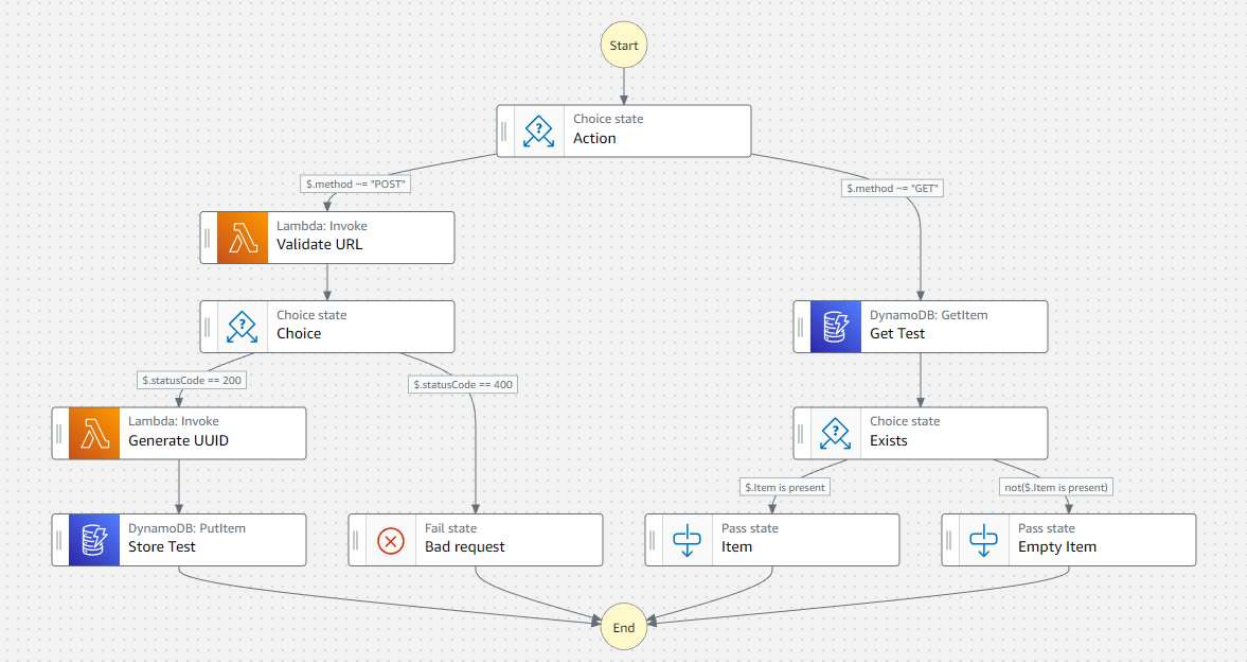
Christos Papastamos csd4569
HY452 Services Technologies
Assignment 1

---

1. Task A

1.1 Stracture

For the first task I created a State Machine based on the description given in the assignment which can be seen below:



When a request arrives to the Step Function the choice state checks its method.

If the request is a **POST** request, the choice state forwards it to the Validate URL lambda function which checks all of the request's fields. If the fields are valid, the next choice state forwards it to the Generate UUID lambda function where a random number is generated for each new test entry. Finally, DynamoDB stores the test to the table and returns SUCCESS.

If the request is a **GET** request, the choice state forwards the request to DynamoDB to get the test with the requested ID. If the test exists an item containing all the parameters of the test is passed to the State Functions output. If the test does not exist an empty item is generated and passed to the output.

1.2 Code

The code of the **Validate URL** lambda function can be seen below:

```
import json

accepted_methods=[ #a list of accepted methods
    "GET",
    "HEAD",
    "POST",
    "PUT",
    "DELETE",
    "CONNECT",
    "OPTIONS",
    "TRACE",
    "PATCH"
    ]

def error_handler(): #a basic error handler
    return {
        "statusCode" : 400
    }

def lambda_handler(event, context):

    if "url" in event:#check if the url is in the event
        url = event["url"]
    else:
        return error_handler()

    if "expectedResult" in event: #check if the expected result is in the event
        expectedResult = event["expectedResult"]
    else:
        return error_handler()

    if "testMethod" in event and (event["testMethod"] in accepted_methods): #check if the test
method is in the event and if it is an accepted method
        testMethod = event["testMethod"]
    else:
        return error_handler()

    return { #return a success result
        "statusCode" : 200,
        "item":{
            "url" : url ,
            "expectedResult" : expectedResult ,
            "testMethod" : testMethod
        }
    }
```

The code for the **Generate UUID** lambda function can be seen below:

```
import json
import random

def lambda_handler(event, context):
    item = event["item"]

    #check the request's fields
    if not ("url" in item):
        return {
            "statusCode": 400,
            "message": "url not included in json input"
        }
    elif not ("testMethod" in item):
        return {
            "statusCode": 400,
            "message": "testMethod not included in json input"
        }
    elif not ("expectedResult" in item):
        return {
            "statusCode": 400,
            "message": "expectedResult not included in json input"
        }

    return {
        "testID":           str(random.randint(0, 99999)), #generate a random
testID
        "testMethod":       item["testMethod"],
        "url":              item["url"],
        "expectedResult":   item["expectedResult"]
    }
```

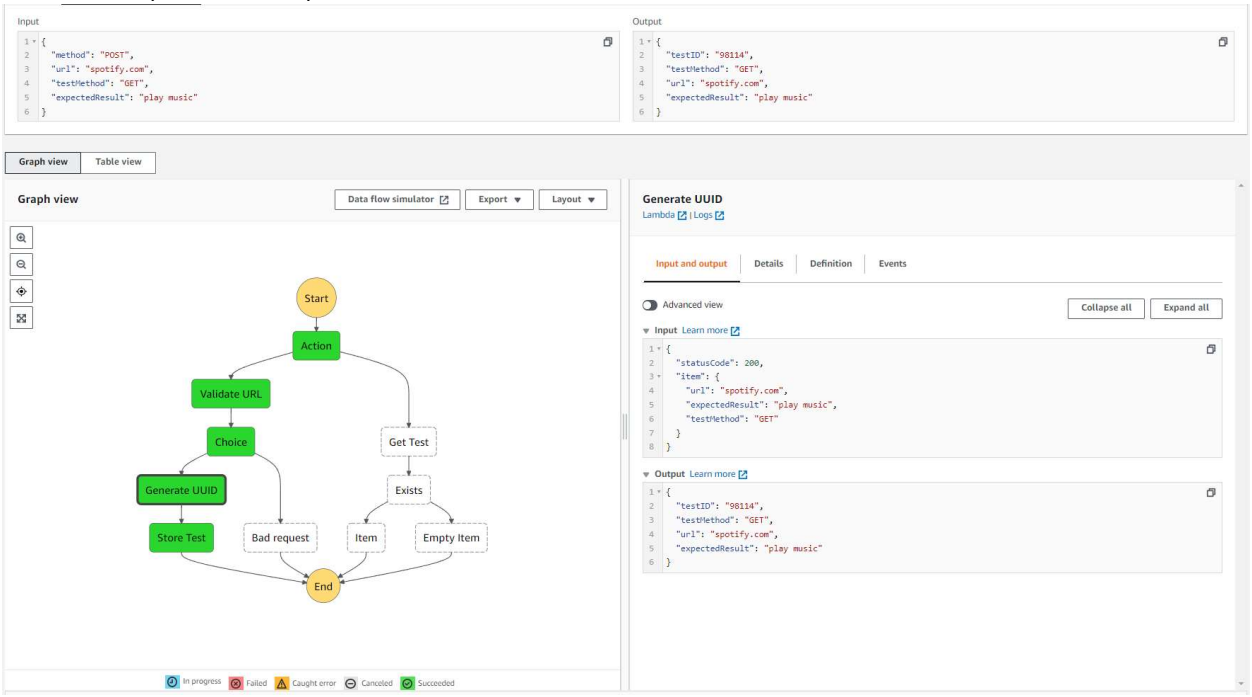
1.3 Execution and Testing

The expected input for this state machine can follow one of the following formats:

