Manual Testing Documentation for AWS Lambda Function

Introduction

This documentation provides a detailed account of the manual testing process for the AWS Lambda function designed to list objects in a specified S3 bucket.

Test Objective for AWS Lambda Function

Goal:

To validate that the AWS Lambda function:

- Executes without errors.
- Accurately lists all objects in a specified S3 bucket.

Error Testing:

- Check response to invalid bucket names.
- Verify behavior under restricted permissions.
- Assess function's response to potential timeouts.

Create Files for Testing the AWS Lambda Function

Purpose

To generate a large set of test files for populating an S3 bucket, enabling comprehensive testing of the Lambda function's ability to list S3 objects. Prepare the Script:

Ensure you have the create_files.py script ready. This script is designed to generate a large number of text files in a specified directory.

Configure the Script:

- Determine the number of files you want to create for the test. The default in the script is 11,200 files, but this can be adjusted based on your testing needs.
- Set the target directory path where the files will be created. This could be a local directory on your machine.

Run the Script: Execute the script by running create_files(directory_path, num_files) in your Python environment.

Verify File Creation: After the script completes execution, check the target directory to ensure that the specified number of files has been created.

Each file should be named in the format file_1.txt, file_2.txt, ..., file_n.txt and contain a line of text.

Usage in Testing : These files will act as the data for your S3 bucket, allowing you to test the Lambda function's ability to list objects in a bucket.

By using a significant number of files, you can effectively simulate a realistic scenario and assess the performance of your Lambda function under typical operational conditions.

Next Steps: After creating these test files, the next step in your testing documentation would be to upload these files to your S3 bucket, which can be done manually through the AWS Management Console or programmatically using a script.

Setting Up AWS Access Keys for Python

Run Python scripts securely with AWS services by setting AWS Access Keys as environment variables.

Steps

Create Access Keys:

- Generate a new access key pair in AWS IAM.
- Record the Access Key ID and Secret Access Key.

Set Environment Variables:

- Windows CMD: Use the **set** command.
- PowerShell: Use **$env:VariableName syntax.**
- Linux/Mac: Use the **export** command.
- Example for PowerShell: **$Env:AWS_ACCESS_KEY_ID='your_access_key_id',**
  $Env: AWS_SECRET_ACCESS_KEY= ‘ your secret access key id’

Use in Python:

- Import and use the boto3 library in your script.
- boto3 will automatically detect and use the credentials from the environment variables.

Best Practices:

- Rotate keys periodically for security.
- Assign only the permissions necessary for the tasks at hand.
- Avoid using the root account keys for regular tasks.

Uploading a Local Directory to AWS S3 Using Python

To use the upload_directory_to_s3 function to upload files from a local directory to your AWS S3 bucket using Python, follow these steps:

Set Up Your AWS Credentials: Ensure your AWS credentials are configured on your machine. These can be set up in the AWS credentials file or through environment variables.

Install boto3: Make sure you have boto3 installed. If not, install it using pip: **pip install boto3**

Set Your Variables:

- Determine the local directory path you want to upload from. For example: 'path/to/local/directory'.
- Choose or create an S3 bucket where the files will be uploaded. For example: 'your-s3-bucket-name'.

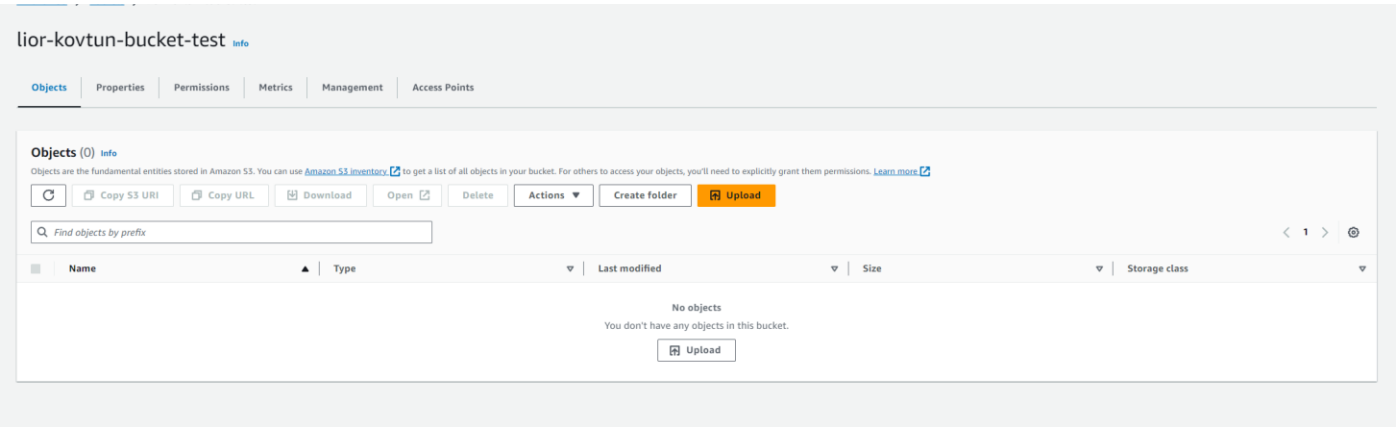
Use the Function: Call the upload_directory_to_s3 function with your specified local directory and S3 bucket name. For example: **upload_directory_to_s3('path/to/local/directory', 'your-s3-bucket-name')**

