[Instructions] Hello World using AWS Lambda

Let us start with Hello World using AWS Lambda. We will use Python3 as a runtime environment.

* Go to AWS Console and use the Lambda template.
* Deploy and Test and make sure it is successful.
* Here are the default names that are used.
  + Program Name: **lambda\_function**
  + Function Name: **lambda\_handler**
* We can change the names of the file, function etc. However, we need to ensure that Handler details as part of Runtime Settings are updated.
* Let us rename the file name to **main.py** and validate. We will revert the name back to **lambda\_function.py**.

[Instructions and Code] Setup Project for local development

Here are the steps using which we can set up a project. We will use this project to download the GitHub Activity files in incremental fashion.

* Create folder for the project - **ghactivity-downloader**
* Create virtual environment for this project - **ghad-venv**

1. python3 -m venv ghad-venv
2. source ghad-venv/bin/activate

* We will install boto3 in the default location within the virtual environment. We need not include it as part of the bundle that will be deployed as a lambda function.

1. pip install boto3
2. pip install requests

* We will install requests as part of **lambdalib** folder. It will be included as part of the bundle that will be deployed as a lambda function.

1. mkdir ghalib
2. pip install requests -t ghalib

* We can create **lambda\_function.py** with the default code we got when we get started.

1. import json
3. def lambda\_handler(event, context):
4. # TODO implement
5. return {
6. 'statusCode': 200,
7. 'body': json.dumps('Hello from Lambda using gha downloader!')
8. }

* We will create an additional file **lambda\_validate.py** and have this piece of code so that we can run locally.

1. from lambda\_function import lambda\_handler
3. res = lambda\_handler(None, None)
4. print(res)

* We can run **lambda\_validate** to validate locally.

python lambda\_validate.py

[Instructions and Code] Deploy Project to AWS Lambda console

Let us understand how we can deploy the locally developed Lambda Function to AWS Lambda Web Console.

* You need to build the zip file with the source code.

zip -r ghactivity-downloader.zip lambda\_function.py

* Use AWS Lambda Web Console to upload the Zip file.
* You can review the source code in Python scripts and test the function by creating a test event.

[Instructions and Code] Develop download functionality using requests

Let us develop the download functionality using requests and validate locally. We will understand how to integrate 3rd party libraries as part of the next topic using requests as an example.

* Develop the base functionality to download the zip file using requests library. I have created a new program called download.py for this.

1. import requests
3. def download\_file(file):
4. res = requests.get(f'https://data.gharchive.org/{file}')
5. return res

* Refactor the code as part of lambda\_function.py to invoke the new function and also to capture the response.

1. import json
2. from download import download\_file
4. def lambda\_handler(event, context):
5. download\_res = download\_file('2021-01-29-0.json.gz')
6. return {
7. 'statusCode': download\_res.status\_code,
8. 'body': json.dumps('Download status code')
9. }

[Instructions and Code] Using 3rd party libraries in AWS Lambda

Let us understand how to include 3rd party libraries as part of the zip file that will be deployed as lambda function.

* You need to ensure that all the 3rd party libraries which are supposed to be deployed along with lambda functions are downloaded to a folder. In our case it is ghalib.
* We need to go to the folder to build the zip file. Make sure the zip file is created in the base directory of the project and update the zip file with source code.

1. rm ghactivity-downloader.zip
2. cd ghalib
3. zip -r ../ghactivity-downloader.zip .
4. cd ..
5. zip -g ghactivity-downloader.zip lambda\_function.py download.py

* We can upload the zip file to AWS Lambda console and validate successfully. Make sure to increase memory size to 512 MB as demonstrated.

[Instructions and Code] Validating s3 access for local development

Let us validate s3 access for local development. We need to have the appropriate credentials to access s3 bucket from the local development environment.

* Import required libraries.
* Configure environment variable **AWS\_PROFILE** to appropriate value.
* Develop the code to upload the contents of the zip file from GitHub archive to s3. We will create a new file called **upload.py**.

