International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

AWS S3 Bucket Connectivity Using Azure Functions

Naveen Muppa
10494 Red Stone Dr
Collierville, Tennessee

Abstract

The paper focus is on connectivity setup procedure between Logic Apps and AWS S3 Bucket using GET and PUT method. Before proceeding to the next section, we assume a valid Logic Apps subscription and AWS S3 bucket details (along with client_id and client secret) is already in place.

Keywords: Due to absence of S3 connector in Logic Apps, connectivity setup is dealt with via Azure Function and HTTP connector in Logic Apps for download/upload of data. Azure Function is designed to list the files (S3 bucket content), handle Unicode characters and large data transfer (for inbound and outbound transactions)

I. INTRODUCTION

Azure Functions is a serverless computing service provided by Microsoft Azure, allowing you to run small pieces of code (functions) without having to worry about managing the underlying infrastructure.
Azure Functions supports multiple programming languages such as C#, JavaScript, Python, etc. You can use the AWS SDK for the language of your choice within your Azure Function to interact with S3. For example, if you're using C#, you can use the AWS SDK for .NET.
To access resources in your AWS S3 bucket, you'll need to provide appropriate AWS credentials (Access Key ID and Secret Access Key) to your Azure Function. These credentials should be securely stored and managed, such as using Azure Key Vault or Azure Managed Identities.
Ensure that the IAM (Identity and Access Management) role associated with the AWS credentials has the necessary permissions to access the S3 bucket and perform the required operations (read, write, etc.). You can define a specific IAM role with the least privilege principle for your Azure Function.

Azure Functions can be triggered by various events such as HTTP requests, timers, or messages from Azure services. When integrating with S3, you might want your function to trigger when a new file is uploaded to the S3 bucket or when an existing file is modified.
Once triggered, your Azure Function can execute custom logic to process the files in the S3 bucket. This could involve reading the file content, performing transformations or analysis, and possibly writing the results back to another storage location or triggering further actions.
Implement robust error handling within your Azure Function to handle any exceptions that might occur during the interaction with S3. Additionally, utilize logging mechanisms provided by Azure Functions to track the execution flow and diagnose issues.

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

II. LISTING AND DOWNLOADING FILES FROM S3 BUCKET VIA GET METHOD

Download Files from AWS S3 Bucket: Below mentioned steps will help to list files from a S3 bucket and download the same via Logic Apps:
Using Azure Function, file listing is performed. Bucket Name, File Path and Region details are passed as an input to the function in Logic Apps.
Using below code snippet we can list the files residing in AWS S3 bucket. Client_Id and Client secret details are needed to be updated in the codebase accordingly. The function expects its input to be bucket Name, file path and region details which are to be passed from Logic Apps. Depending upon these details, file listing is performed.
Code snippet:
Function Declaration along with necessary variables like client_id, client secret, region, bucket Name etc. are initialized. It is designed to assign dynamic values at runtime from Body Data.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
using System.Net.Http;
using Amazon.S3;
using Amazon.S3.Model;
using System.Net;
using System.Text;
using System.Collections.Generic;
using System.Collections;
using Amazon;

namespace <your namespace>
{
    public static class <your class name>
    {
        [FunctionName(<"your function name">)]
        public static async Task<HttpResponseMessage> Run([HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
            ILogger log, ExecutionContext context)
        {
            try
            {
                //throw new System.ArgumentException("Parameter cannot be null", "original");
                string sr = await new StreamReader(req.Body).ReadToEndAsync();
                dynamic BodyData = JsonConvert.DeserializeObject(sr);

                log.LogInformation("C# HTTP trigger function processed a request.");

                string clientid = <AWS ClientId Details>;
                string clientsecret = <AWS Clientsecret Details>;

