**Assignment-3 Solutions**

**Json Data for the Assignment:**

[{"id": 1, "status": "delivered", "amount": 20.5, "date": "2024-03-09"},

{"id": 2, "status": "cancelled", "amount": 15.0, "date": "2024-03-09"},

{"id": 3, "status": "order placed", "amount": 22.5, "date": "2024-03-09"},

{"id": 4, "status": "delivered", "amount": 19.5, "date": "2024-03-09"},

{"id": 5, "status": "cancelled", "amount": 18.0, "date": "2024-03-09"},

{"id": 6, "status": "delivered", "amount": 23.5, "date": "2024-03-09"},

{"id": 7, "status": "order placed", "amount": 20.0, "date": "2024-03-09"},

{"id": 8, "status": "delivered", "amount": 25.5, "date": "2024-03-09"},

{"id": 9, "status": "delivered", "amount": 21.5, "date": "2024-03-09"},

{"id": 10, "status": "cancelled", "amount": 17.5, "date": "2024-03-09"}]

The file is stored in the working repository.

**Setting up S3 buckets:**

* Create two S3 buckets: de-doordash-landing-zn for incoming raw files and de-doordash-target-zn for processed files.
* In the first bucket, Uploaded the raw files which is the json data to the bucket. Second bucket stores the processed files from lambda function.

**Setting up SNS:**

* Creating a standard topic with name notifyme which sends an email notifications to the subscribers.
* Creating a subscription in the console and providing previously created topic ARN’s and assigning email protocol with the desired email address in which the email has to be sent.
* Upon successful creation of topic and subscription, the subscription email will be sent to the email address and prompts the user to subscribe for the SNS to receive notification

**Setting up IAM role for the lambda function:**

In the IAM console, create role and select Lambda as use case and providing the policies and providing name as lambdarole.

The policies which required for the task are:

* **AmazonAPIGatewayPushToCloudWatchLogs** – allows lambda to access the CloudWatch logs
* **AmazonS3FullAccess** – allows lambda to perform all the necessary actions on S3 buckets
* **AmazonSNSFullAccess** – allows lambda to access SNS and Send notification to the users
* **AmazonS3ObjectLambdaExecutionRolePolicy** – allows lambda to execute role on the objects in the S3.

**Creation of Lambda and configure lambda function code:**

* In the Lambda console, creating the lambda function with the name doordash and providing a runtime environment of python3.11
* Providing previously created IAM role in the role section and creating the function.
* After creation, adding a Pythonpandas311 layer to the existing function.
* In the code section, writing a code to read the Json file from the bucket and filter the records where status is “delivered”
* Writing the filtered dataframe to new Json file which target files. Json in de-doordash-target-zn
* And also sending the success or failure message to the SNS topic

This the Python code for the above tasks:

import boto3

import pandas as pd

def lambda\_handler(event, context):

    # Retrieve bucket names and file paths from the event

    source\_bucket = event['Records'][0]['s3']['bucket']['name']

    source\_key = event['Records'][0]['s3']['object']['key']

    target\_bucket = 'de-doordash-target-zn'

    sns\_topic\_arn = 'arn:aws:sns:us-east-1:038538804832:notifyme'

    # Read JSON file into Pandas DataFrame

    s3 = boto3.client('s3')

    obj = s3.get\_object(Bucket=source\_bucket, Key=source\_key)

    df = pd.read\_json(obj['Body'])

    # Filter records where status is "delivered"

    filtered\_df = df[df['status'] == 'delivered']

    # Write filtered DataFrame to S3

    target\_key = 'filtered\_data.json'

    s3\_resource = boto3.resource('s3')

    s3\_resource.Object(target\_bucket, target\_key).put(

        Body=filtered\_df.to\_json())

    # Publish success message to SNS topic

    sns = boto3.client('sns')

    sns.publish(

        TopicArn=sns\_topic\_arn,

        Message='Data processing completed successfully!'

    )

    return {

        'statusCode': 200,

        'body': 'Data processing completed successfully!'