1. import os
2. import boto3
3. import requests
5. os.environ.setdefault('AWS\_DEFAULT', 'itvgithub')
7. s3\_client = boto3.client('s3')
9. file = '2021-01-29-0.json.gz'
10. res = requests.get(f'https://data.gharchive.org/{file}')
12. upload\_res = s3\_client.put\_object(
13. Bucket='itv-github',
14. Key='2021-01-29-0.json.gz',
15. Body=res.content
16. )
18. print(upload\_res)

[Instructions and Code] Develop upload functionality to s3

Let us understand how to upload the GitHub activity file to s3.

* We need to ensure that the role using which lambda function is being executed has permission on the target bucket.
* Let us create a sandbox folder under **itv-github** bucket.
* We need to develop a new function as part of a new program which takes the response object and uploads it as an object in s3.

1. import boto3
3. def get\_client():
4. return boto3.client('s3')
6. def upload\_s3(body, bucket, file):
7. s3\_client = get\_client()
8. res = s3\_client.put\_object(
9. Bucket=bucket,
10. Key=file,
11. Body=body
12. )
13. return res

* We also need to update the logic in the lambda handler to call the function which uploads the response as an object in s3.

1. import os
2. from download import download\_file
3. from upload import upload\_s3
5. def lambda\_handler(event, context):
6. file = '2021-01-29-2.json.gz'
7. download\_res = download\_file(file)
8. bucket = os.environ.get('BUCKET\_NAME')
9. environ = os.environ.get('ENVIRON')
10. if environ == 'DEV':
11. print(f'Running in {environ} environment')
12. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
13. upload\_res = upload\_s3(
14. download\_res.content,
15. bucket,
16. file
17. )
18. return upload\_res

* We can rebuild the zip file, upload and run.

1. cd ghalib
2. zip -r ../ghactivity-downloader.zip .
3. cd ..
4. zip -g ghactivity-downloader.zip download.py lambda\_function.py

[Instructions and Code] Validating using AWS Lambda Console

Let us validate using AWS Lambda Console by uploading the zip file.

* Upload the zip file using AWS Lambda Console.
* Add required environment variables.
* The test will fail as there are no required permissions granted to Lambda function.

1. import os
2. from download import download\_file
3. from upload import upload\_s3
5. def lambda\_handler(event, context):
6. file = '2021-01-29-2.json.gz'
7. download\_res = download\_file(file)
8. bucket = os.environ.get('BUCKET\_NAME')
9. file\_prefix = os.environ.get('FILE\_PREFIX')
10. environ = os.environ.get('ENVIRON')
11. if environ == 'DEV':
12. print(f'Running in {environ} environment')
13. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
14. upload\_res = upload\_s3(
15. download\_res.content,
16. bucket,
17. f'{file\_prefix}/{file}'
18. )
19. return upload\_res

[Instructions] Run using AWS Lambda Console

We need to grant required permissions on s3 bucket to the role associated with Lambda Function. We can take care of it using the policy which has permissions on s3 bucket.

* Identify the role associated with the lambda function by going to the permissions tab.
* Go to the role and attach the policy to the role associated with Lambda Function.
* We will also copy the files to the sandbox folder within the s3 bucket. Hence, let us create the folder, refactor the code and then deploy.
* Once deployed we can test using AWS Lambda Console and then validate the logs as well as the target folder within s3.

[Instructions and Code] Validating files incrementally

Let us implement the logic to validate files incrementally. We will use the current day’s beginning hour as baseline.

* We can convert the date and hour part of the file to timestamp.
* Once we get the timestamp we should be able to use **timedelta** to add 1 hour in each iteration.
* Using the new date and time, we should be able to generate the next file. We can check whether the file is already available or not.

1. from datetime import datetime as dt
2. from datetime import timedelta as td
3. import requests
5. next\_file = '2021-01-30-0.json.gz'
7. while True:
8. res = requests.get(f'https://data.gharchive.org/{next\_file}')
9. if res.status\_code != 200:
10. break
11. print(f'The status code for {next\_file} is {res.status\_code}')
12. dt\_part = next\_file.split('.')[0]
13. next\_file = f"{dt.strftime(dt.strptime(dt\_part, '%Y-%M-%d-%H') + td(hours=1), '%Y-%M-%d-%-H')}.json.gz"

[Instructions and Code] Reading and Writing Bookmark using s3

Let us go through s3 APIs to read and write bookmark details using s3. For now we will maintain the last copied file as a bookmark.

