CSE-3250 Cloud Computing

HW04 - S3 Events to Lambda / DynamoDB

Objective

You will create a lambda function that is able to insert records into a DynamoDB table when it is triggered by the event of a file being uploaded into a S3 bucket.

Instructions

- Git
    - Create folder named "HW04" in your repo and store all your files in this folder.
    - Create a feature-hw04 branch for all your work. (Do not commit directly to "main".) Submit and complete pull requests when milestone progress is completed.
    - Use git add/commit/push to develop frequently. (This is a requirement, not an option.)
    - You HW04 folder will contain the python function you create below.
- Create a new private S3 bucket
    - Name it starting with the string "hw04-"
    - Enable bucket versioning
    - Set an appropriate bucket policy to complete the coding and testing of this homework.
- Create a DynamoDB table
    - Name your table beginning with "hw04-"
    - Your table will contain the list of all files uploaded to a S3 bucket. Attributes in the table should include: file name, file size, upload date, file ARN, and eTag.
    - You must decide what your "partition key" will be, and whether you need/want a "sort key" defined. (Remember, they cannot be changed once the table is created. If you want/need to change your keys in the future, you must delete the table and recreate it.)
- Create a lambda function
    - Create a lambda function in AWS
        - Name it so it starts with the string "hw04-"
        - Specify Python (python version is your choice)
        - Select the "existing execution" role as "LabRole"
        - Add a Trigger. Specify your hw04 S3 bucket as the trigger source. The trigger should occur whenever a file is uploaded to that S3 bucket.
        - **EVERY STUDENT** must read this short tutorial and make sure any lambda function created in the Learner Lab limits the reserved concurrency. (You will be graded on this.)
    - The code of the lambda function should determine the file metadata (e.g. filename, file size, upload timedate, bucket ARN, and eTag) and write that information to the DynamoDB table (created above).

    - In your lambda function, log to CloudWatch the values you are writing to DynamoDB. You may also log additional information to Cloudwatch that helps you debug your code.
- Manual testing
    - Test your configuration and code on AWS by manually uploading a few files and making sure the correct records end up in DynamoDB. This is a basic type of testing that every developer will start with. You are not handing in any proof of manual testing, but you will save a lot of time testing manually before trying to run tests in GHA.
- CICD
    - After you have initially created your lambda function using the Management Console, you will create a GitHub Actions workflow to deploy your python code to lambda whenever it changes in the git repo.
    - Note: follow the tips in the lambda tutorial on naming your python file and function within it.
    - Automated testing in GHA
        - Use what you've learned so far about GHA in previous homeworks. And take a look at the GHA tutorials in my repo.
        - Create pytests that will upload objects to S3 and thus trigger your lambda code.
        - Use assert statements in your pytests to check the dynamodb table contents and verify that your test did what you expected it to do.

Extra credit opportunities

- Create a second python function that is triggered whenever an object is deleted from your bucket.
    - The function should create a CSV file containing a list of the files currently in your S3 bucket after the deletion. Each row of the CSV could contain the file name, file size, upload date, file ARN, and eTag for each of the files.
    - The function should writes the csv file to the same S3 bucket it is listing files. (Yes, the upload of the csv file to the S3 bucket should cause your first function to get triggered and write a record to the DynamoDB table. But since your second lambda function is only triggered on a delete action, the loop will end there.)
    - The CSV file can be named anything you wish, but always use the same name and overwrite the file if it exists.
    - Logging to CloudWatch is optional.
    - Deploy your code to lambda using GHA

Deliverables

- Make sure to "TAG" all AWS resources you create for this assignment with "hw=04" (key = "hw", value = "04"). Do not modify these HW04 resources on AWS in ANY way after the due date!
- Create a design diagram for your solution using DrawIO. (You may use the online version of DrawIO or the downloaded version.) In your repo, submit your DrawIO source file along with a PDF of that diagram. So, two additional files in your github repo in the HW04 folder. You can see some examples of design diagrams in the lecture slides from the first Mod4 lecture.
- Create a PDF and submit to Gradescope:
  - Direct link to your git repo for HW04
  - List of all websites and online resources and prompts referenced for coding
  - Name of any Cloudwatch Log Group(s) generated by your code
  - Design diagram of your solution. Show all the AWS components you created and how they interact with one another. (Put the DrawIO source file in your hw04 folder of your git repo.)
  - Specify whether you attempted the extra credit

Example Grading Rubric

Fully functioning code & AWS configuration	50%
GHA - automated pytests	20%
GHA - deploy python to lambda	10%
Design diagram	10%
Cloudwatch Logs	5%
GIT branches, commits, PRs	5%
EC Opportunity	Up to 20%