A Lambda function that triggers when objects are uploaded or deleted in an S3 bucket, and then inserts that information into a DynamoDB table.

Step 1: Create a DynamoDB Table

Go to the AWS DynamoDB console

Click "Create table"

Name your table (e.g., "S3ObjectLog")

Set the partition key to "Object Key" (String)

Add a sort key called "EventTime" (String)

Use default settings for the rest and create the table

Step 2: Create a Lambda Function

Go to the AWS Lambda console

Click "Create function"

Choose "Author from scratch"

Name your function (e.g., "S3ObjectLogger")

Select Python 3.9 (or the latest available Python runtime)

For execution role, choose "Create a new role with basic Lambda permissions"

Click "Create function"

Step 3: Configure Lambda Function Code

Replace the default code with the following Python script:

```
import json
import boto3
```

```
import os
from datetime import datetime

def lambda_handler(event, context):
    s3 = boto3.client('s3')
    dynamodb = boto3.resource('dynamodb')

    table_name = os.environ['DYNAMODB_TABLE']
    table = dynamodb.Table(table_name)

    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        object_key = record['s3']['object']['key']
        event_time = record['eventTime']
        event_name = record['eventName']

        # Determine if it's a creation or deletion event
        if event_name.startswith('ObjectCreated'):
            action = 'created'
        elif event_name.startswith('ObjectRemoved'):
            action = 'deleted'
        else:
            print(f"Unsupported event type: {event_name}")
            continue

        # Insert item into DynamoDB
        try:
```

```
        response = table.put_item(
            Item={
                'ObjectKey': object_key,
                'EventTime': event_time,
                'Bucket': bucket,
                'Action': action
            }
        )
        print(f"Successfully logged {action} event for {object_key}")
    except Exception as e:
        print(f"Error logging event: {str(e)}")

return {
    'statusCode': 200,
    'body': json.dumps('Function executed successfully!')
}
```

Step 4: Configure Environment Variables

Scroll down to the "Environment variables" section in your Lambda function

Add a new variable with key "DYNAMODB_TABLE" and value as your DynamoDB table name (e.g., "S3ObjectLog")

Click on Deploy

Step 5: Configure Lambda Execution Role

Go to the "Configuration" tab and click on "Permissions"

Click on the role name to go to the IAM console

Add these managed policies to the role:

AmazonS3ReadOnlyAccess

AmazonDynamoDBFullAccess

Step 6: Configure S3 Trigger

In your Lambda function, go to the "Configuration" tab and select "Triggers"

Click "Add trigger"

Select "S3" as the trigger type

Choose your S3 bucket

For event types, select "All object create events" and "All object delete events"

Acknowledge the recursive invocation warning if it appears

Click "Add"

Step 7: Test the Function

Upload a file to your S3 bucket

Check the CloudWatch logs for your Lambda function to ensure it ran successfully

Go to your DynamoDB table and verify that a new item was added with the correct information

Additional Notes:

This setup will log both object creations and deletions.

The DynamoDB table uses the object key as the partition key and the event time as the sort key. This allows you to track multiple events for the same object over time.

Make sure your S3 bucket and Lambda function are in the same region for best performance.

For large-scale applications, consider using DynamoDB streams or AWS Step Functions for more complex workflows.

Remember to monitor your Lambda function's performance and adjust the timeout and memory settings as needed based on your specific use case and the size of your S3 objects