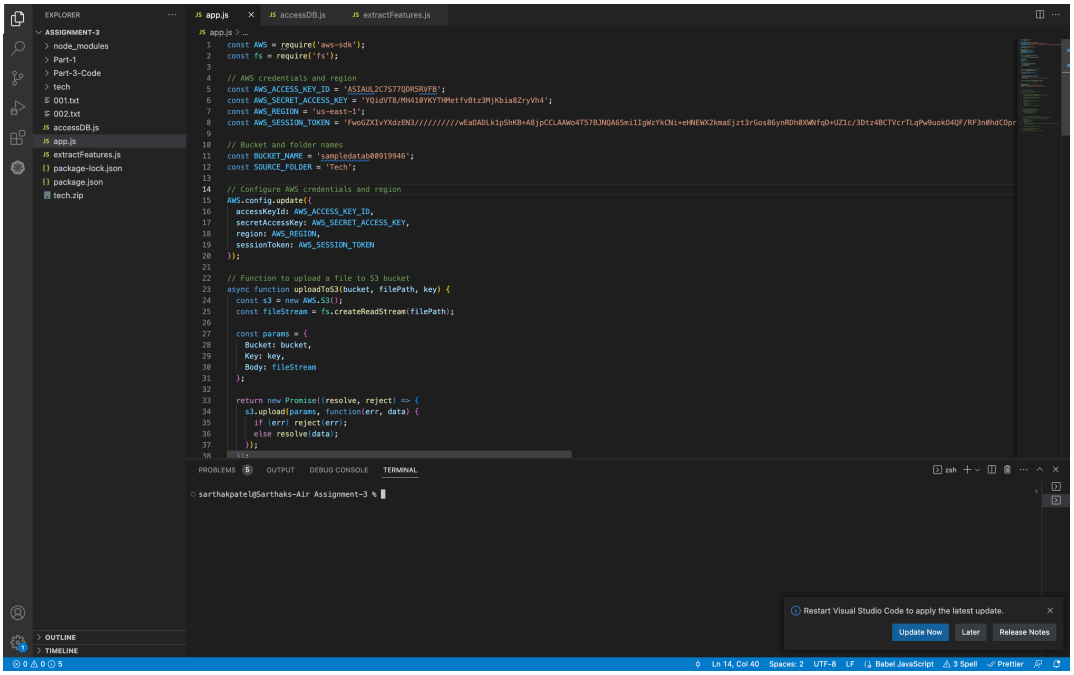

Fig. 12: Javascript code written to put the tech folder files in sampleDataB00919946 S3 bucket with 100 milliseconds.