    }

**AWS CodeBuild for the CI/CD:**

* Creating a public repository with name CICD-assign3 and adding readme file to explain the tasks in the repository.
* Adding the above Python code to the repository using the VS code terminal
* Here are some of the commands:
* **git clone** [**https://github.com/somesh1312/cicd-assign3.git**](https://github.com/somesh1312/cicd-assign3.git)
* **git checkout -b test** -creating test environment in the repository to test the code and pushing the code the main env
* **git status**
* **git add .**
* **git commit -m "imported lambda code for the task"**
* **git push origin test**
* In the GitHub creating a pull request and merging the pull request to main environment.

**Creating the project for CI/CD:**

* Creating the project with the project name as CICD-to-lambda and providing the description as automate the code for the lambda function.
* Providing tags as env and dev.
* In the source section, selecting GitHub and choosing desired working repository and providing source version as main
* Enable primary source webhook events, and in the event type choosing pull\_merge\_request.
* In the build spec section choosing buildspec file option and keeping all other options as default, and clicking create project.
* After creating, inside the project role change the permission to give the execution access to lambda and executing the S3 objects
* These are the Json file which should be added in the permission role

[{

    "Effect": "Allow",

    "Action": [

        "lambda:AddPermission",

        "lambda:RemovePermission",

        "lambda:CreateAlias",

        "lambda:UpdateAlias",

        "lambda:DeleteAlias",

        "lambda:UpdateFunctionCode",

        "lambda:UpdateFunctionConfiguration",

        "lambda:PutFunctionConcurrency",

        "lambda:DeleteFunctionConcurrency",

        "lambda:PublishVersion"

    ],

    "Resource": "arn:aws:lambda:us-east-1:038538804832:function:doordash"

},

{

    "Effect": "Allow",

    "Resource": [

        "arn:aws:s3:::de-doordash-landing-zn",

        "arn:aws:s3:::de-doordash-landing-zn/\*"

    ],

    "Action": [

        "s3:PutObject",

        "s3:GetObject",

        "s3:GetObjectVersion",

        "s3:GetBucketAcl",

        "s3:GetBucketLocation"

    ]

}]

Now, writing buildspec.yml file to automate the process in the lambda function and executing the role.

In the VS code, creating buildspec.yml :

version: 0.2

phases:

  install:

    runtime-versions:

      python: 3.11

    commands:

      - echo "Installing dependencies..."

  build:

    commands:

      - echo "Building Lambda function code..."

      # Add any build commands needed here (e.g., compilation, packaging)

  post\_build:

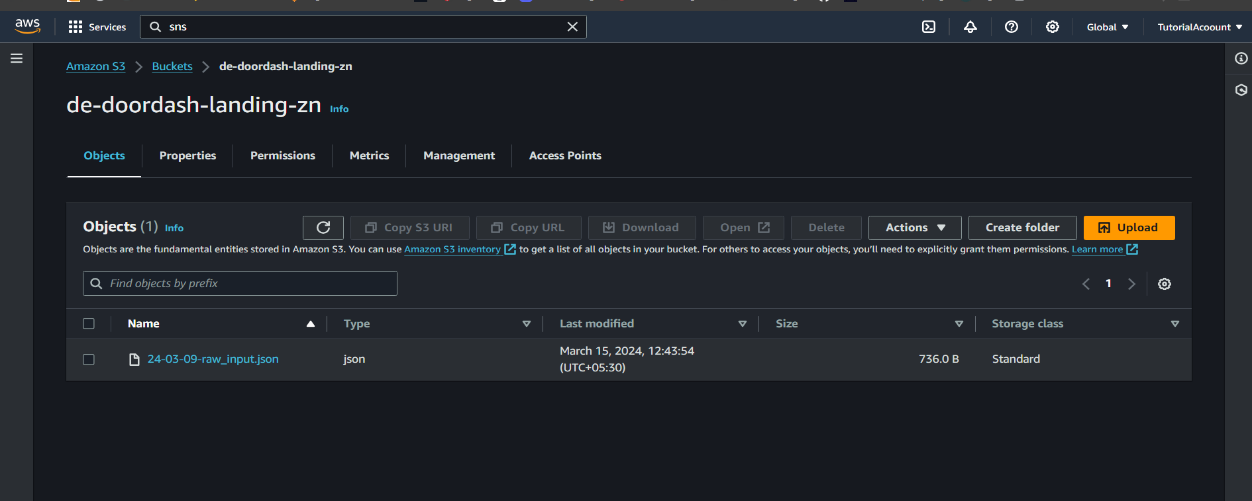
    commands:

