# Code

Link to Github Repository: <https://github.com/S00189824/Cloud_Project.git>

## Lambda Function for DynamoDb Backup with Cloud Watch

import datetime

import boto3

MAX\_BACKUPS = 3

dynamo = boto3.client('dynamodb')

def lambda\_handler(event, context):

if 'TableName' not in event:

raise Exception("No table name specified.")

table\_name = event['TableName']

create\_backup(table\_name)

delete\_old\_backups(table\_name)

def create\_backup(table\_name):

print("Backing up table:", table\_name)

backup\_name = table\_name + '-' + datetime.datetime.now().strftime('%Y%m%d%H%M%S')

response = dynamo.create\_backup(

TableName=table\_name, BackupName=backup\_name)

print(response)

def delete\_old\_backups(table\_name):

print("Deleting old backups for table:", table\_name)

backups = dynamo.list\_backups(TableName=table\_name)

backup\_count = len(backups['BackupSummaries'])

print('Total backup count:', backup\_count)

if backup\_count <= MAX\_BACKUPS:

print("No stale backups. Exiting.")

return

sorted\_list = sorted(backups['BackupSummaries'],

key=lambda k: k['BackupCreationDateTime'])

old\_backups = sorted\_list[:MAX\_BACKUPS]

for backup in old\_backups:

arn = backup['BackupArn']

print("ARN to delete: " + arn)

deleted\_arn = dynamo.delete\_backup(BackupArn=arn)

status = deleted\_arn['BackupDescription']['BackupDetails']['BackupStatus']

print("Status:", status)

return

## Contact Information Form Lambda Function

import json

import boto3

s3\_client = boto3.client('s3')

dynamyoDB = boto3.resource('dynamodb')

def lambda\_handler(event, context):

bucket = event['Records'][0]['s3']['bucket']['name']

json\_file = event['Records'][0]['s3']['object']['key']

json\_object = s3\_client.get\_object(Bucket = bucket,Key = json\_file)

jfilereader = json\_object['Body'].read()

jfile = json.loads(jfilereader)

table = dynamyoDB.Table('Contact')

table.put\_item(jfile)

## Important Visual Studio Code for AWS Cloud

private readonly string \_applicationName;

public Startup(IHostingEnvironment env)

{

var builder = new ConfigurationBuilder()

.SetBasePath(env.ContentRootPath)

.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)

.AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)

.AddEnvironmentVariables();

Configuration = builder.Build();

\_applicationName = Configuration["ApplicationName"];

}

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

services.AddDefaultAWSOptions(Configuration.GetAWSOptions());

services.AddAWSService<IAmazonDynamoDB>();

services.AddAWSService<IAmazonSQS>();

services.AddTransient<IContactDbRepository, ContactDbRepository>();

services.AddTransient<IContactQueueRepository, ContactQueueRepository>();

services.AddTransient<IContactService, ContactService>();

});

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)

{

loggerFactory.AddAWSProvider(this.Configuration.GetAWSLoggingConfigSection(),

formatter: (logLevel, message, exception) => JsonConvert.SerializeObject(

new Dictionary<string, string> {

{ "DateTimeUTC", DateTimeOffset.UtcNow.ToString("s") },

{"logLevel", logLevel.ToString() },

{"message", message?.ToString() },

{"exception", Common.GetJsonFromException(exception) }

}));

var logger = loggerFactory.CreateLogger<Program>();

logger.LogInformation($"New instance of {\_applicationName} started");

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCookiePolicy();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Home}/{action=Index}/{id?}");

});

}

## Contact Controller Code

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Logging;

using SampleContact;

using SampleContact.Models;

using SampleContact.Controllers;

using SampleContact.Services;

namespace SampleContact.Controllers

{

public class Contact : Controller

{

private IConfiguration \_configurable;

private readonly ILogger \_logger;

private IContactService \_contactService;

public Contact(IConfiguration configuration, ILogger logger, IContactService contactService)

{

\_configurable = configuration;

\_logger = logger;

\_contactService = contactService;

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public async Task<IActionResult> PostAsync([Bind("Name,Email,Phone,Comments")] ContactFormModel model)

{

model.IP = Common.ResolveIPAddress(HttpContext);

await \_contactService.AddAsync(model);