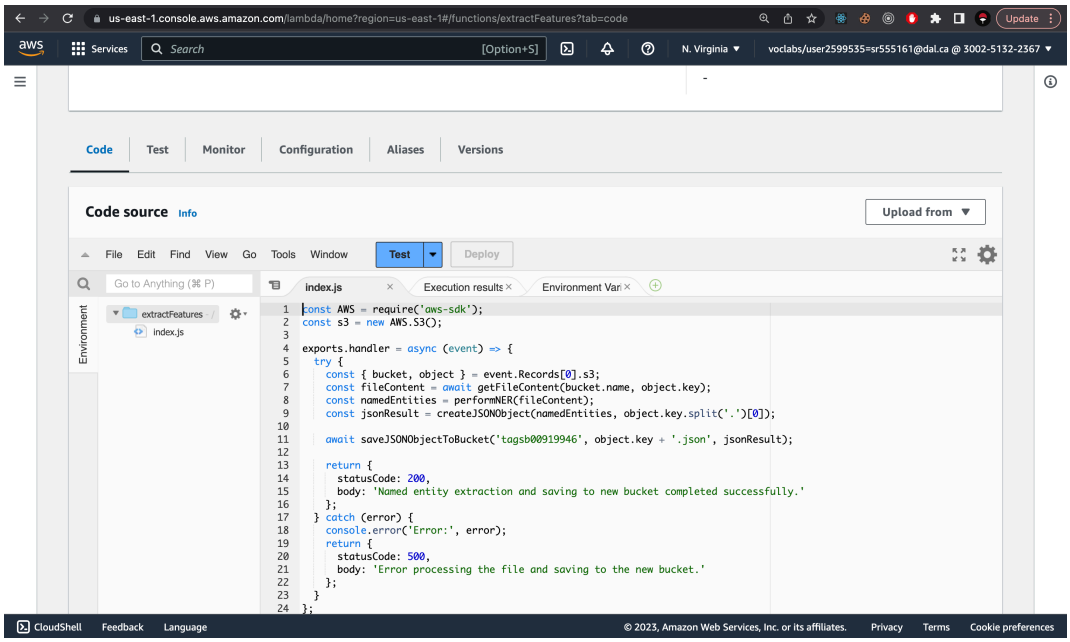

Fig. 13: extractFeatures Lambda function.

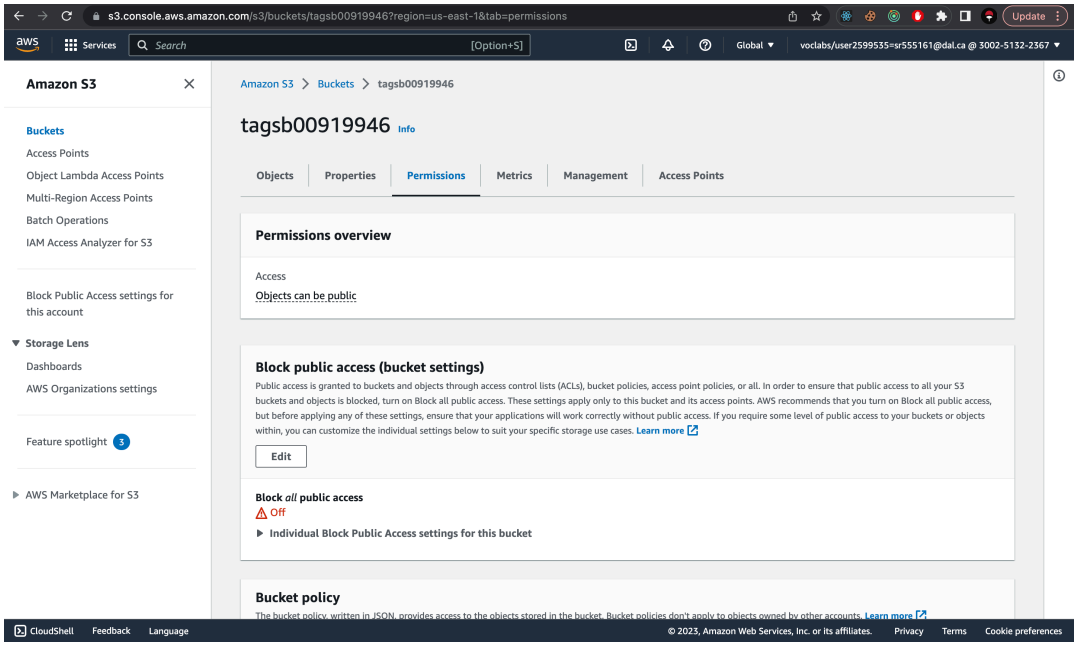

Fig. 14: Setting permissions for the tagsB00919946 S3 Bucket.