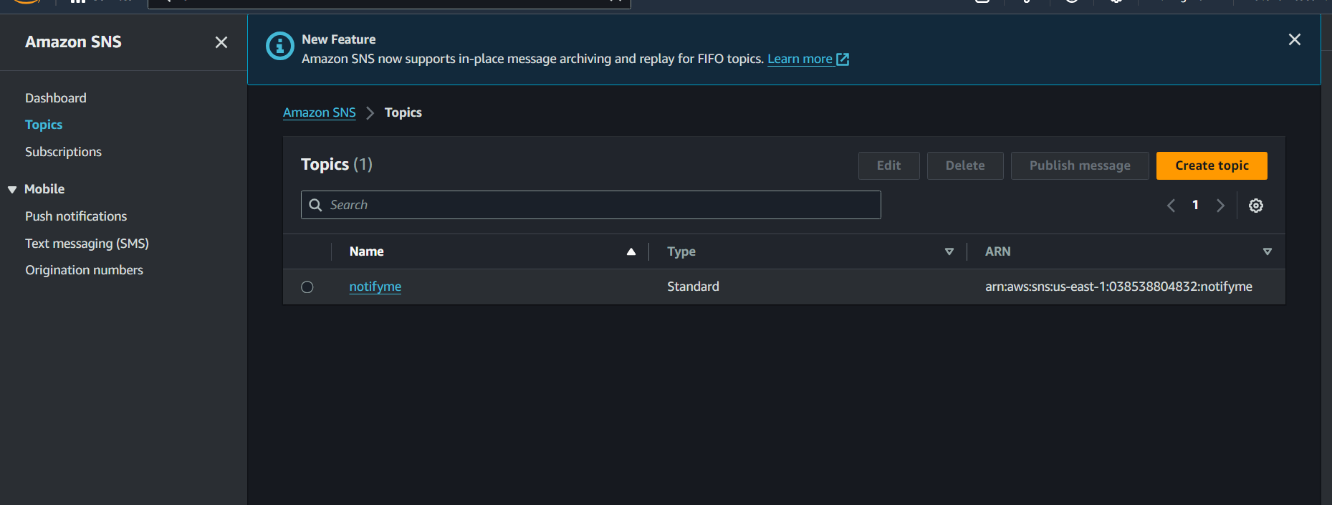
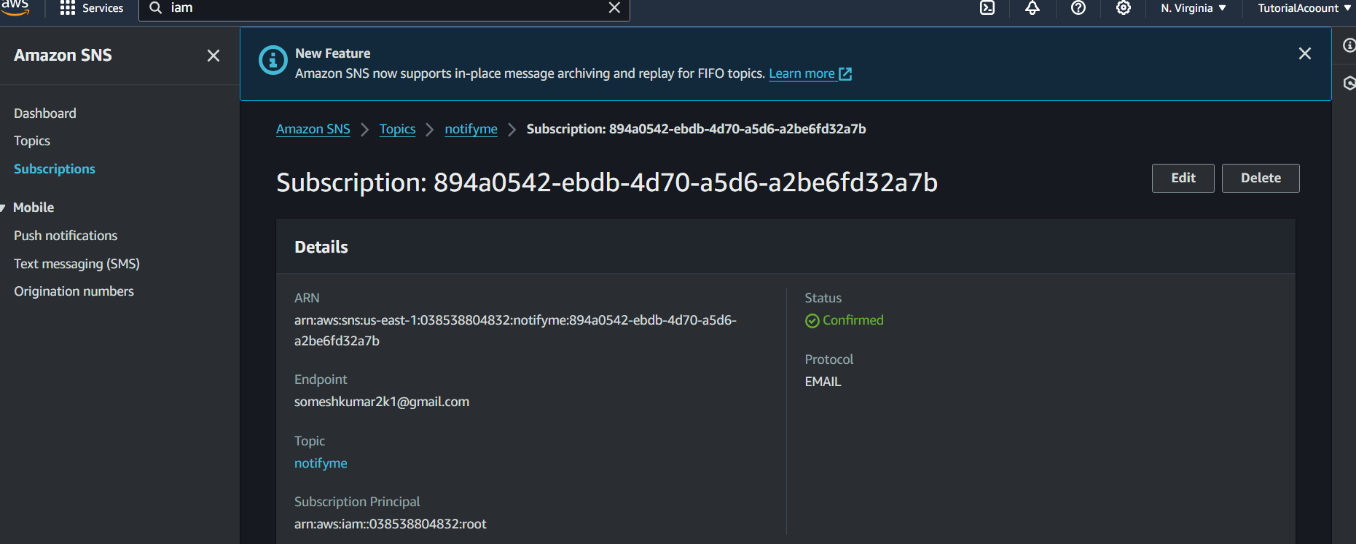
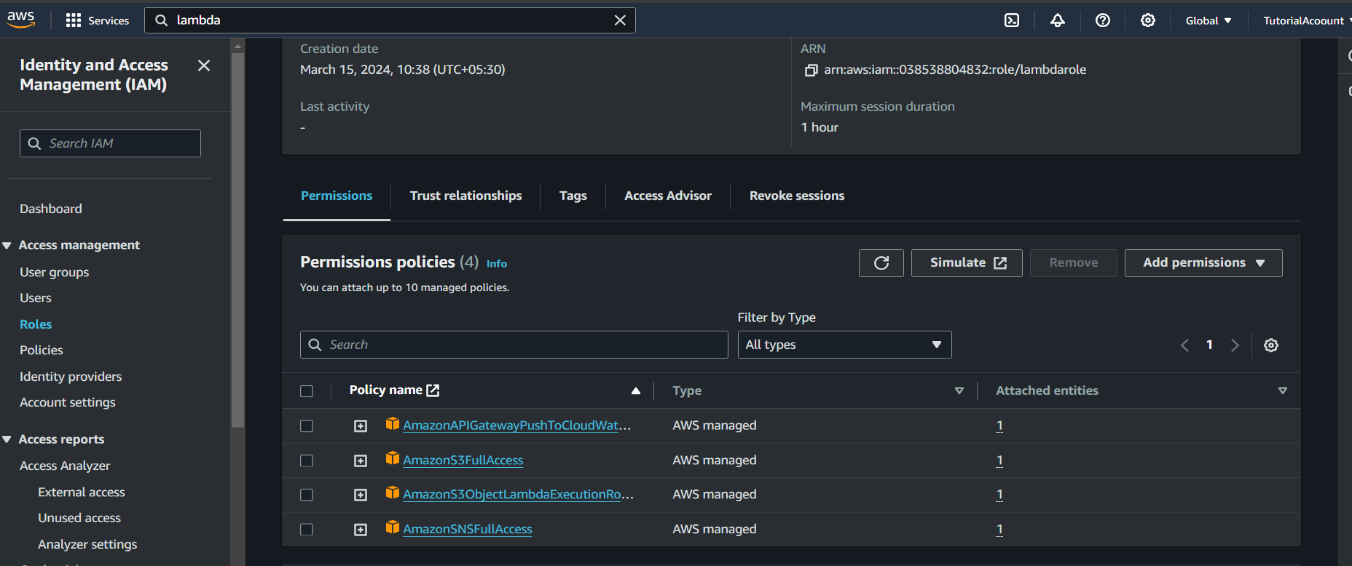