Add a new test	Get a test from a testID

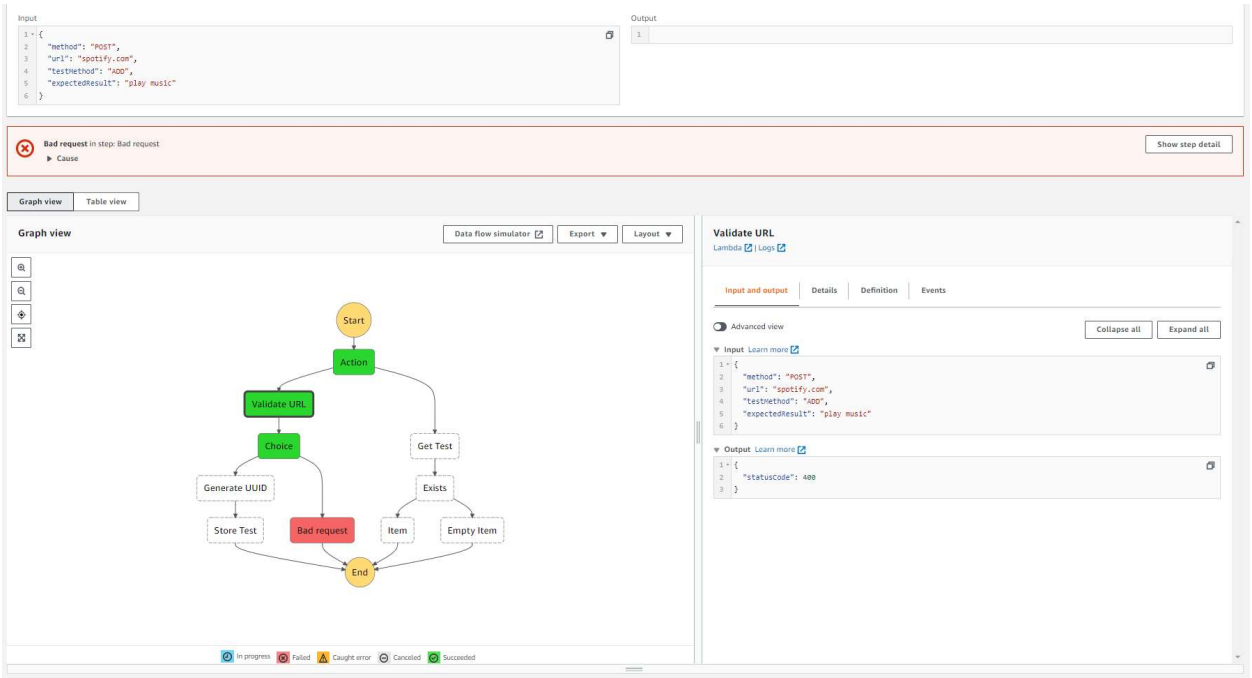
{
"method": “POST”,
"url": <String>,
"testMethod": <HTTP method>,
"expectedResult": <String>
}
	{
	"method" : "GET",
	"testID" : <Int>
	}

- For the **POST** part of the Step Function a <u>successful</u> test run can be seen below:



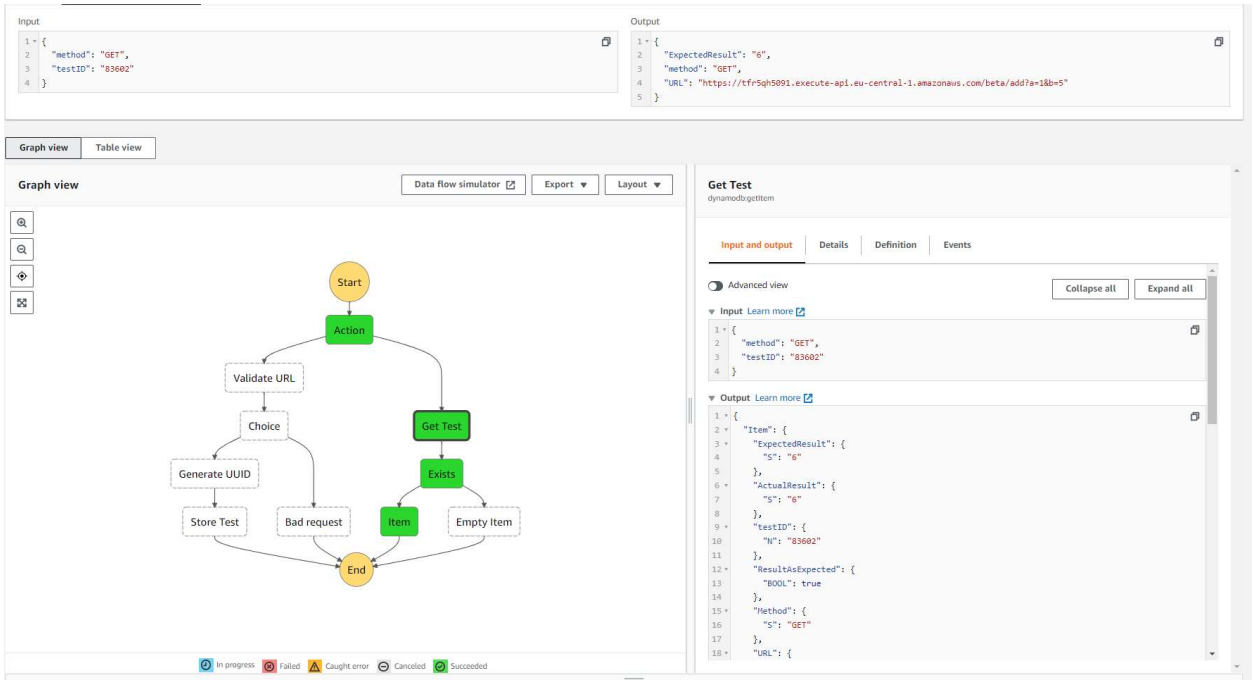
On the right of the Step Function we can see the input and output of the Generate UUID lambda function. This functions creates a random testID for each new entry.

- For the **POST** part of the Step Function an <u>unsuccessful</u> test run can be seen below:



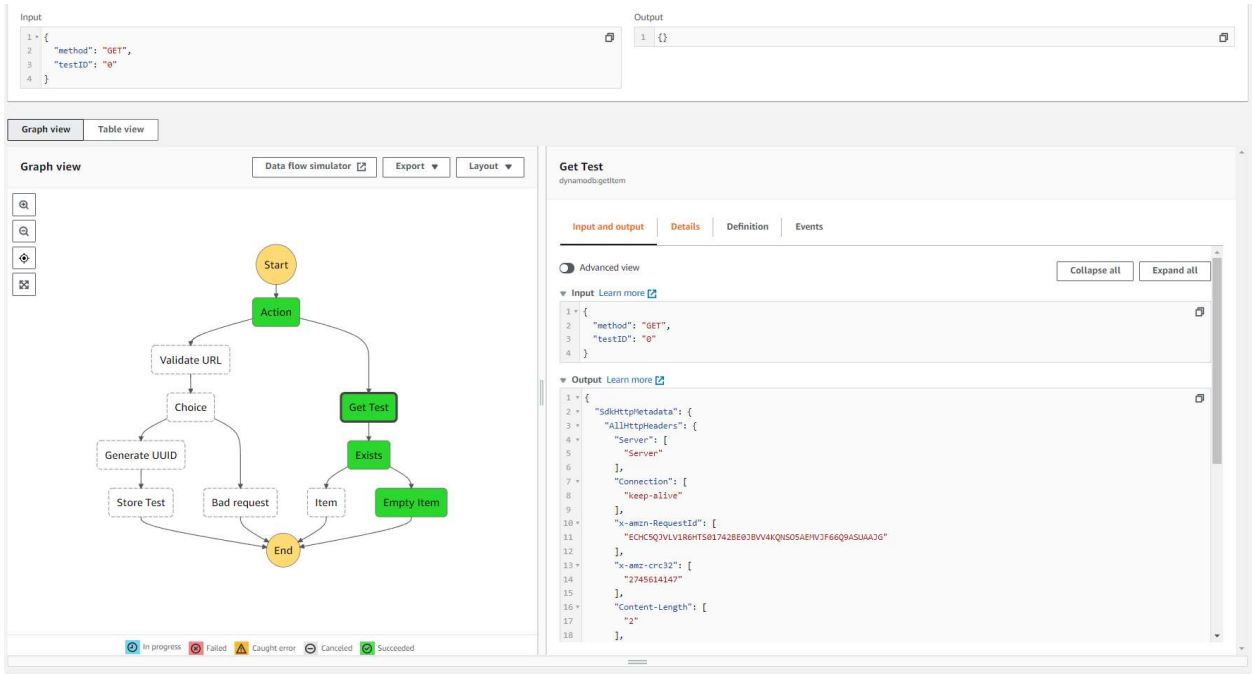
We can see that the testMethod field has the value “ADD” which is an invalid HTTP method, so the validate URL lambda function returned a status code of 400 which results in a State Machine failiure