\_logger.LogInformation($"Contact added to queue. {model.LogSerialized}");

return RedirectToAction("Index");

}

}

}

## Contact Form Model

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SampleContact.Models

{

public class ContactFormModel

{

public string IP { get; set; } = "not-set";

public string Name { get; set; }

public string Email { get; set; }

public string Phone { get; set; }

public string Comments { get; set; }

public string LogSerialized

{

get

{

return $"IP: { IP } | Name: {Name} | Email: {Email} | Phone: {Phone} | Comments: {Comments}";

}

}

}

}

## Common Class

using Microsoft.AspNetCore.Http;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SampleContact

{

public static class Common

{

public static string ResolveIPAddress(HttpContext context)

{

return context?.Connection.RemoteIpAddress?.ToString();

}

public static string GetJsonFromException(Exception ex)

{

if (ex == null) return null;

return JsonConvert.SerializeObject(ex);

}

}

}

## Appsetting.json file

{

"Logging": {

"Region": "eu-west-1",

"LogGroup": "SampleContact",

"LogLevel": {

"Default": "Debug",

"System": "Information",

"Microsoft": "Information"

}

},

"AWS": {

"Region": "eu-west-1"

},

"AllowedHosts": "\*",

"ApplicationName": "SampleContact",

"SendMailQueueUrl": "https://sqs.eu-west-1.amazonaws.com/838476267186/Contact",

"ContactTableName": "Contact"

}

## Elastic Beanstalk json file

{

"region" : "eu-west-1",

"configuration" : "Debug",

"framework" : "net461",

"self-contained" : false,

"application" : "SampleContact",

"environment" : "SampleContact-dev",

"app-path" : "/",

"iis-website" : "Default Web Site",

"enable-xray" : false,

"enhanced-health-type" : "basic",

"additional-options" : "",

"solution-stack" : "64bit Windows Server Core 2019 v2.9.1 running IIS 10.0",

"environment-type" : "SingleInstance",

"cname" : "samplecontact-dev",

"instance-type" : "t3a.medium",

"key-pair" : "Linux Server Key",

"instance-profile" : "aws-elasticbeanstalk-ec2-role",

"service-role" : "aws-elasticbeanstalk-service-role"

}

## Contact Queue Repository

using Amazon.SQS;

using Amazon.SQS.Model;

using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SampleContact.Repository

{

public class ContactQueueRepository : IContactQueueRepository

{

private readonly IAmazonSQS \_sqsClient;

private readonly ILogger \_logger;

public ContactQueueRepository(ILogger<ContactQueueRepository> logger, IAmazonSQS sqsClient)

{

\_logger = logger;

\_sqsClient = sqsClient;

}

public async Task<bool> AddAsync(string contactId, string sendMailQueueUrl)

{

var sendRequest = new SendMessageRequest

{

QueueUrl = sendMailQueueUrl,

MessageBody = $"{{'ContactId' : {JsonConvert.SerializeObject(contactId)} }}"

};

var response = await \_sqsClient.SendMessageAsync(sendRequest);

return response.HttpStatusCode == System.Net.HttpStatusCode.OK;

}

}

}

## Contact Service

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Logging;

using SampleContact.Models;

using SampleContact.Repository;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SampleContact.Services

{

public class ContactService : IContactService

{

private readonly IContactDbRepository \_contactDbRepository;

private readonly IContactQueueRepository \_contactQueueRepository;

private readonly ILogger \_logger;

private IConfiguration \_configuration;

public ContactService(IConfiguration configuration,

ILogger<ContactService> logger,

IContactDbRepository contactDbRepository,

IContactQueueRepository contactQueueRepository)

{

\_logger = logger;

\_contactQueueRepository = contactQueueRepository;

\_contactDbRepository = contactDbRepository;

\_configuration = configuration;

}

public async Task<bool> AddAsync(ContactFormModel contactForm)

{

var id = Guid.NewGuid().ToString();

\_ = \_contactDbRepository.AddAsync(contactForm, id);

await \_contactQueueRepository.AddAsync(id, \_configuration["SendMailQueueUrl"]);

return true;

}

}

}

## Program

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().Run();

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.ConfigureLogging(logging =>

{

logging.AddAWSProvider();

logging.SetMinimumLevel(LogLevel.Debug);

})

.UseStartup<Startup>();