Cloud Computing Exercise #21

Lambda Function Manipulating Files in S3 Buckets

A. Preparation

1. Sign in to your AWS account as the non-root admin user.
2. In this lab, you will create two S3 buckets: an input bucket, and an output bucket. Then, you will create a Lambda function that is automatically triggered when a file is uploaded into the input bucket. The Lambda function will transform this file by replacing all upper-case characters by lower-case characters and upload the transformed file into the output bucket.

B. Create two S3 buckets

1. Go to S3 and create two buckets; one should hold the input files and the other should hold the output files. Choose appropriate names for the buckets (e.g. “<username>-input-bucket” and “<username>-output-bucket”, where <username> is your user name). Note the ARNs of your buckets as you will need them when writing the IAM policy in the next step.

C. Create a role for the Lambda

1. Create an IAM policy (e.g. called “lambda-policy”) that will allow the Lambda to read (download) any file/object from the input bucket identified by its ARN, and write (upload) any file/object to the output bucket identified by its ARN. The policy should be similar to what we have done before. You can check “lambda\_policy” uploaded in the modules section to refer the policy.
2. Create a role for the Lambda function (Trusted entity as AWS service) and give a name to your role (e.g. “lambda-role”). Associate the IAM policy you just created with this role. You will also need to associate the built-in policy called “AWSLambdaBasicExecutionRole” with your role. This policy will allow the Lambda to write into CloudWatch logs (that is, allow you to use Python’s print() function or the logging module to produce output in the log files). Thus, you will have two policies attached to your role: the one you just defined (“lambda-policy”), and the policy “AWSLambdaBasicExecutionRole”.

D. Create a Lambda function

1. Go to the Lambda dashboard (AWS Lambda/Dashboard) and create a Lambda function from scratch named “MyLambda” using the latest Python runtime (Python 3.9). You need to associate your Lambda function with the previously created role. Click on the “Change default execution role” section, select “Use an existing role”, and provide the name of the role you have created for the Lambda function (“lambda-role”).
2. In the code composer window, write a Lambda function that imports the boto3 module (API to interact with the AWS services), the os module (API to issue OS-level commands, e.g. creating a directory on the local file system), and the standard Python logging module. The event handler code should
   * Wrap the rest of the code in a try/except structure catching any Exception raised. The exception handler code should log the error using the Python standard logging facility with logging level ERROR.
   * Obtain an S3 client object, which is an instance of the AWS S3 client class S3.Client.
   * Process the input parameters from the event variable by iterating over the S3 event records and retrieving the name of the bucket that generated the event (the value corresponding to the key “name” in the dictionary, which is itself a value corresponding to the key “bucket”, which is a value corresponding to the key “s3” in each record). This is the input bucket that has triggered the Lambda execution when a file was uploaded to it. The code should also retrieve the S3 object’s key (which is a value corresponding to the key “key” in a dictionary which is a value corresponding to the key “object”). This is the key corresponding to the newly uploaded file (object) that triggered the event execution for the Lambda. You can look at an example JSON S3 notification event at https://docs.aws.amazon.com/lambda/latest/dg/with-s3.html, which shows the data hierarchy in the S3 event descriptor with these two items highlighted.
   * Download the file from S3 and save it in the /tmp/ directory on the local file system using the S3 object’s key as the file name. You can use the download\_file() function of the S3 client object for this purpose. Note that the storage in the local file system is ephemeral: different invocations of the Lambda function should not rely on data/state stored there.
   * Open the downloaded (input) text file for reading, and open a new (output) text file for writing. The output file’s name should be the input file’s name with the “lowercase-” string prepended. E.g. if the downloaded file’s name is “example.txt”, the output file name should be “lowercase-example.txt”.
   * Iterate over the lines in the input text file and change the characters to lower case only. E.g. the line “Hello, I am a line!” should become “hello, i am a line!”. The lower case version of each line should be written into the output file. Thus, the output file’s content will be identical to the input files content, except for changing the upper case letters to lower case letters. At the end of this process, both the input and output files should be closed.
   * Upload the output file (the one containing the lower-case version of the input file) from the /tmp/ directory on the local file system to the S3 output bucket (named “<username>-output-bucket”) with the associated key obtained by prepending “lowercase-” to the key name of the downloaded object’s key. You can use the upload\_file() function of the S3 client object for this purpose.
   * Delete the input and output files from the /tmp/ directory in the local file system. You can use the remove() function of the os module to do that. The documentation for the os module can be found here: https://docs.python.org/3/library/os.html.
3. Deploy the Lambda function.

E. Test the Lambda function manually

1. Create a text file on your local machine (e.g. test01.txt), which has some mixed-case characters in it, e.g.

This Is My TeSt FiLe.

LeT Us Do this!!!

Upload this text file to your input bucket (“<username>-input-bucket”).

1. Create a mock event data JSON document with the bucket and object parameters corresponding to your uploaded file. Configure a test event called “TestEvent” for your Lambda and select the “s3-put” event template. Modify the bucket name and the object key parts in the JSON document so that they match the name of your input bucket and the object (file) you just uploaded.
2. Test your Lambda function. If the test is successful, you should see an execution result message similar to this:

**Test Event Name**

TestEvent

**Response**

null

**Function Logs**

START RequestId: f5180788-ea14-4e79-a090-e27e7d2e3429 Version: $LATEST

END RequestId: f5180788-ea14-4e79-a090-e27e7d2e3429

REPORT RequestId: f5180788-ea14-4e79-a090-e27e7d2e3429 Duration: 1973.60 ms Billed Duration: 1974 ms Memory Size: 128 MB Max Memory Used: 70 MB Init Duration: 248.77 ms

**Request ID**

f5180788-ea14-4e79-a090-e27e7d2e3429

If you go to S3 and look at the content of your output bucket (“<username>-output-bucket”), you should see a new file appearing in it, called “lowercase-test01.txt” containing the transformed text file. If you download and open the file, you should see only lower-case characters in it.

F. Configure the S3 event trigger

1. The last step is to configure an S3 event trigger, which will automatically invoke the Lambda function whenever an object is uploaded to the input bucket.
2. Create a resource-based policy for the Lambda function, which will allow the S3 service to invoke it. Go to your Lambda function, select the “Configuration” tab, Click on the “permissions” tab on the left and scroll down to the “Resource-based policy” section, and click on “Add permissions”. Select “AWS Service” to grant permission to, and select “S3” as the service. Fill out the form with your own account number, the ARN of the input bucket (“<username>-input-bucket”), set the action to “lambda:InvokeFunction", and provide an arbitrary statement ID. Create the resource-based policy, which should appear on the list of resource-based policy statements.
3. Create an event notification for the S3 service when a new file is uploaded into your input bucket. Go to your input bucket (“<username>-input-bucket”) in S3, and select the “Properties” tab. Scroll down to the “Event notifications” section, and choose “Create event notification”. Give your event a name (e.g. “lambda-trigger”), and add a suffix “.txt” so that the event would only be triggered by files having this extension (i.e. text files). Under “Event types”, select “All object create events”, and under “Destination”, select “Lambda function”. On the drop-down list, find and select your Lambda function (“myLambda”), and save the changes.

G. Test the Lambda with automatic S3 event trigger

1. Create a second test file (e.g. test02.txt) with some mixed-case letters, and upload the file to your input bucket. A new file should appear in your output bucket (e.g. lowercase-test02.txt) having only lower-case characters in it.

H. Clean up after yourself

1. Delete your Lambda function, the execution role (lambda-role), and the associated policy (lambda-policy). Finally, delete all files in your buckets and delete your input and output buckets.
2. Log out of AWS.