Run the Script: Run your Python script. The function will upload all files from the specified directory to your S3 bucket. It will print the status of each file upload, and any errors encountered.

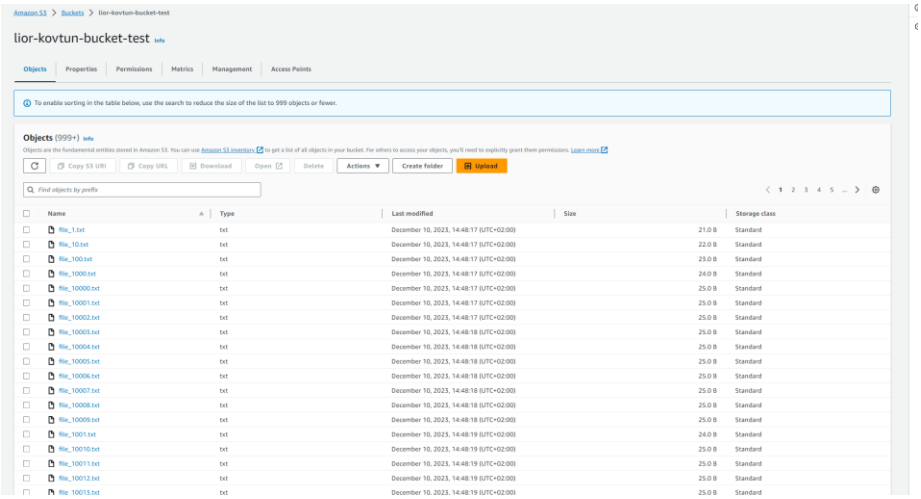
Remember:

- The function uploads files maintaining their names but not their directory structure. All files will be placed in the root of the S3 bucket.
- Make sure the AWS IAM role/user you're using has the necessary permissions to upload files to the specified S3 bucket.

Before running the script: upload_directory_to_s3('path/to/test/files/directory', 'lior-kovtun-bucket-test')

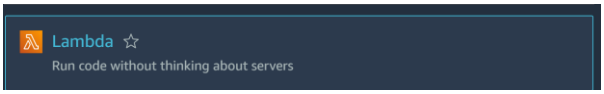


After running the script:

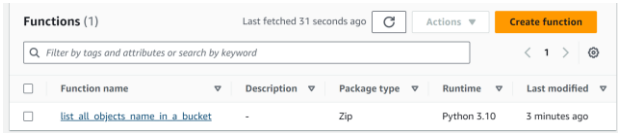


Creating Manual Test Events for AWS Lambda

1. **Access the Lambda Function:**
    - Navigate to the AWS Management Console.
    - Locate the lambda service and select it.