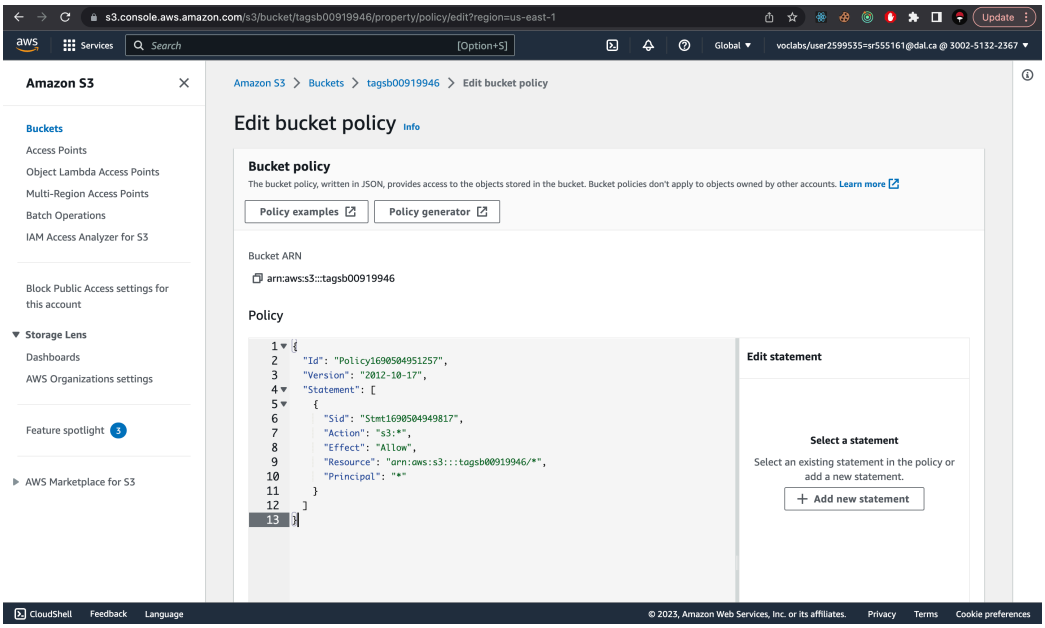

Fig. 15: Setting policy for the tagsB00919946 S3 Bucket.

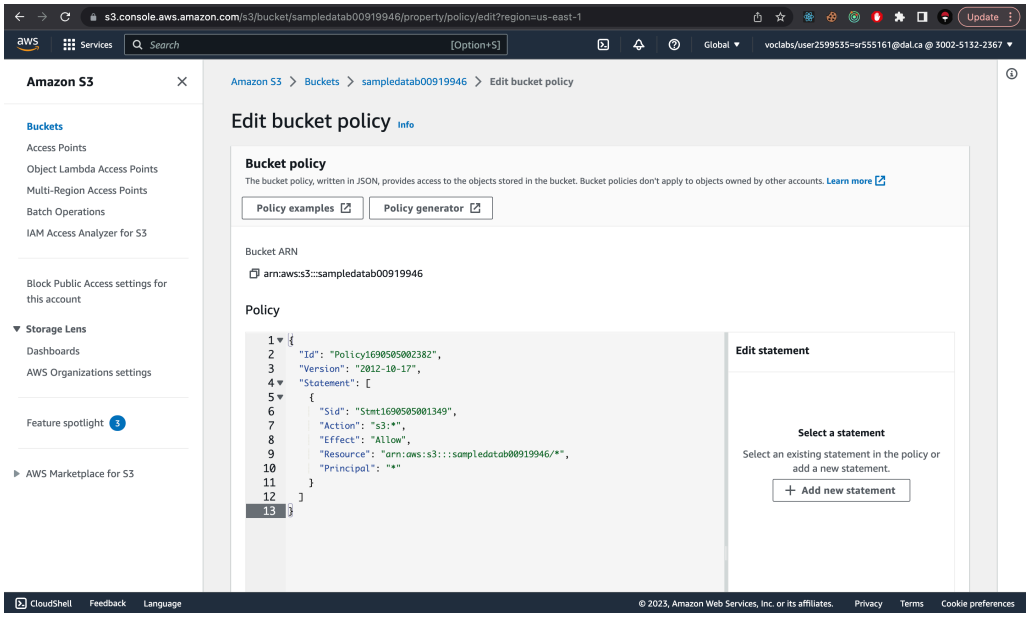

Fig. 16: Setting the policy for the sampleB00919946 S3 Bucket.