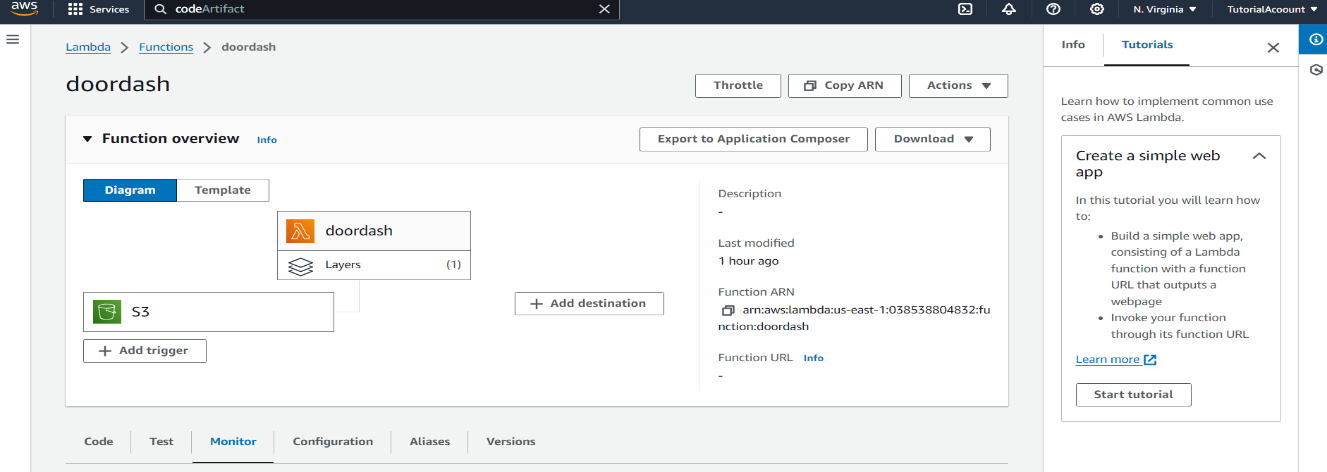
      - echo "Deploying Lambda function code..."