- For the **GET** part of the Step Function a <u>successful</u> test run can be seen below (assuming ID 83602 is in the table):



Here we can see that the Step Function outputted the requested fields of the test successfully.

- For the **GET** part of the Step Function an <u>unsuccessful</u> test run can be seen below (ID 0 is not in the table):

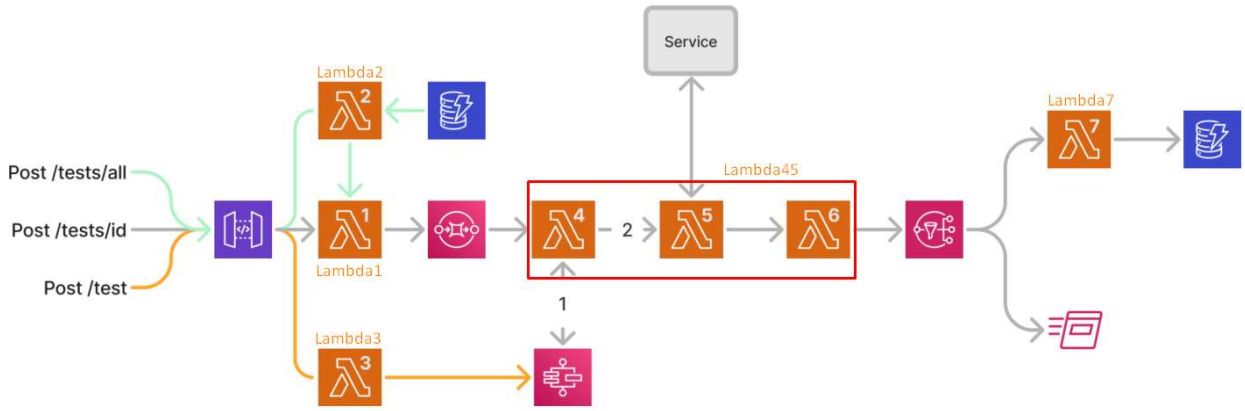


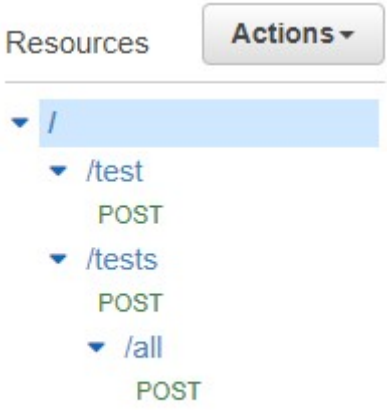
In this case the output of the Get Test DynamoDB Getitem returned a json which didn't contain the "item" field in it. That means that the test with the given ID is not in the table so an empty item is created for the output of the Step Function

2. Task B

2.1 Stracture

For the second task of the assignment I implemented the desired tool by making one change in the stracture.I merged the functionality of the lambda4-6 fucnctions to a single lambda function called Lambda45





Lets take if from the start. For the API gateway I created the resources and methods as seen on the screenshot on the right.

- /test?URL=<u>&Method=<m>&ExpectedResult=<e> (POST) : post a new service to test to the tool
- /tests?id=<ID> (POST) : test a test by ID
- /tests/all (POST) : Test all IDs currently in the table

In order to get the querry part parameters I used a mapping template in the Intergration Request ass seen below

/test?URL=<u>&Method=<m>&ExpectedResult=<e>	/tests?id=<ID>
{ "URL": "$input.params('URL')", "Method": "$input.params('Method')", "ExpectedResult": "$input.params('ExpectedResult')" }	{ "id": "$input.params('id')" }

Each resource is connected to their respective lambda function. Lambda functions will be listed in the code section.

For the Simple Queue Service I created a simple queue and added lambda45 as a trigger.

On the proccess of making lambda45 send a request and wait for a response from the State Machines I realized that Standard State Machines cannot run synchronously and for that I had to create a new State Machine of type Express with the exact same json code snippet. After that I was able to call the State Machine from lambda45 synchronously using the boto3 library.

For the communication of lamda2 and 7 with the DynamoDB I used the boto3 library as recommended.

Finally for the Simple Notification Service I created a standard topic (since I didn't want to add another SQS to the mix) and added a subscription to lambda7 and one to my academic email.

2.1 Code

A common found function in the code is print_debug which Is a extremely simple debug function I implemented.

- Lamda1

```
import json
import boto3

sqs = boto3.resource('sqs')
queue = sqs.get_queue_by_name(QueueName="Ask1_2_Queue1")

debug = 1

def print_debug(message): #simple debug function
    if debug:
        print("DEBUG: "+message)

def lambda_handler(event, context):
    print_debug("Recieved an event, begining execution ("+str(event)+")")

    if type(event) is dict:
        if "id" in event: #if the event is a dict and has an id, it came from the API
            print_debug("Request came for single id: ("+str(event["id"])+")")
            response = queue.send_message(MessageBody=event["id"]) #send the id to
the queue
            print_debug("Sending id "+str(event["id"])+" to SQS")
        else:
            print("ERROR: id not included in the input json object")
            return

    elif type(event) is list: #if the event is a list, it came from lambda2
        print_debug("Request came for id list: ("+str(event)+")")
        for id in event:
            if type(id) is int:#send each id to the queue
                response = queue.send_message(MessageBody=str(id))
                print_debug("Sending id "+str(id)+" to SQS")

            else:
                print("ERROR: non-inetger id detected in list ("+id+")")

    else :
        print("ERROR: input is not type list or json ("+event+")")
        return
```

- Lambda2

```
import json
import boto3

debug = 1

#resource for DynamoDB and lambda
dynamodb = boto3.resource('dynamodb')
client = boto3.client('lambda')

def print_debug(message): #simple debug function
    if debug:
        print("DEBUG: "+message)

def lambda_handler(event, context):
    print_debug("Recieved event, begining execution ("+str(event)+")")
    table = dynamodb.Table('Ask1Tests')

    response = table.scan() #get all items from table
    items = response['Items']
    ids = []

    for item in items:
        ids.append(int(item["testID"])) #get all testIDs

    print_debug("Recieved ids from table: "+str(ids))
    print_debug("Sending ids to Lambda1")
    response = client.invoke( #send ids to Lambda1
        FunctionName='Assignment1_Ask2_Lamda1',
        InvocationType='Event',
        LogType='None',
        Payload=str(ids),
    )

    print_debug("Payload sent, finishing execution")
```