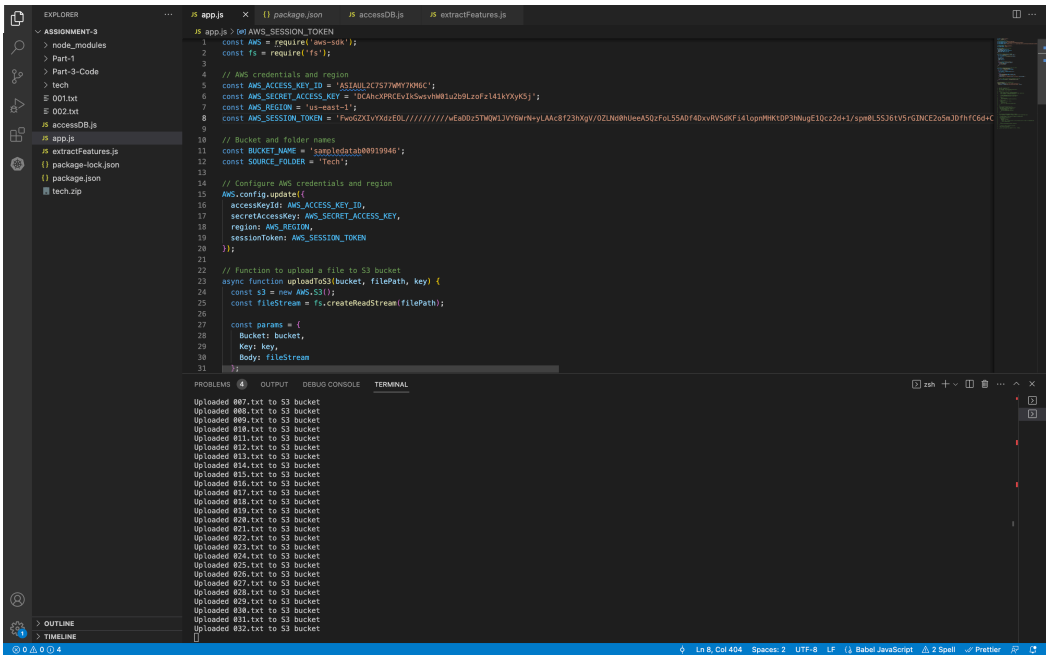

Fig. 17: Files getting stored in sampledatab00919946 S3 Bucket.