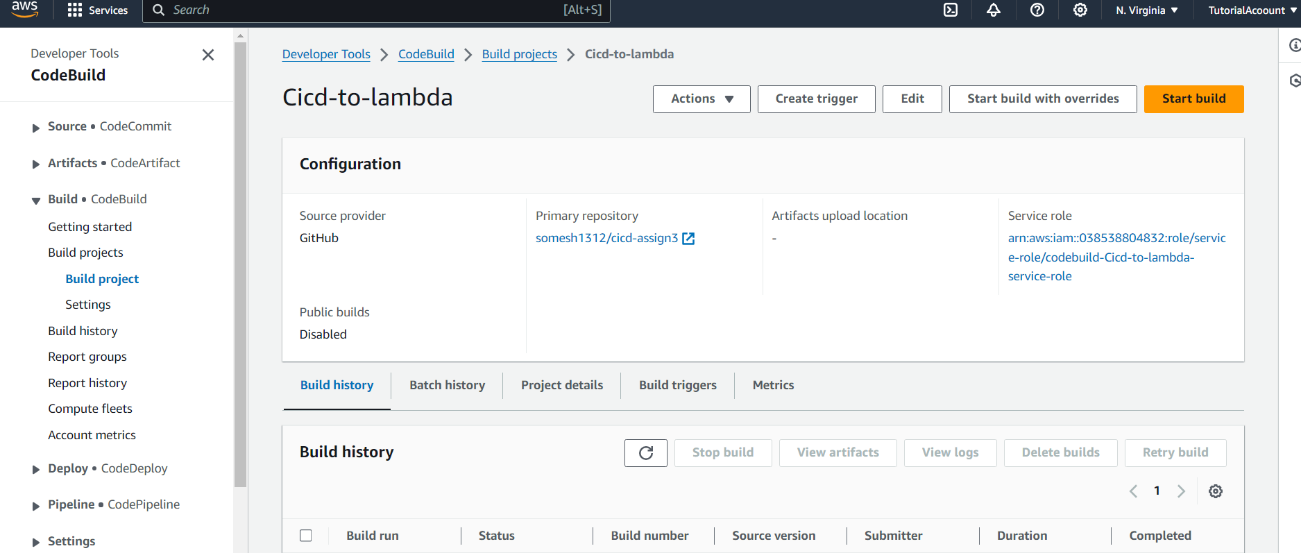
      - aws lambda update-function-code --function-name doordash --zip-file fileb://lambda\_function.zip