- Lambda3 (Some lines are cliped so copy and paste the code to an editor e.g. VS Code)

```
import json
import boto3

debug = 1

sf_client = boto3.client('stepfunctions')
lambda_client = boto3.client('lambda')

def print_debug(message): #simple debug function
    if debug:
        print("DEBUG: "+message)

def error(message): #simple error function
    print("ERROR: "+message)

def lambda_handler(event, context):
    print_debug("Recieved event, begining execution ("+str(event)+")")

    if not("URL" in event): #check if required parameters are in event
        error("URL not specified in the request")
        return
    elif not("Method" in event):
        error("Method not specified in the request")
        return
    elif not("ExpectedResult" in event):
        error("ExpectedResult not specified in the request")
        return

    #Send the data to the SM
    print_debug("Sending data to State Machine")
    response = sf_client.start_sync_execution(
        stateMachineArn="arn:aws:states:us-east-
1:613787790506:stateMachine:ASK1_2_Express",
        input='{"method" : "POST","url":"'+event["URL"]+'","testMethod":
"'+event["Method"]+'","expectedResult": "'+event["ExpectedResult"]+'"}'
    )
    print_debug("State Machine returned status:"+response["status"]+"
("+str(response)+")")

    #Get the SM answer
    if response["status"]=="FAILED":
        return {"statusCode": 400, "Description": "Bad Request"}

    #craft the state machine output
    sm_output = '{"testID" : '+str(json.loads(response["output"])["testID"])+'}'
    print_debug("State Machine crafted output: "+sm_output)

    #Send the state machine output to the lambda45
    print_debug("Sending "+str(sm_output)+" to lambda45")
    response = lambda_client.invoke(
        FunctionName='Assignment1_Ask2_Lamda45',
        InvocationType='Event',
        LogType='None',
        Payload=sm_output
    )
```

- Lambda45

```
import json
import boto3
import urllib.request

debug = 1

sf_client = boto3.client('stepfunctions')
sns_client = boto3.client('sns')

def print_debug(message): #Simple debug function
    if debug:
        print("DEBUG: "+message)

def format_result(result): #Simple function to format the result
    print_debug("Formater: Result contains: "+str(result))
    print_debug("Formater: Result has type: "+str(type(result))+" .Formatting...")
    output = ""
    if type(result) is str:
        output = result.replace('"','\'')
    elif type(result) is bytes:
        output = str(result, "utf-8").replace('"','\'')

    print_debug("Formater: Result formating ended. final output string: '"+output+"'")
    return output

def lambda_handler(event, context):
    print_debug("Recieved event, begining execution ("+str(event)+")")

    #Simple error case
    if "testID" in event:
        test_id = event["testID"]
        print_debug("Call came from Lambda3 for id="+str(event["testID"]))
    elif not ("Records" in event) or not ("body" in event["Records"][0]):
        print("ERROR")
        return
    else:
        test_id = event["Records"][0]["body"]
        print_debug("Call came from SQS for id="+str(test_id))

    #send the ID to the state machine to get furhter details
    print_debug("Sending ID to State Machine")
    response = sf_client.start_sync_execution(
        stateMachineArn="arn:aws:states:us-east-1:613787790506:stateMachine:ASK1_2_Express",
        input='{"method" : "GET","testID" : "'+str(test_id)+'" }'
    )
    print_debug("State Machine returned ("+str(response)+")")

    #Get just the SM output
    sm_output = json.loads(response["output"])
    sm_output["testID"] = int(test_id)
    print_debug("State Machine crafted output: "+str(sm_output))

    #Validate the URL and also get result
    valid_URL = 1
    test_output = sm_output

    print_debug("Validating the URL")
    req = urllib.request.Request(url=sm_output["URL"], method=sm_output["Method"] ) #Create
the request
    try: response = urllib.request.urlopen(req) #Send the request
```

```
    except (urllib.error.URLError, ValueError) as e:
        # print_debug(e)
        valid_URL=0
        print_debug("URL invalid, storing error output")

    if valid_URL:
        print_debug("URL validated, storing the output")
        test_output["ActualResult"] = format_result(response.read())
    else:
        test_output["ActualResult"] = "Invalid URL"

    print_debug("URL output stored ("+str(test_output["ActualResult"])+")")

    #Compare Expected and Actual result
    if test_output["ActualResult"] == test_output["ExpectedResult"]:
        test_output["ResultAsExpected"] = True
    else:
        test_output["ResultAsExpected"] = False

    # Send the json to the SNS
    print_debug("Publishing into SNS (Message: "+str(test_output)+")")
    response = sns_client.publish(
        TopicArn='arn:aws:sns:us-east-1:613787790506:Ask1_2SNS',
        Message=str(test_output),
    )

    print_debug("End of execution, returning ("+str(response)+")")
    return response
```

The Lambda45 function needed a change in the configuration about its trigger from the SQS. The default batch size was 10 and as a result some triggers were lost. I changed the size to 1 and they seemed to work flawlessly together.

- Lambda7 (also clipped)

```
import json
import boto3
import ast

debut = 1

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Ask1Tests')

def debug(message): #simple debug function
    if debug:
        print("DEBUG: "+message)

def error(message):#simple error function
    print("ERROR: "+message)

def lambda_handler(event, context):

    if not("Records" in event) or not ("Sns" in event["Records"][0]) or not
("Message" in event["Records"][0]["Sns"]):
        error('Input has wrong format (["Records"][0]["Sns"]["Message"] field not
found)')
        return

    #format the test fields to store them into DynamoDB
    debug("event index = "+event["Records"][0]["Sns"]["Message"])
    eventstr = event["Records"][0]["Sns"]["Message"]
    debug("eventstr = "+eventstr)
    item = ast.literal_eval(eventstr) #i used ast.literal_eval to convert the string
to a dictionary because json.loads wants double quotes
    item["testID"] = int(item["testID"])
    debug("item = "+str(item))

    response = table.put_item(
        Item= item
    )
```

2.3 Execution and Testing

Let's start with an empty database and test the three API inputs and see some of the logs:

- /test/URL="https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5"&Method=GET&ExpectedResult=6

Timestamp	Message
	No older events at this moment. Retry
2022-10-22T01:21:16.294+03:00	START RequestId: 118488f7-ec00-40a8-9782-5aad0c900db8 Version: $LATEST
2022-10-22T01:21:16.295+03:00	DEBUG: Recieved event, begining execution ({'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'Method': 'GET', 'ExpectedResult': '6'})
2022-10-22T01:21:16.295+03:00	DEBUG: Sending data to State Machine
2022-10-22T01:21:17.197+03:00	DEBUG: State Machine returned status:SUCCEEDED ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:6e375900-9607-4eda-bbe9-c06ded3771d7:3c70d826-16e2-44b8-…
2022-10-22T01:21:17.197+03:00	DEBUG: State Machine crafted output: {"testID" : 95677}
2022-10-22T01:21:17.197+03:00	DEBUG: Sending {"testID" : 95677} to lambda45
2022-10-22T01:21:17.404+03:00	END RequestId: 118488f7-ec00-40a8-9782-5aad0c900db8
2022-10-22T01:21:17.404+03:00	REPORT RequestId: 118488f7-ec00-40a8-9782-5aad0c900db8 Duration: 1110.04 ms Billed Duration: 1111 ms Memory Size: 128 MB Max Memory Used: 69 MB Init Duration: 369.17 ms
	No newer events at this moment. Auto retry paused. Resume

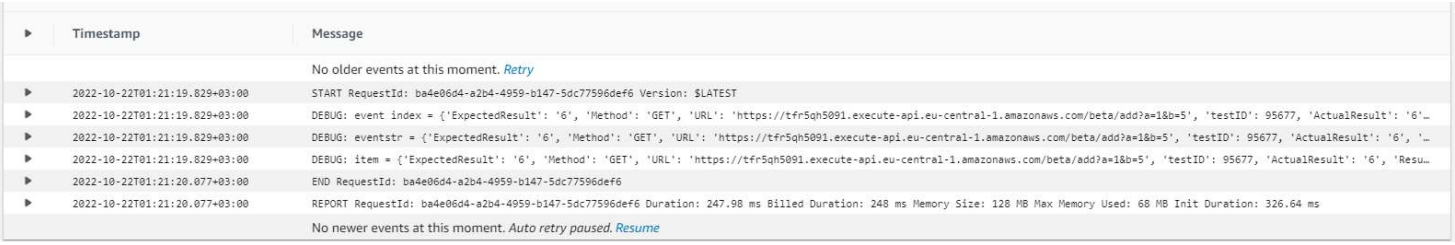
Lambda3 Function Logs

Above we can see through the DEBUG: prints that Lambda3 sent the test parameters to the Step Function and got an ID as a reply. Finally it sent the new test ID to Lambda45.