                RegionEndpoint bucketRegion = RegionEndpoint.USEast1;

                string Region = BodyData.region != null ? BodyData.region : BodyData.region;
                string bucketName = BodyData.bucketName != null ? BodyData.bucketName : BodyData.bucketName;
                string objectKey = BodyData.filePath != null ? BodyData.filePath : BodyData.filePath;
```

Figure 1: Sample Code Snipper

Codebase Changes in Logic Apps: We assume azure function is already deployed in the Resource Group.

- Step-1: Add an action and in “choose an operation” dialogue box search for Azure function.

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

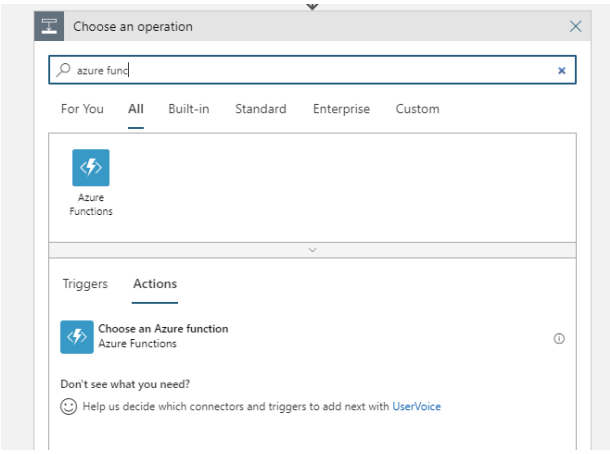

Figure 2: Azure Functions

III. DOWBLOAD THE LISTED FILES FROM AWS S3 BUCKET

The objective of this section is to download 'filename1.txt' (as shown in previous file listing walkthrough section) from AWS S3 bucket file path.

Azure Function Codebase for downloading files: Function App Codebase is designed to dynamically accept Region, Bucket Name, HTTP and File path details as shown below:
Defining namespace, class name and function name in association with necessary assemblies.
Declaring constant variables like EMPTY_BODY_SHA256, SCHEME, ALGORITHM etc is done in this portion of the codebase. Region, Bucket Name, Object Key and HTTP Method values are dynamically initialized during runtime.

```
log.LogInformation("C# HTTP trigger function processed a request.");
//string name = req.Query["name"];

const string EMPTY_BODY_SHA256 = "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855";
// some common x-amz-* parameters
const string X_Amz_Date = "X-Amz-Date";
const string X_Amz_Content_SHA256 = "X-Amz-Content-SHA256";
string hash256 = EMPTY_BODY_SHA256;
const string SCHEME = "AWS4";
const string ALGORITHM = "HMAC-SHA256";
const string TERMINATOR = "aws4_request";
const string Service = "s3";
string clientid = <AWS Clientid>;
string clientsecret = <AWS Clientsecret>;

//string clientid = System.Configuration.ConfigurationManager.AppSettings["ClientIdFromKeyVault"];
//string clientsecret = System.Configuration.ConfigurationManager.AppSettings["ClientSecretFromKeyVault"];

//string clientsecret = System.Configuration.ConfigurationManager.AppSettings["ClientSecretFromKeyVault"];
string Region = BodyData.region != null ? BodyData.region : BodyData.region;
string bucketName = BodyData.bucketName != null ? BodyData.bucketName : BodyData.bucketName;
string objectKey = BodyData.filePath != null ? BodyData.filePath : BodyData.filePath;
string HttpMethod = BodyData.httpMethod != null ? BodyData.httpMethod : BodyData.httpMethod;

log.LogInformation(Region + " : " + bucketName + " : " + objectKey);