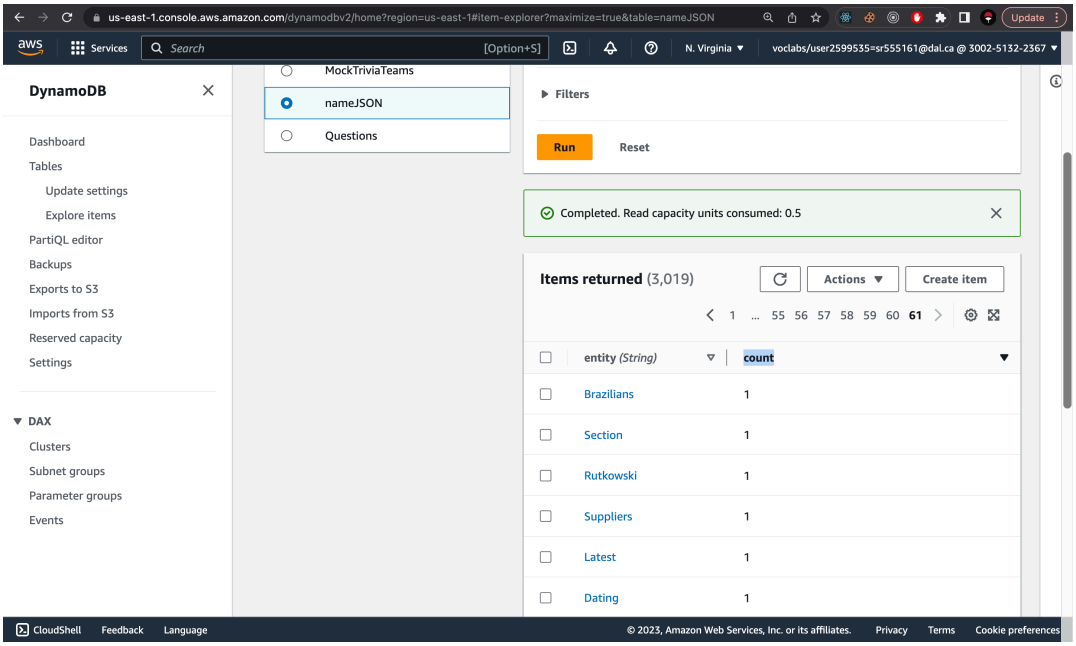

Fig. 18: Named Entities stored in DynamoDB after the triggers of lambda from S3 Bucket.

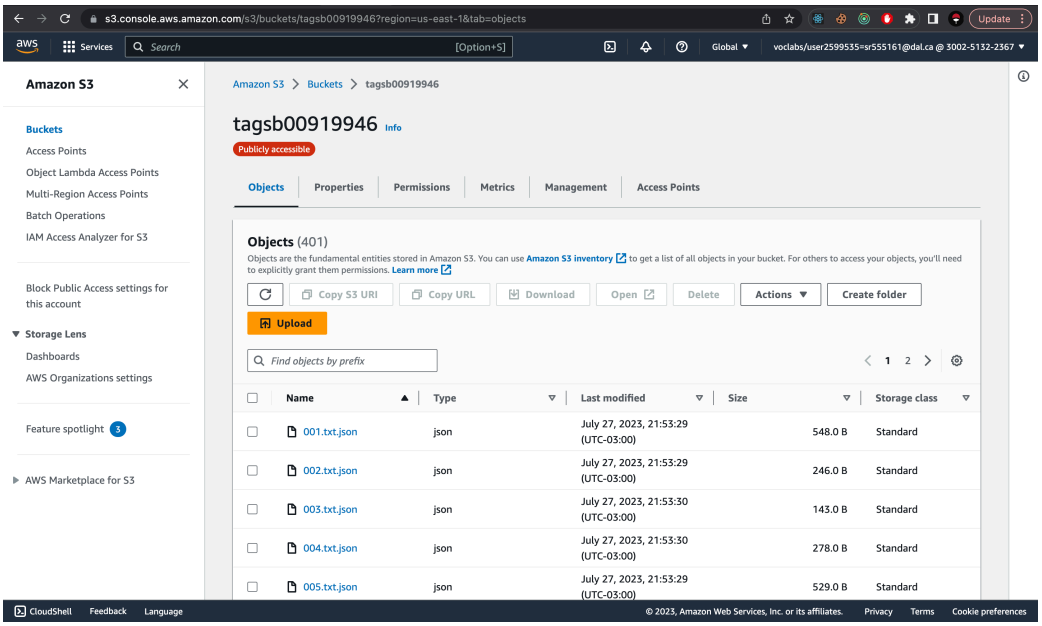

Fig. 19: 401 JSON files stored in tagsb00919946 S3 Bucket.