* Writing content to s3. We will use **s3://itv-github/sandbox/bookmark**.

1. bookmark\_contents = '2021-01-30-0.json.gz'
3. s3\_client.put\_object(
4. Bucket='itv-github',
5. Key='sandbox/bookmark',
6. Body=bookmark\_contents.encode('utf-8')
7. )

* Reading content from s3. If the bookmark is not there we need to catch the exception and use the baseline date as a bookmark.

1. try:
2. bookmark\_file = s3\_client.get\_object(
3. Bucket='itv-github',
4. Key='sandbox/bookmark'
5. )
6. prev\_file = bookmark\_file['Body'].read().decode('utf-8')
7. except ClientError as e:
8. if e.response['Error']['Code'] == 'NoSuchKey':
9. # Catch exception
10. # prev\_file = baseline\_file
11. else:
12. raise

[Instructions and Code] Maintaining Bookmark using s3

We need to maintain the bookmark for the files copied so that when we schedule the job to incrementally pull the next one, in case if the file is available.

1. from datetime import datetime as dt
2. from datetime import timedelta as td
3. import requests, boto3, os
4. from botocore.errorfactory import ClientError

7. # Make sure to change the date to one or two day older than based on the date you are running.
8. baseline\_file = '2021-01-30-0.json.gz'
10. os.environ.setdefault('AWS\_PROFILE', 'itvgithub')
11. s3\_client = boto3.client('s3')
13. while True:
14. try:
15. bookmark\_file = s3\_client.get\_object(
16. Bucket='itv-github',
17. Key='sandbox/bookmark'
18. )
19. prev\_file = bookmark\_file['Body'].read().decode('utf-8')
20. except ClientError as e:
21. if e.response['Error']['Code'] == 'NoSuchKey':
22. prev\_file = baseline\_file
23. else:
24. raise
26. dt\_part = prev\_file.split('.')[0]
27. next\_file = f"{dt.strftime(dt.strptime(dt\_part, '%Y-%M-%d-%H') + td(hours=1), '%Y-%M-%d-%-H')}.json.gz"
28. res = requests.get(f'https://data.gharchive.org/{next\_file}')
29. print(f'https://data.gharchive.org/{next\_file}')
30. if res.status\_code != 200:
31. break
32. print(f'The status code for {next\_file} is {res.status\_code}')
33. bookmark\_contents = next\_file
34. s3\_client.put\_object(
35. Bucket='itv-github',
36. Key='sandbox/bookmark',
37. Body=bookmark\_contents.encode('utf-8')
38. )

[Instructions and Source Code] - ghactivity-downloader Lambda Function

Here are the instructions you need to follow to go through the source code.

1. Create a folder by name **ghactivity-downloader** - mkdir ghactivity-downloader
2. Go to the folder **ghactivity-downloader** - cd ghactivity-downloader
3. Download the zip file from the downloadable resources and move the zip file to **ghactivity-downloader**
4. Unzip the zip file **ghactivity-downloader-src.zip** - On Mac or Linux you can use the command unzip ghactivity-downloader-src.zip
5. Create virtual environment using Python 3 - python3 -m venv ghad-venv
6. Activate virtual environment - source ghad-venv/bin/activate
7. Install required dependencies in the virtual environment (boto3, requests)
8. pip install requests
9. pip install boto3

8. Create a folder by name **ghalib** using mkdir ghalib. This will be used for dependencies that are supposed to be shipped along with the source code for our lambda function.

9. Install dependencies that need to be shipped to deploy as a lambda function

pip install requests -t ghalib

10. Build the zip file with the source code and dependencies such as requests using the following commands.

1. cd ghalib
2. zip -r ghactivity-downloader.zip .
3. cd ..
4. zip -r ghactivity-downloader.zip \*.py

11. Now zip file is ready for deployment as a lambda function.

[Instructions] Schedule Lambda Function using AWS Event Bridge

We can schedule the job using EventBridge so that files are downloaded and uploaded to s3 as soon as files are available in gharchive.

* Go to AWS Console and search for EventBride.
* Create rules and schedule every 15 minutes.
* Clean up the bookmark and the files added beyond baseline
* Validate whether the files are downloaded or not.

After the validation is done make sure the event bridge rule is disabled and deleted so that you do not end up paying unnecessarily.