Timestamp	Message
	No older events at this moment. Retry
2022-10-22T01:21:17.922+03:00	START RequestId: a73fa5ec-22bc-4bc0-90bd-96240d3c6853 Version: $LATEST
2022-10-22T01:21:17.922+03:00	DEBUG: Recieved event, begining execution ({'testID': 95677})
2022-10-22T01:21:17.922+03:00	DEBUG: Call came from Lambda3 for id=95677
2022-10-22T01:21:17.922+03:00	DEBUG: Sending ID to State Machine
2022-10-22T01:21:18.217+03:00	DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:19127c70-035c-42ee-86f8-034fce5323f3:d8841a4d-0462-4952-931c-64709c5188e4…
2022-10-22T01:21:18.217+03:00	DEBUG: State Machine crafted output: {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677}
2022-10-22T01:21:18.217+03:00	DEBUG: Validating the URL
2022-10-22T01:21:19.047+03:00	DEBUG: URL validated, storing the output
2022-10-22T01:21:19.048+03:00	DEBUG: Formater: Result contains: b'6'
2022-10-22T01:21:19.048+03:00	DEBUG: Formater: Result has type: <class 'bytes'> .Formatting...
2022-10-22T01:21:19.048+03:00	DEBUG: Formater: Result formating ended. final output string: '6'
2022-10-22T01:21:19.048+03:00	DEBUG: URL output stored (6)
2022-10-22T01:21:19.048+03:00	DEBUG: Publishing into SNS (Message: {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'Ac…
2022-10-22T01:21:19.244+03:00	DEBUG: End of execution, returning ({'MessageId': '1c7060b4-17d9-5060-b14d-3b790a46d4eb', 'ResponseMetadata': {'RequestId': '075cb938-5491-50a9-a5c5-e2f5490ea422', 'HTTPStatusCode': 2…
2022-10-22T01:21:19.246+03:00	END RequestId: a73fa5ec-22bc-4bc0-90bd-96240d3c6853
2022-10-22T01:21:19.246+03:00	REPORT RequestId: a73fa5ec-22bc-4bc0-90bd-96240d3c6853 Duration: 1324.59 ms Billed Duration: 1325 ms Memory Size: 128 MB Max Memory Used: 69 MB Init Duration: 329.75 ms
	No newer events at this moment. Auto retry paused. Resume

Lambda45 Function Logs

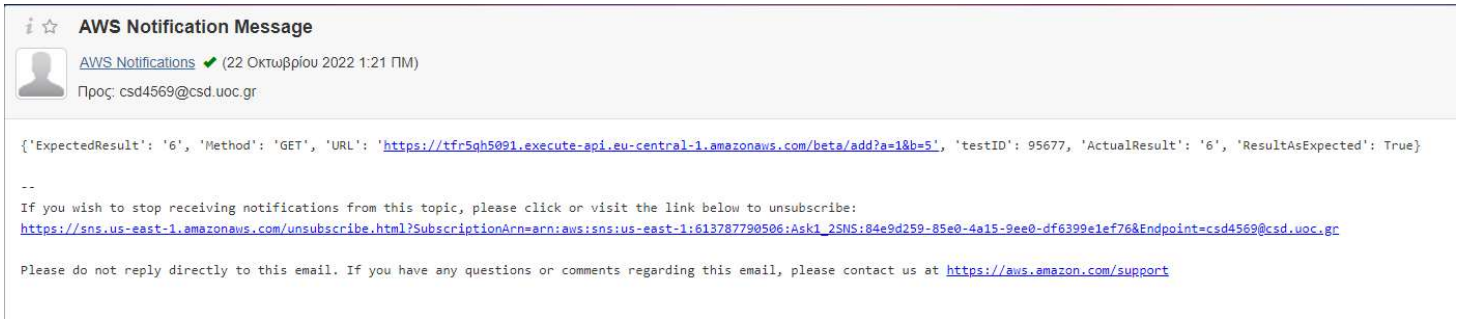
In Lambda 45 function logs we can observe the function getting the test parameters from the Step Function using the test ID. Then it validates the website and gets the response from it. Finally it compares the actual and expected output and sends all the gathered data to the SNS for storing and emailing.



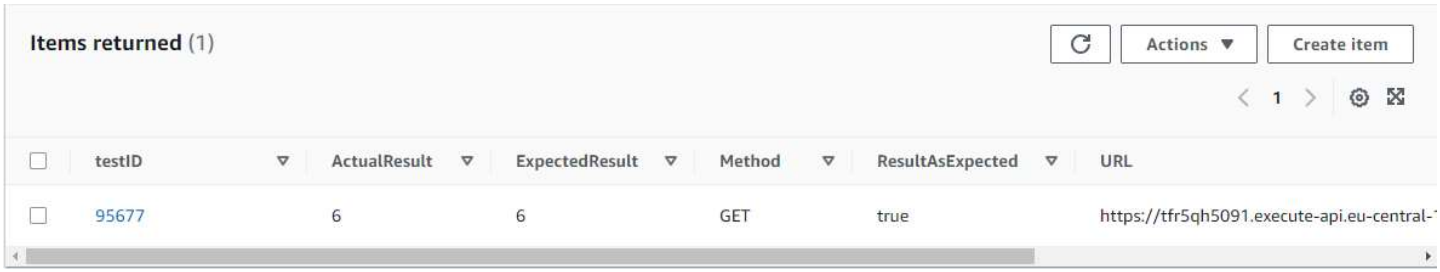
Timestamp	Message
	No older events at this moment. Retry
2022-10-22T01:21:19.829+03:00	START RequestId: ba4e06d4-a2b4-4959-b147-5dc77596def6 Version: $LATEST
2022-10-22T01:21:19.829+03:00	DEBUG: event index = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6'…
2022-10-22T01:21:19.829+03:00	DEBUG: eventstr = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6', '…
2022-10-22T01:21:19.829+03:00	DEBUG: item = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6', 'Resu…
2022-10-22T01:21:20.077+03:00	END RequestId: ba4e06d4-a2b4-4959-b147-5dc77596def6
2022-10-22T01:21:20.077+03:00	REPORT RequestId: ba4e06d4-a2b4-4959-b147-5dc77596def6 Duration: 247.98 ms Billed Duration: 248 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 326.64 ms
	No newer events at this moment. Auto retry paused. Resume

Lambda7 Function Logs

In the screenshot above Lambda7 sends the results to DunamoDB (after some formatting).