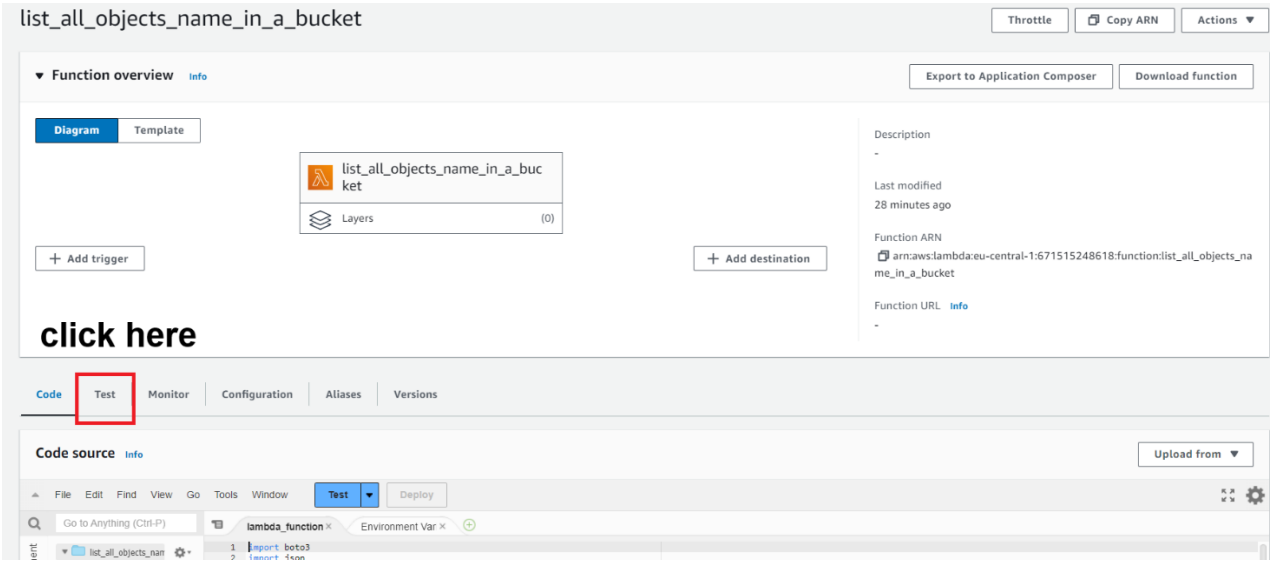


    - Select list_all_objects_name_in_a_bucket.