//string ObjectContent = postdata.fileContent;

string returnDateTime = "";
Regex CompressWhitespaceRegex = new Regex("\\s+");
```

Figure3: Sample code Snippet

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

Single file named "filename1.txt" (as shown in previous example) is downloaded.

Select the desired Azure function in Logic Apps interface to download files from AWS S3 bucket. In below snippet we can see BucketName, FilePath, HTTP Method and Region details are dynamically passed. Depending upon this attributes function app internally computes signature using SHA-256 algorithm. Post successful function execution Authorization Headers and Hash Key is received.

Authorization and Hash Key details are mandatory attributes which need to be passed in HTTP header to download files from S3 bucket. To proceed with the same the output of the previous step needs to be parsed with Parse JSON action. Input Content of this action is the output body of the previous Function App. Schema structure can be defined manually where you define how the data needs to be parsed. It can be also generated in an automated manner by using "Use sample payload to generate schema". In this option JSON payload of the function app output can be provided to generate the schema structure.

IV. POSTING FILES TO S3 BUCKET VIA PUT METHOD

Upload Files to AWS S3 Bucket: We assume FileName1.txt is stored in Azure File Storage which needs to be processed towards AWS. Below mentioned steps will help to put files to a S3 bucket via Logic Apps: Get file from Azure Storage: We choose "Get File" action of Azure Storage in Logic Apps to fetch FileName1.txt details.

AWS Signature Calculator and S3 File Posting: While sending any data to S3 bucket, we need to pass Authorization and Hash Key details in the request header. At runtime AWS calculates signature depending upon the data content and signed header values. If the Signature passed in header, matches with the AWS Signature generated, data is successfully posted in S3 Bucket. Using Azure Function here, we are calculating the Signature of the data content for successful data posting via Logic Apps.

Azure Function Codebase for listing files: Using below code snippet we can calculate the Signature for successful file posting in AWS S3 Bucket. Function expects dynamic input like bucketName, filepath, region, Data content details which are to be passed from Logic Apps. Depending upon these details, Signature is calculated.

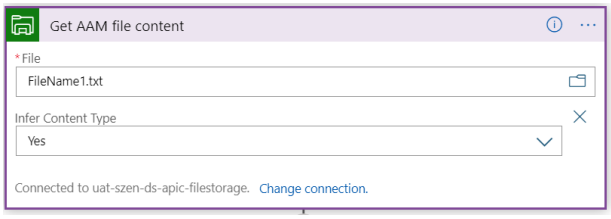

Figure 4: Filename sample

LOGIC APPS IMPLEMENTATION FOR AWS SIGNATURE CALCULATION AND DATA POSTING
In this scenario we assume a single file named "FileName1.txt" (as shown in previous example) is passed as an input to the Azure function along with other necessary inputs.

Select the desired Azure function in Logic Apps interface to calculate signature before posting the file to S3 Bucket. In below snippet we can see Bucket Name, File Path, HTTP Method and Region details and File Content are dynamically passed. Depending upon this attributes function app internally computes signature using SHA-256 algorithm. Post successful function execution Authorization Headers and Hash Key is received.