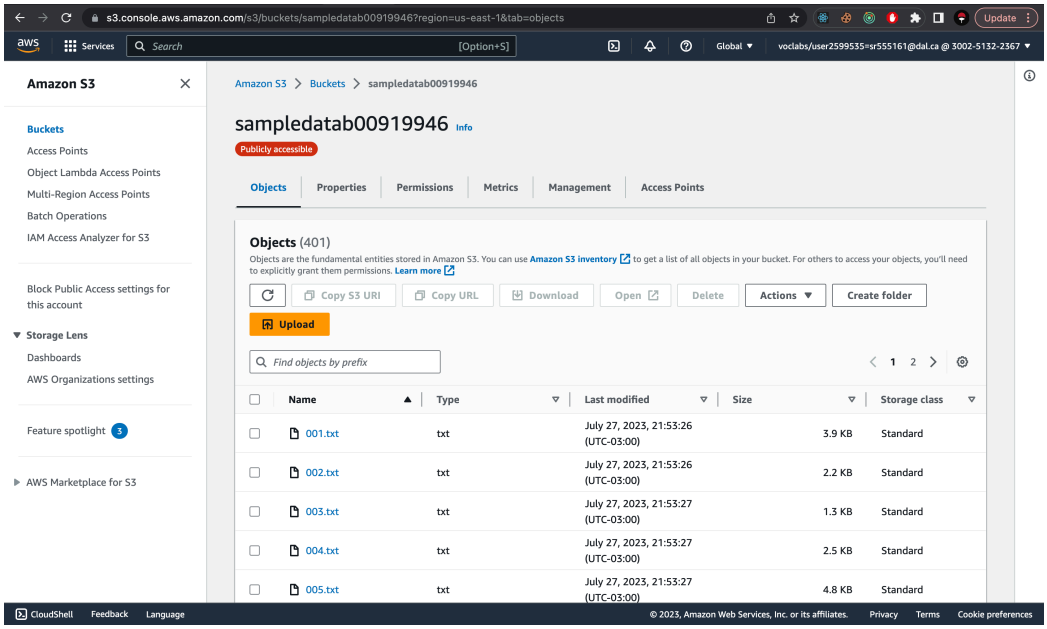

Fig. 20: As it is file from tech folder stored in sampledatab00919946 S3 Bucket.

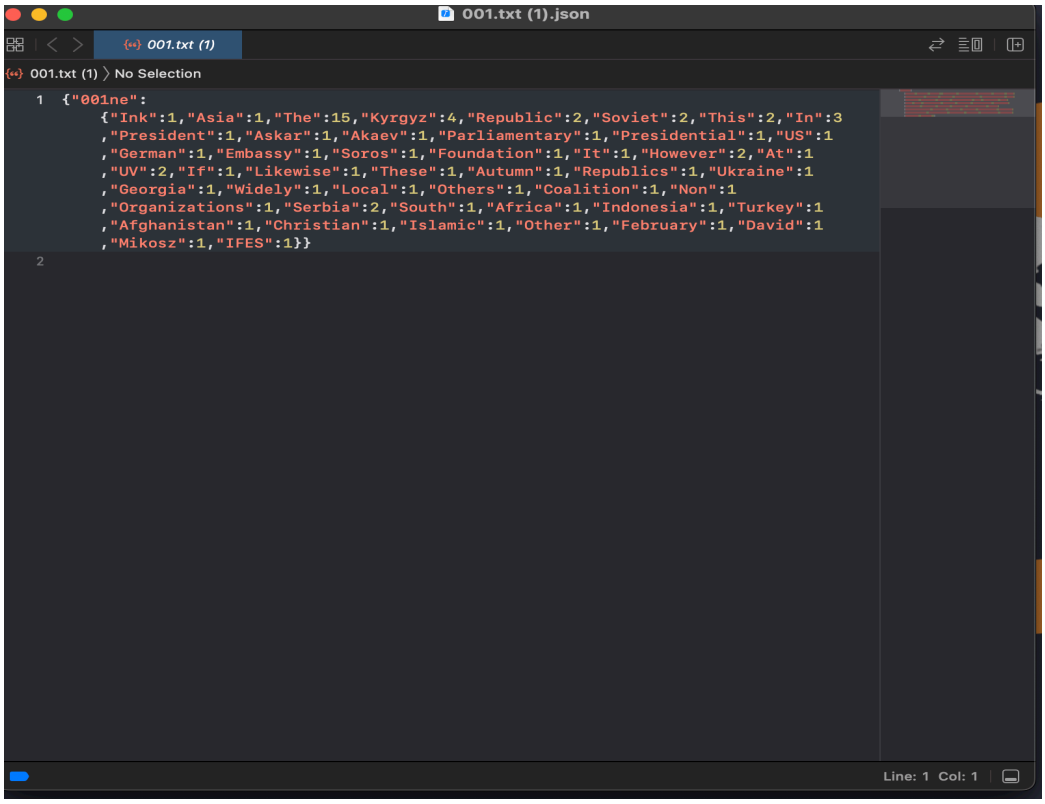

Fig. 21: Random JSON file selected from tagsb00919946 S3 bucket.

