Module 11

Enhancing the ProcessS3Event Lambda Function Developed In The Previous Module

Overview

In the previous module (Module 10), you authored a ProcessS3Event lambda function and verified that it is getting triggered when a new file is added to an S3 bucket. The purpose of Module 10 was to learn how to set up that connectivity. In the code of the Lambda function, you were asked to process a few XML files and print some content to CloudWatch. In this module we look closely at that. We will also look at some Lambda properties that we haven't covered yet (e.g., environment variables).

In most cases when you configure lambda to listen to some S3 event (like the PUT object event), you want to process that object/file. And that usually means inspecting the file content and performing some action.

Objective of Module

The objective of this module is to practice reading the content of an S3 file from the Lambda code. We will also look at environment variables and how and when to use them.

Steps

1. Enclosed with this module is an XML file (results.xml). Drag it into a browser to inspect its content. Assume this is a medical test result of a patient. We want to write a Lambda that processes this data (and maybe add its details to a database – not in this module).

2. Open the Visual Studio solution you created in Module 10.

3. Open file Function.cs in project **ProcessS3Event**. You want to change the code of method `FunctionHandler` to parse the XML file (when S3 triggers your Lambda function that the file was added to S3).

4. We will use a Stream object to read the file content, and XML parsing to parse the content. Therefore, add the following at the top of Function.cs:

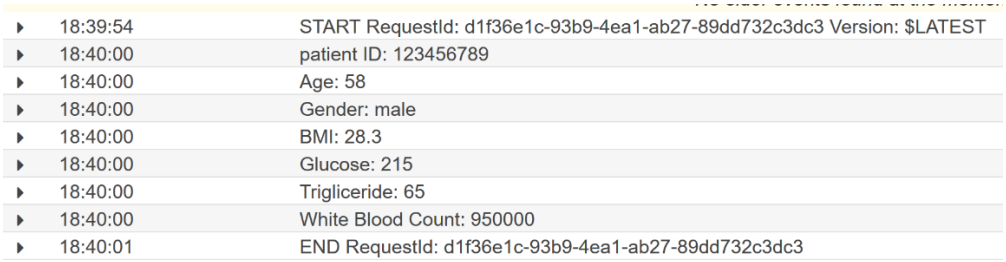
```
using System.Xml;
using System.IO;
using System.Text;
```

Also remove this line that we just added for debugging:

5. We can use XML parsing (and XPath) to parse the XML and extract all relevant data we are interested in. But first you need to read the content of the file. You can do that using code like this:

```
string bucketName = s3Event.Bucket.Name;
string objectKey = s3Event.Object.Key;

Stream stream = await S3Client.GetObjectStreamAsync(bucketName, objectKey, null);

using (StreamReader reader = new StreamReader(stream))
{
    string content = reader.ReadToEnd();
    // Now you have the text content of the file

    reader.Close();

}
```

6. Parse the XML using XPath and print the extracted details to CloudWatch. For example, something like this:



18:39:54	START RequestId: d1f36e1c-93b9-4ea1-ab27-89dd732c3dc3 Version: $LATEST
18:40:00	patient ID: 123456789
18:40:00	Age: 58
18:40:00	Gender: male
18:40:00	BMI: 28.3
18:40:00	Glucose: 215
18:40:00	Trigliceride: 65
18:40:00	White Blood Count: 950000
18:40:01	END RequestId: d1f36e1c-93b9-4ea1-ab27-89dd732c3dc3

(Remember that we covered XPath in Module2. Go back to that module if you need to refresh your memory about XPath syntax).

7. In this case we made the assumption that the files we expect are XML files. But they could be something else (e.g., json, or something else) and we can still parse them.

8. Exercise to do on your own:

   Enclosed with this module is a JSON file (movie1.json). Modify the code of your ProcessS3Event project to parse and print to CloudWatch the content of movie1.json when someone adds it to an S3 bucket that your lambda has it configured as a trigger.

   You learned how to parse JSON in Module 3. Go back there, see what was done, and adapt it to your lambda function.

9. In some cases you need variables in your code that their values might change. For example, assume you are writing a Lambda function that needs to call some URL endpoint (like a web

service). This URL is likely to customer-specific and is different for every customer you are selling your cloud application to. In this case it is not appropriate to hardcode this URL value in your code. If you do that you would need to maintain separate code bases for different customers. Instead, you want to make the value of this URL a variable that your Lambda can read at run time. Lambda provides the concept of an environment variable.

Login to the AWS console, go to the Lambda service, and click on your lambda function to open it.

Click the **Configuration** tab. Then click **Environment variables** from the list on the left.
Click **Edit**, then **Add environment variable**.
Here you can define an environment variable by giving it a key and a value. For example:

CUSTOMER_URL https://domain/resource/path

You can define as many environment variables as your code needs. You can then read the values of these environment variables from your code using code like this:

```
System.Environment.GetEnvironmentVariables()
```

(See documentation here).

What to Submit

Nothing to submit for this module. No quiz.