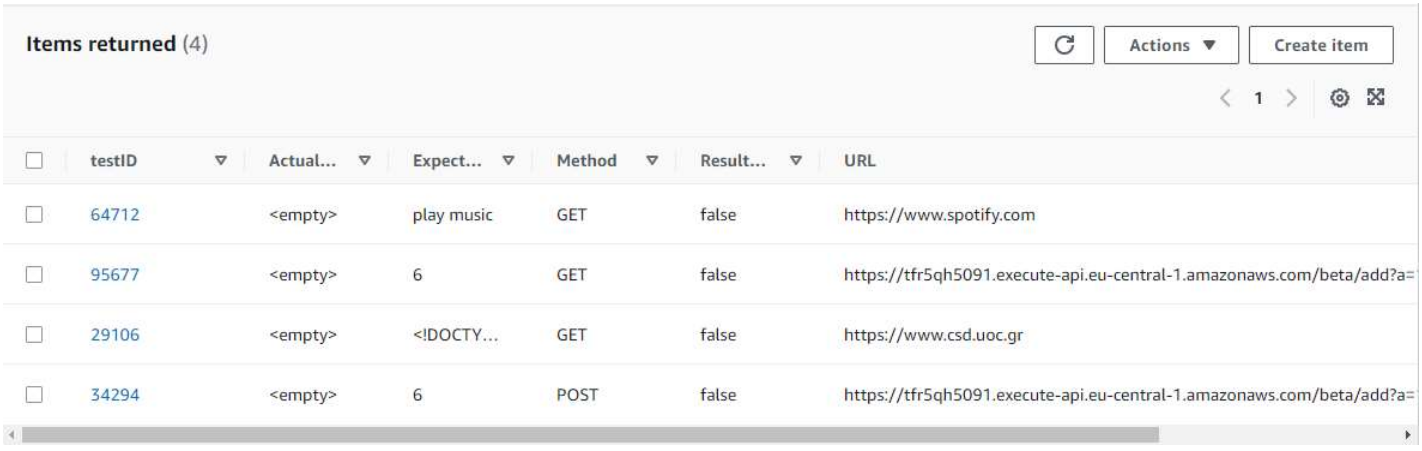


Email Received



Database updated

For the next tests we need some additions to the database (I removed all the actual results)



- /tests/64712

Timestamp	Message
	No older events at this moment. *Retry*
2022-10-22T01:48:45.638+03:00	START RequestId: 0c269cfc-e7e7-431e-b157-77ac186e850c Version: $LATEST
2022-10-22T01:48:45.638+03:00	DEBUG: Recieved an event, begining execution ({'id': '64712'})
2022-10-22T01:48:45.638+03:00	DEBUG: Request came for single id: (64712)
2022-10-22T01:48:45.678+03:00	DEBUG: Sending id 64712 to SQS
2022-10-22T01:48:45.695+03:00	END RequestId: 0c269cfc-e7e7-431e-b157-77ac186e850c
2022-10-22T01:48:45.695+03:00	REPORT RequestId: 0c269cfc-e7e7-431e-b157-77ac186e850c Duration: 57.51 ms Billed Duration: 58 ms Memory Size: 128 MB Max Memory Used: 67 MB Init Duration: 558.88 ms
	No newer events at this moment. *Auto retry paused. Resume*

Lambda1 Function Logs

When the API resource is called it calls Lambda1 which takes the test ID and just passes it to SQS

2022-10-22T01:48:45.689+03:00	START RequestId: 2dac9c82-b662-5d1c-b47b-c74f9265b0a5 Version: $LATEST
2022-10-22T01:48:45.689+03:00	DEBUG: Recieved event, begining execution ({'Records': [{'messageId': '7523390f-26d1-48c5-8c7f-82ebb2e0dfee', 'receiptHandle': 'AQEBKhc3imwbabOkn3UHdioQcgyJGP1K4pe1b0SU9B3KcolWHrzxbYH…
2022-10-22T01:48:45.689+03:00	DEBUG: Call came from SQS for id=64712
2022-10-22T01:48:45.689+03:00	DEBUG: Sending ID to State Machine
2022-10-22T01:48:45.759+03:00	DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:6e247e5c-2ddb-4968-8f04-5075ec2e3893:80ae0fc9-3d74-48ed-b342-a20ab6e62c96…
2022-10-22T01:48:45.759+03:00	DEBUG: State Machine crafted output: {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712}
2022-10-22T01:48:45.759+03:00	DEBUG: Validating the URL
2022-10-22T01:48:46.259+03:00	DEBUG: URL validated, storing the output
2022-10-22T01:48:46.280+03:00	DEBUG: Formater: Result contains: b'<!DOCTYPE html><html lang="en" dir="ltr"><head nonce="533e19299b92442d9f619cb9a215ae53"><link rel="preload" href="https://encore.scdn.co/1.2.3/Circ…
2022-10-22T01:48:46.280+03:00	DEBUG: Formater: Result has type: <class 'bytes'> .Formatting...
2022-10-22T01:48:46.317+03:00	DEBUG: Formater: Result formating ended. final output string: '<!DOCTYPE html><html lang='en' dir='ltr'><head nonce='533e19299b92442d9f619cb9a215ae53'><link rel='preload' href='https:…
2022-10-22T01:48:46.317+03:00	@media (min-width:768px){.kKvTd{font-size:64px;line-height:72px;-webkit-letter-spacing:-2px;-moz-letter-spacing:-2px;-ms-letter-spacing:-2px;letter-spacing:-2px;}}/*!sc*/
2022-10-22T01:48:46.317+03:00	@media (min-width:992px){.kKvTd{font-size:80px;line-height:88px;}}/*!sc*/
2022-10-22T01:48:46.317+03:00	
2022-10-22T01:48:46.317+03:00	
2022-10-22T01:48:46.317+03:00	© 2022 Spotify AB
2022-10-22T01:48:46.317+03:00	</nav>
2022-10-22T01:48:46.317+03:00	</footer> <script id='__NEXT_DATA__' type='application/json' nonce='533e19299b92442d9f619cb9a215ae53'>{'props':{'pageProps':{'experimentVariants':{'audiobookEnabled':false}…
2022-10-22T01:48:46.321+03:00	DEBUG: Publishing into SNS (Message: {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='e…
2022-10-22T01:48:46.732+03:00	DEBUG: End of execution, returning ({'MessageId': 'c0fe4e66-c10f-5d98-bd6a-15abe50150d5', 'ResponseMetadata': {'RequestId': '021ad116-78f2-5371-bb6b-8adb5cfc8e3c', 'HTTPStatusCode': 2…
2022-10-22T01:48:46.734+03:00	END RequestId: 2dac9c82-b662-5d1c-b47b-c74f9265b0a5
2022-10-22T01:48:46.734+03:00	REPORT RequestId: 2dac9c82-b662-5d1c-b47b-c74f9265b0a5 Duration: 1045.25 ms Billed Duration: 1046 ms Memory Size: 128 MB Max Memory Used: 30 MB
	No newer events at this moment. *Auto retry paused. Resume*

Back to top

Lambda45 Function Logs

The formater debug output has spammed the logs but with the power of editing I made it somewhat readable. Lambda45 as said in the previus run takes the tests parameters, validates the url, stores the result, compares it with the actual one and finally sends all the findings to SNS

2022-10-22T01:48:46.895+03:00	START RequestId: 7f83e9cd-4cdb-426f-a01c-63d215898a70 Version: $LATEST
2022-10-22T01:48:46.896+03:00	DEBUG: event index = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><he…
2022-10-22T01:48:46.896+03:00	DEBUG: eventstr = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><head …
2022-10-22T01:48:46.898+03:00	DEBUG: item = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><head nonc…
2022-10-22T01:48:47.111+03:00	END RequestId: 7f83e9cd-4cdb-426f-a01c-63d215898a70
2022-10-22T01:48:47.111+03:00	REPORT RequestId: 7f83e9cd-4cdb-426f-a01c-63d215898a70 Duration: 215.38 ms Billed Duration: 216 ms Memory Size: 128 MB Max Memory Used: 70 MB

Lambda7 Function Logs

On the Last mile again, Lambda7 sends the results to DynamoDB for storing

- /tests/all

Table Contents after the test run

testID	Actual...	Expect...	Method	Result...	URL
64712	<!DOCTY...	play music	GET	false	https://www.spotify.com
95677	<empty>	6	GET	false	https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=
29106	<empty>	<!DOCTY...	GET	false	https://www.csd.uoc.gr
34294	<empty>	6	POST	false	https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=