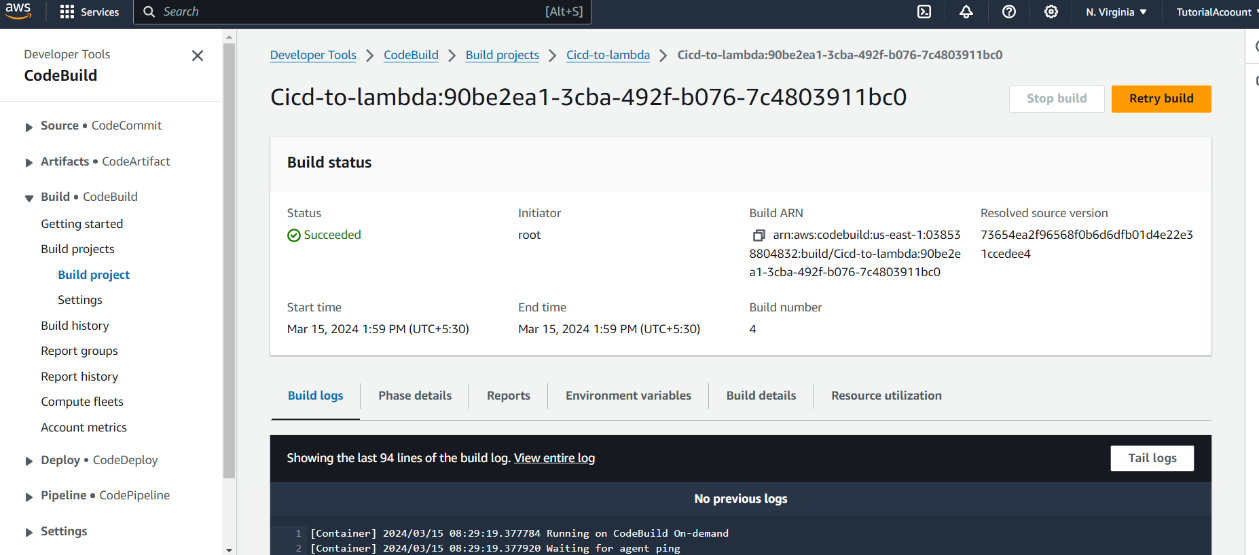
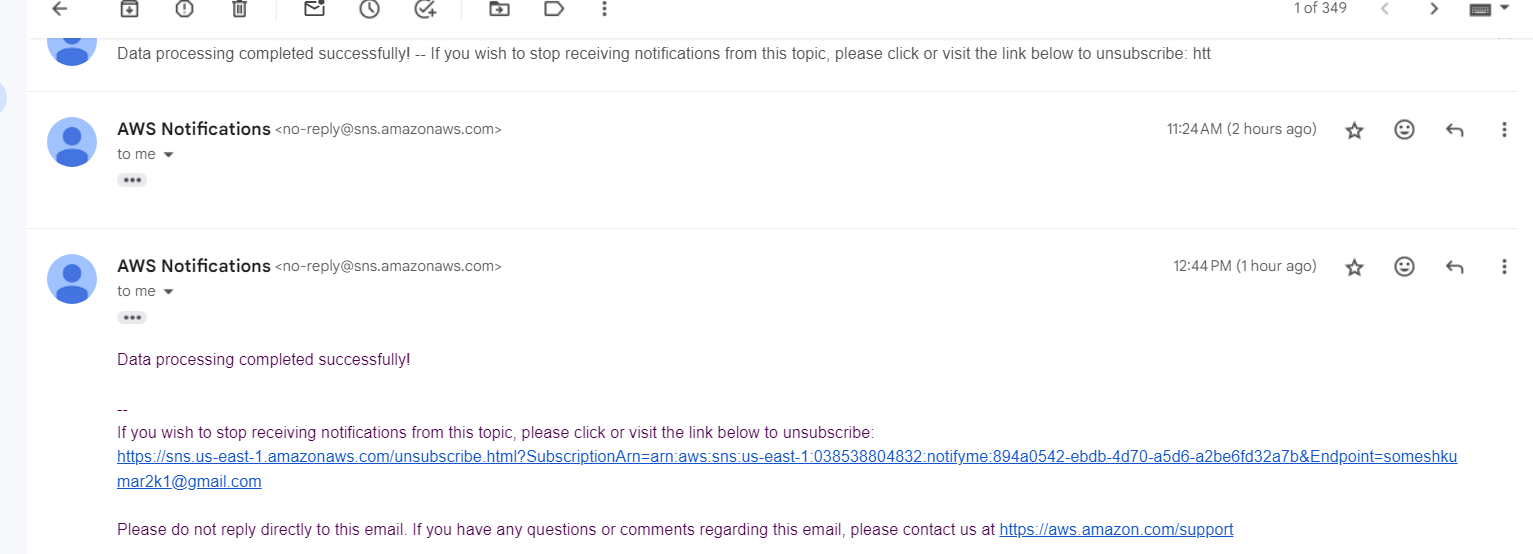
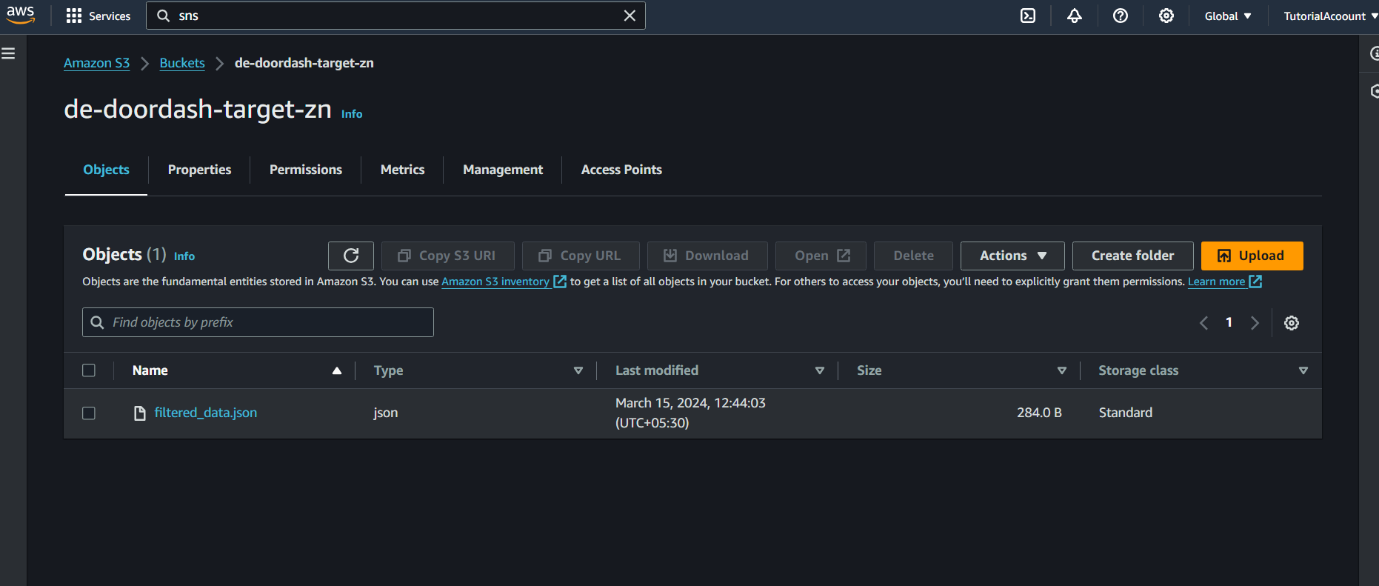
And pushing this file from test to the main environment. Upon pushing the it to the main environment, it will automatically build the lambda function code to automate the tasks.

**Here are some of the snapshots of the task:**

****

****

****

****