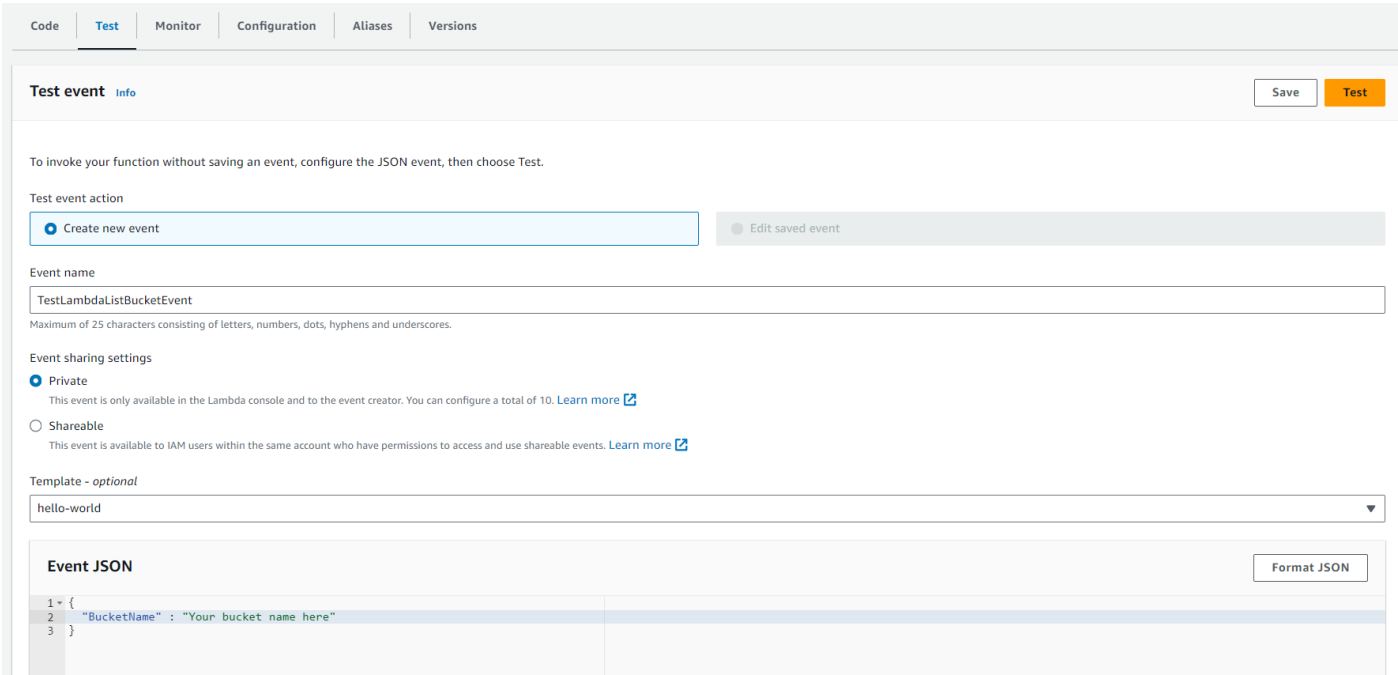
‘

2. **Access Test Event Configuration:** Click on the 'Test' tab above the function code editor to configure a new test event.



3. **Create New Test Event:**
    - Select 'Create new event'.
    - Enter a descriptive Event name, like TestLambdaListBucketEvent.
4. **Set Up Event JSON:** In the 'Event JSON' editor, input a JSON object that your Lambda function expects. Your function is programmed to look for the BucketName within the event object. Structure your JSON accordingly:

Replace “Your bucket name here” with the actual bucket name you're testing. This key-value pair tells the Lambda where to find the bucket name within the event. Then select “save” on top right.



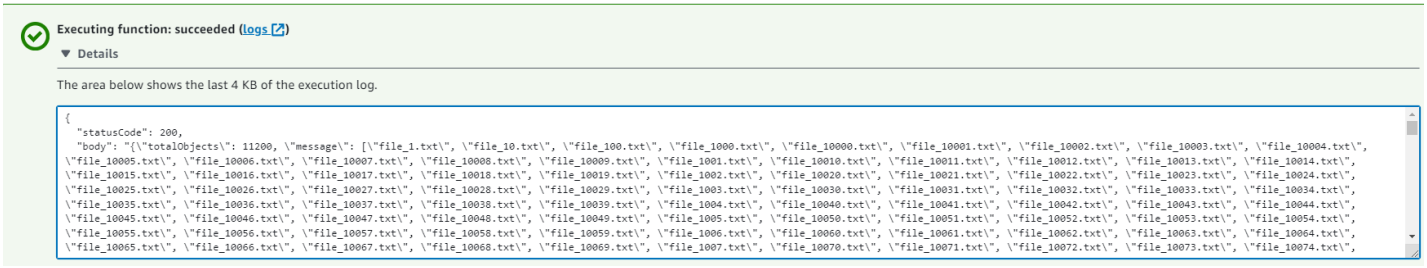
Execute Test:

- Save the event.
- Click 'Test' to invoke the function with your event.

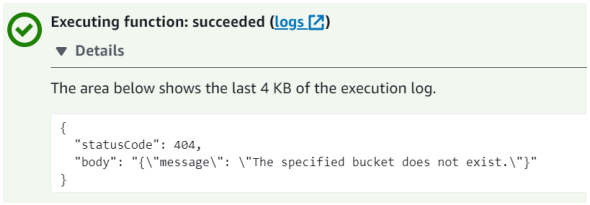
Results:

I used the create_files.py script to generate 11,200 text files. These files were then uploaded to the lior-kovtun-bucket-test S3 bucket using the upload_s3_files_to_bucket.py script."

TEST1 : Valid execution. Note: If the execution is successful, the response will also include totalObjects, which represents the total number of objects in the bucket."



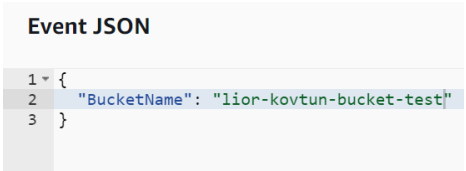
TEST2: Bucket name input that doesn't exist.

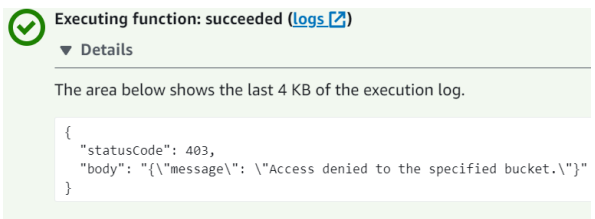


CloudWatch service log:

```
Error response: The specified bucket does not exist.
```

TEST3: Removing the S3 permissions from the lambda role.





CloudWatch service log:

```
Error response: Access denied to the specified bucket.
```