With the database as seen above we send a request to the resource

```
2022-10-22T02:21:33.237+03:00  START RequestId: 8edad369-ce26-489f-bcdf-ca6be3771a11 Version: $LATEST
2022-10-22T02:21:33.237+03:00  DEBUG: Recieved event, begining execution ({})
2022-10-22T02:21:33.520+03:00  DEBUG: Recieved ids from table: [64712, 95677, 29106, 34294]
2022-10-22T02:21:33.520+03:00  DEBUG: Sending ids to Lambda1
2022-10-22T02:21:33.758+03:00  DEBUG: Payload sent, finishing execution
2022-10-22T02:21:33.759+03:00  END RequestId: 8edad369-ce26-489f-bcdf-ca6be3771a11
2022-10-22T02:21:33.759+03:00  REPORT RequestId: 8edad369-ce26-489f-bcdf-ca6be3771a11 Duration: 522.30 ms Billed Duration: 523 ms Memory Size: 128 MB Max Memory Used: 71 MB Init Duration: 461.81 ms
```

Lambda2 Function Logs

Lambda2 gathers all test IDs from DynamoDB table and adds them to a list. Then forwards it to Lambda1

```
2022-10-22T02:21:34.369+03:00  START RequestId: 28284d47-ba56-4a7e-b3d4-084bcba78f42 Version: $LATEST
2022-10-22T02:21:34.369+03:00  DEBUG: Recieved an event, begining execution ([64712, 95677, 29106, 34294])
2022-10-22T02:21:34.369+03:00  DEBUG: Request came for id list: ([64712, 95677, 29106, 34294])
2022-10-22T02:21:34.405+03:00  DEBUG: Sending id 64712 to SQS
2022-10-22T02:21:34.425+03:00  DEBUG: Sending id 95677 to SQS
2022-10-22T02:21:34.465+03:00  DEBUG: Sending id 29106 to SQS
2022-10-22T02:21:34.485+03:00  DEBUG: Sending id 34294 to SQS
2022-10-22T02:21:34.504+03:00  END RequestId: 28284d47-ba56-4a7e-b3d4-084bcba78f42
2022-10-22T02:21:34.504+03:00  REPORT RequestId: 28284d47-ba56-4a7e-b3d4-084bcba78f42 Duration: 135.15 ms Billed Duration: 136 ms Memory Size: 128 MB Max Memory Used: 67 MB Init Duration: 442.93 ms
```

Lambda1 Function Logs

Lambda1 gets the ID list and sends every ID individually to SQS

```
2022-10-22T02:21:34.874+03:00  START RequestId: d9a5cd12-46a3-5282-94e2-96df6d132887 Version: $LATEST
2022-10-22T02:21:34.875+03:00  DEBUG: Recieved event, begining execution ({'Records': [{'messageId': 'eaf5bbed-ddc1-4aa1-873b-22a079f3f10e', 'receiptHandle': 'AQEBCI1wkDJce3BY0HJXvThzDRWsHkDHZJGFv6UVQyv70nFtsUyMGii…
2022-10-22T02:21:34.875+03:00  DEBUG: Call came from SQS for id=95677
2022-10-22T02:21:34.875+03:00  DEBUG: Sending ID to State Machine
2022-10-22T02:21:35.175+03:00  DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:09c22a0b-4d46-47a4-847b-23593628b95b:69adfd3b-7c25-407f-99cb-4ba907a7745d…
2022-10-22T02:21:35.175+03:00  DEBUG: State Machine crafted output: {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677}
2022-10-22T02:21:35.175+03:00  DEBUG: Validating the URL
2022-10-22T02:21:35.982+03:00  DEBUG: URL validated, storing the output
2022-10-22T02:21:35.983+03:00  DEBUG: Publishing into SNS (Message: {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'Ac…
2022-10-22T02:21:36.174+03:00  DEBUG: End of execution, returning ({'MessageId': '387bcb6a-09cb-5d09-b889-90a7220bfcd4', 'ResponseMetadata': {'RequestId': 'd90c8b26-38df-5888-be49-d0eb131fd955', 'HTTPStatusCode': 2…
2022-10-22T02:21:36.175+03:00  END RequestId: d9a5cd12-46a3-5282-94e2-96df6d132887
2022-10-22T02:21:36.175+03:00  REPORT RequestId: d9a5cd12-46a3-5282-94e2-96df6d132887 Duration: 1300.74 ms Billed Duration: 1301 ms Memory Size: 128 MB Max Memory Used: 69 MB Init Duration: 352.29 ms
2022-10-22T02:21:37.537+03:00  START RequestId: 6525d751-46c7-56ad-b808-8c9c6d066ca8 Version: $LATEST
2022-10-22T02:21:37.537+03:00  DEBUG: Recieved event, begining execution ({'Records': [{'messageId': 'f143416a-6627-4d06-8ea5-9570e2876429', 'receiptHandle': 'AQEBjFw6FTLGr+TGVsBe6F/AcZQ7ki+pTsvVeu9x/zmke0G51OSI4Ov…
2022-10-22T02:21:37.537+03:00  DEBUG: Call came from SQS for id=34294
2022-10-22T02:21:37.537+03:00  DEBUG: Sending ID to State Machine
2022-10-22T02:21:37.662+03:00  DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:a4a02219-c83c-402f-84ea-71648d1dc8d8:f0ac1ae5-e5dc-43df-8460-d624041dfa7b…
2022-10-22T02:21:37.662+03:00  DEBUG: State Machine crafted output: {'ExpectedResult': '6', 'Method': 'POST', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 34294}
2022-10-22T02:21:37.662+03:00  DEBUG: Validating the URL
2022-10-22T02:21:38.189+03:00  DEBUG: URL invalid, storing error output
2022-10-22T02:21:38.190+03:00  DEBUG: Publishing into SNS (Message: {'ExpectedResult': '6', 'Method': 'POST', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 34294, 'A…
2022-10-22T02:21:38.236+03:00  DEBUG: End of execution, returning ({'MessageId': 'ed59a831-b0dd-56d8-a23b-f3fc9b1f9705', 'ResponseMetadata': {'RequestId': '0b603624-7e6e-594d-96b5-6de955acc890', 'HTTPStatusCode': 2…
2022-10-22T02:21:38.237+03:00  END RequestId: 6525d751-46c7-56ad-b808-8c9c6d066ca8
2022-10-22T02:21:38.237+03:00  REPORT RequestId: 6525d751-46c7-56ad-b808-8c9c6d066ca8 Duration: 700.45 ms Billed Duration: 701 ms Memory Size: 128 MB Max Memory Used: 70 MB
```