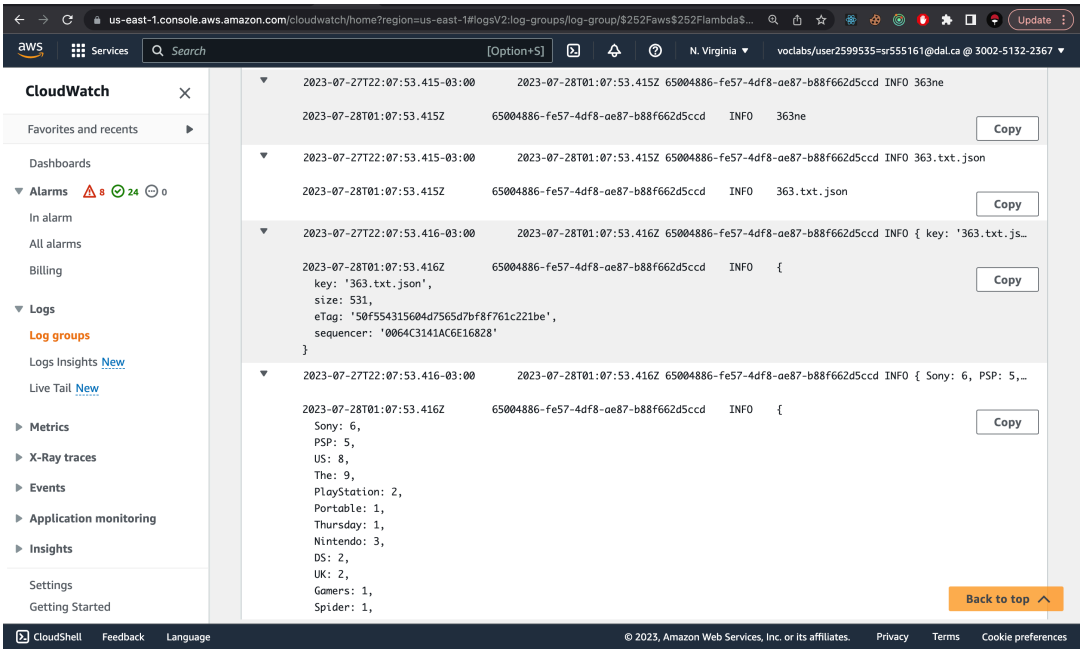

Fig. 22: Random event cloud watch logs.