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

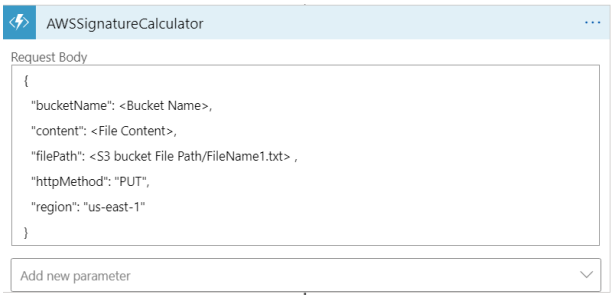

Figure 5: Aws signature calculator

Authorization and Hash Key details are mandatory attributes which need to be passed in HTTP header to upload files to S3 bucket. To proceed with the same the output of the previous step needs to be parsed with Parse JSON action. Input Content of this action is the output body of the previous Function App. Schema structure can be defined manually where you define how the data needs to be parsed. It can be also generated in an automated manner by using "Use sample payload to generate schema". In this option JSON payload of the function app output can be provided to generate the schema structure.

HTTP PUT request is initiated to post the desired file "FileName1.txt" to AWS S3 bucket. In request header, Authorization, Date, Hostname and SHA-256 Hash Key details need to be passed.

V. HTTP PUT REQUEST

On successful execution of the HTTP action, response is received in body in binary octet stream format.

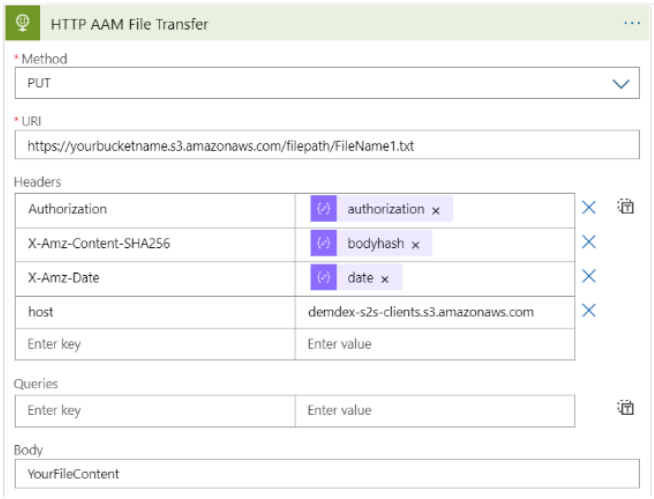

Figure 6: Sample put request

International Journal of Core Engineering & Management
Volume-7, Issue-05, 2023 ISSN No: 2348-9510

VI. LOGIC APPS CALLING AZURE GATEWAY API

Now that we have the signature done, we can call the AWS API Gateway. To do so, I used an HTTP connector because I didn't have much reuse. But the best way of reusing this would be to create a custom connector that I'll post about soon. Also, another thing that we can do is to configure properly the Open API specification of the Azure Function to avoid all the parsing and composing of JSON.

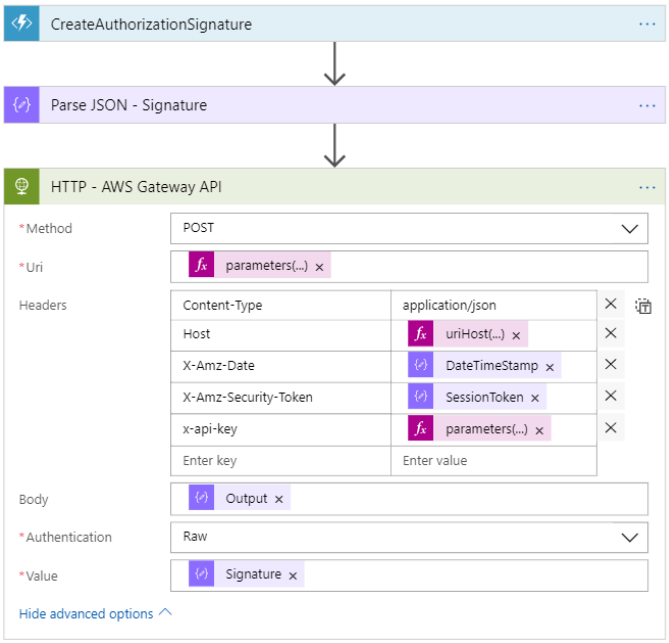

Figure 7: Authorization signature

VII. CONCLUSION

This document provides a walkthrough on how to connect with AWS S3 bucket from Azure Logic Apps for data posting and retrieval along with listing mechanism. Azure Function codebase for Signature Calculation is extensively tested with data content <=10MB. For data content >10MB additional changes might be required in the Azure Function codebase. In case of Signature Mismatch error, validation needs to be done whether similar attributes are passed as input of Azure Function and in header of GET/POST HTTP request for AWS S3 connect. For further debugging of Azure Function, Application Insight needs to be configured via Azure Portal to validate verbose log and test Function App.

REFERENCES

The following section summarizes the links to external resources that this document references. The aim of this section is to make it easier for you to add links to your own documentation.

[1] http://www.alessandromoura.com.br/2018/05/19/aws4-signature-logic-apps-azure-functions.
[2] https://docs.microsoft.com/en-us/azure/azure-functions/functions-twitter-email.