Lambda45 Function Logs (1/2)

```
2022-10-22T02:21:34.836+03:00  START RequestId: bac54cd2-d7e4-5a1f-a5de-171ffc492103 Version: $LATEST
2022-10-22T02:21:34.836+03:00  DEBUG: Recieved event, begining execution ({'Records': [{'messageId': '05e56b2f-1b3d-46e5-b4ff-5548a7ebb88d', 'receiptHandle': 'AQEBGQsaRqn68pBS46sLqrKq+bSCKJjYPnbjoMqcjeWURix9jTeK886…
2022-10-22T02:21:34.836+03:00  DEBUG: Call came from SQS for id=64712
2022-10-22T02:21:34.836+03:00  DEBUG: Sending ID to State Machine
2022-10-22T02:21:35.174+03:00  DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:79c9fdd5-d82e-4c5f-bcbf-b059074a759e:b2c7411b-1531-4ad9-8533-828889123a04…
2022-10-22T02:21:35.174+03:00  DEBUG: State Machine crafted output: {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712}
2022-10-22T02:21:35.174+03:00  DEBUG: Validating the URL
2022-10-22T02:21:35.681+03:00  DEBUG: URL validated, storing the output
2022-10-22T02:21:35.695+03:00  DEBUG: Publishing into SNS (Message: {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='e…
2022-10-22T02:21:36.071+03:00  DEBUG: End of execution, returning ({'MessageId': 'ecd20f05-0962-5114-8e3c-54c822a27286', 'ResponseMetadata': {'RequestId': 'b84f288d-05c6-50b0-9c13-73402e064322', 'HTTPStatusCode': 2…
2022-10-22T02:21:36.074+03:00  END RequestId: bac54cd2-d7e4-5a1f-a5de-171ffc492103
2022-10-22T02:21:36.074+03:00  REPORT RequestId: bac54cd2-d7e4-5a1f-a5de-171ffc492103 Duration: 1238.13 ms Billed Duration: 1239 ms Memory Size: 128 MB Max Memory Used: 70 MB Init Duration: 339.45 ms
2022-10-22T02:21:37.591+03:00  START RequestId: 997701e2-0ece-555b-b046-5a2f250f8e47 Version: $LATEST
2022-10-22T02:21:37.592+03:00  DEBUG: Recieved event, begining execution ({'Records': [{'messageId': '1873b44d-f0cd-4b13-995c-131d001ee690', 'receiptHandle': 'AQEBsMnInJDXD1/C8DKryQamN+tx71f5dFgYG4Ca4W+nP0yyGiGc0UD…
2022-10-22T02:21:37.592+03:00  DEBUG: Call came from SQS for id=29106
2022-10-22T02:21:37.592+03:00  DEBUG: Sending ID to State Machine
2022-10-22T02:21:37.717+03:00  DEBUG: State Machine returned ({'executionArn': 'arn:aws:states:us-east-1:613787790506:express:ASK1_2_Express:66e562e7-2f9c-4d5d-9af7-bb4ff3ed80ae:68ce5eaa-b0de-4c72-972d-bf2672b89693…
2022-10-22T02:21:37.717+03:00  DEBUG: State Machine crafted output: {'ExpectedResult': "\n\n\n\n\n\n\n<!DOCTYPE html>\n<html>\n <head>\n <meta name='author' content='Designed and developed as a bachelor thesis from…
2022-10-22T02:21:37.717+03:00  DEBUG: Validating the URL
2022-10-22T02:21:38.737+03:00  DEBUG: URL validated, storing the output
2022-10-22T02:21:39.054+03:00  DEBUG: Publishing into SNS (Message: {'ExpectedResult': "\n\n\n\n\n\n\n<!DOCTYPE html>\n<html>\n <head>\n <meta name='author' content='Designed and developed as a bachelor thesis from…
2022-10-22T02:21:39.546+03:00  DEBUG: End of execution, returning ({'MessageId': '49f97dd3-1f25-5e84-b3fd-261e34e6dd16', 'ResponseMetadata': {'RequestId': '234ed8f6-9988-5a06-9174-833bb24ba994', 'HTTPStatusCode': 2…
2022-10-22T02:21:39.547+03:00  END RequestId: 997701e2-0ece-555b-b046-5a2f250f8e47
2022-10-22T02:21:39.547+03:00  REPORT RequestId: 997701e2-0ece-555b-b046-5a2f250f8e47 Duration: 1955.76 ms Billed Duration: 1956 ms Memory Size: 128 MB Max Memory Used: 72 MB
```

Lambda45 Function Logs (2/2)

On Lambda45 we can see that all 4 test IDs are processed and passed to SNS (I removed some debug prints because the spam became uncontrollable)

```
2022-10-22T02:21:36.246+03:00  START RequestId: 21daeb39-d923-4c60-abdf-5a77f7c27d51 Version: $LATEST
2022-10-22T02:21:36.248+03:00  DEBUG: event index = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><he…
2022-10-22T02:21:36.248+03:00  DEBUG: eventstr = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><head …
2022-10-22T02:21:36.249+03:00  DEBUG: item = {'ExpectedResult': 'play music', 'Method': 'GET', 'URL': 'https://www.spotify.com', 'testID': 64712, 'ActualResult': "<!DOCTYPE html><html lang='en' dir='ltr'><head nonc…
2022-10-22T02:21:36.477+03:00  END RequestId: 21daeb39-d923-4c60-abdf-5a77f7c27d51
2022-10-22T02:21:36.478+03:00  REPORT RequestId: 21daeb39-d923-4c60-abdf-5a77f7c27d51 Duration: 231.52 ms Billed Duration: 232 ms Memory Size: 128 MB Max Memory Used: 71 MB
```

Lambda7 Function Logs (1/4)

```
2022-10-22T02:21:36.844+03:00  START RequestId: 6f07ff45-6e72-4956-af9f-5f73dcfd06fa Version: $LATEST
2022-10-22T02:21:36.844+03:00  DEBUG: event index = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6'…
2022-10-22T02:21:36.844+03:00  DEBUG: eventstr = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6', '…
2022-10-22T02:21:36.844+03:00  DEBUG: item = {'ExpectedResult': '6', 'Method': 'GET', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 95677, 'ActualResult': '6', 'Resu…
2022-10-22T02:21:37.091+03:00  END RequestId: 6f07ff45-6e72-4956-af9f-5f73dcfd06fa
2022-10-22T02:21:37.091+03:00  REPORT RequestId: 6f07ff45-6e72-4956-af9f-5f73dcfd06fa Duration: 247.23 ms Billed Duration: 248 ms Memory Size: 128 MB Max Memory Used: 68 MB Init Duration: 357.69 ms
2022-10-22T02:21:38.354+03:00  START RequestId: 3bd756f3-5f78-47e1-846d-cb37835ffc1d Version: $LATEST
2022-10-22T02:21:38.354+03:00  DEBUG: event index = {'ExpectedResult': '6', 'Method': 'POST', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 34294, 'ActualResult': 'I…
2022-10-22T02:21:38.354+03:00  DEBUG: eventstr = {'ExpectedResult': '6', 'Method': 'POST', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 34294, 'ActualResult': 'Inva…
2022-10-22T02:21:38.354+03:00  DEBUG: item = {'ExpectedResult': '6', 'Method': 'POST', 'URL': 'https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=1&b=5', 'testID': 34294, 'ActualResult': 'Invalid …
2022-10-22T02:21:38.370+03:00  END RequestId: 3bd756f3-5f78-47e1-846d-cb37835ffc1d
2022-10-22T02:21:38.370+03:00  REPORT RequestId: 3bd756f3-5f78-47e1-846d-cb37835ffc1d Duration: 15.87 ms Billed Duration: 16 ms Memory Size: 128 MB Max Memory Used: 69 MB
2022-10-22T02:21:39.695+03:00  START RequestId: b6b3fdd5-3246-4859-91df-eec5f5877122 Version: $LATEST
2022-10-22T02:21:39.709+03:00  DEBUG: event index = {'ExpectedResult': "\n\n\n\n\n\n\n<!DOCTYPE html>\n<html>\n <head>\n <meta name='author' content='Designed and developed as a bachelor thesis from Siopis Ilias ( …
2022-10-22T02:21:39.709+03:00  DEBUG: eventstr = {'ExpectedResult': "\n\n\n\n\n\n\n<!DOCTYPE html>\n<html>\n <head>\n <meta name='author' content='Designed and developed as a bachelor thesis from Siopis Ilias ( ils…
2022-10-22T02:21:39.769+03:00  DEBUG: item = {'ExpectedResult': "\n\n\n\n\n\n\n<!DOCTYPE html>\n<html>\n <head>\n <meta name='author' content='Designed and developed as a bachelor thesis from Siopis Ilias ( ilsio9@…
2022-10-22T02:21:39.830+03:00  END RequestId: b6b3fdd5-3246-4859-91df-eec5f5877122
2022-10-22T02:21:39.830+03:00  REPORT RequestId: b6b3fdd5-3246-4859-91df-eec5f5877122 Duration: 135.44 ms Billed Duration: 136 ms Memory Size: 128 MB Max Memory Used: 71 MB
```

Lambda7 Function Logs (2-4/4)

Lambda7 takes all notifications and stores them to DynamoDB as can be seen below

testID	Actual...	Expect...	Method	Result...	URL
64712	<!DOCTY...	play music	GET	false	https://www.spotify.com
95677	6	6	GET	true	https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=
29106	<!DOCTY...	<!DOCTY...	GET	false	https://www.csd.uoc.gr
34294	Invalid URL	6	POST	false	https://tfr5qh5091.execute-api.eu-central-1.amazonaws.com/beta/add?a=