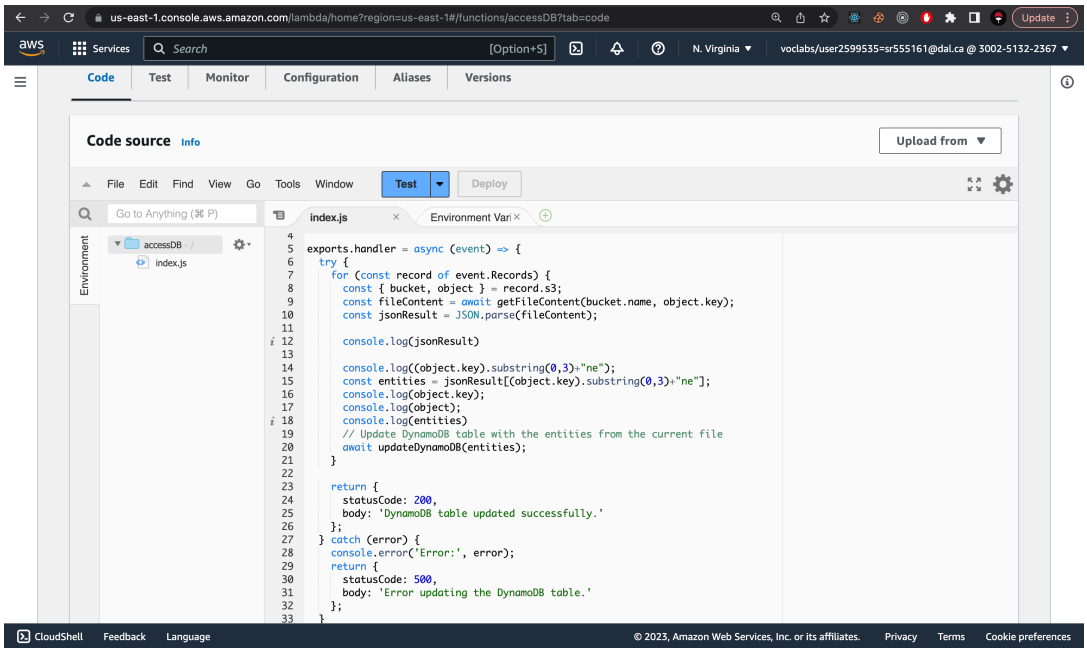

Fig. 23: The accessDB lambda function code for which cloud watch logs shown in fig. 22.

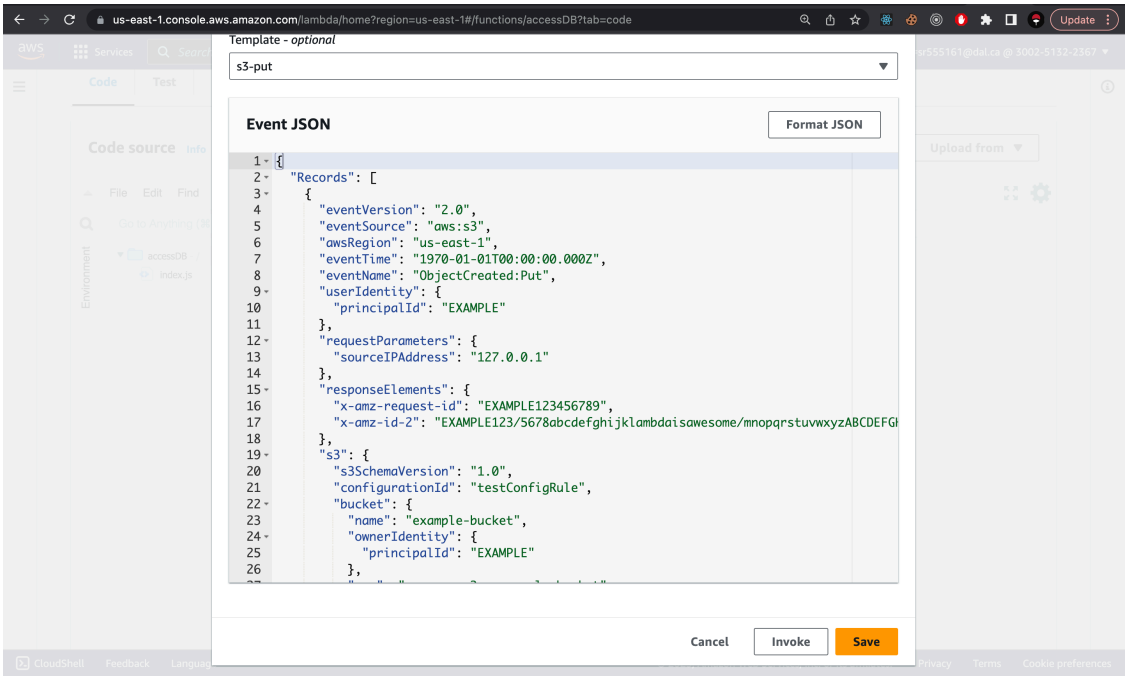

Fig. 24: Test event for the accessDB lambda function.

Note

I couldn't find the way to display the lambda function events log, as they are directly triggered from S3 bucket. So I displayed the cloud watch logs to complete the requirement of functional testing.