Archiving objects with Amazon Glacier and .NET

# Overview

Amazon Glacier is a secure, durable, and extremely low-cost cloud storage service for data archiving and long-term backup. It is designed to deliver 99.999999999% durability and provides comprehensive security and compliance capabilities that can help meet even the most stringent regulatory requirements. Amazon Glacier provides query-in-place functionality, allowing you to run powerful analytics directly on your archive data at rest.

For this walk-through, we’ll create a new .NET Core console application and add the AWS SDK for Glacier to it and walk through common functionality such as creating vaults, setting access policies on vaults, uploading archives to vaults, listing vaults and deleting archives/vaults.

# Prerequisites

* .NET Core 2.0 or higher installed
* AWS Account with credentials configured locally in Visual Studio or using the CLI
* Optional: Visual Studio 2017 (you can also use the command line for .NET Core)

# Create and Configure a Console Application

**Step 1: Create an empty console application project**

We’ll create a new directory, and then create the console app project in it using the following commands in either a Windows command prompt, Mac OS X bash shell or Linux bash shell:

1. mkdir glacier-sample
2. cd glacier-sample
3. dotnet new console

The *dotnet new* command will create the project files, and restore packages referenced by the template.

**Step 2: Add the Glacier NuGet Package**

**Command Line:**

Add the NuGet package *AWSSDK.Glacier* to the project with the following command in either a Windows command prompt or Mac OS X or Linux bash shell:

1. dotnet add package AWSSDK.Glacier

**Step 3: Edit the C# Code**

You can download the complete project here and then edit the *Program.cs* file or copy/paste the code here:

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using System.Threading.Tasks;

using Amazon;

using Amazon.Glacier;

using Amazon.Glacier.Model;

namespace TestGlacier

{

class Program

{

**static** string vaultName = "vault-" + Guid.NewGuid().ToString("n").Substring(0, 8);

**static** string key = "key-" + Guid.NewGuid().ToString("n").Substring(0, 8);

**static** string accountId = "YOUR ACCOUNT ID";

**static** void Main(string[] args)

{

**AmazonGlacierClient** glacier = new **AmazonGlacierClient**(RegionEndpoint.USWest2);

CreateVault(glacier);

SetVaultAccessPolicy(glacier);

AddTagsToVault(glacier);

ListVaults(glacier);

GetVaultAccessPolicy(glacier);

string archiveId = UploadArchive(glacier);

DeleteArchive(glacier, archiveId);

DeleteVault(glacier);

}

**private** **static** void DeleteArchive(**AmazonGlacierClient** glacier, string archiveId)

{

**DeleteArchiveRequest** req = new **DeleteArchiveRequest**();

req.VaultName = vaultName;

req.ArchiveId = archiveId;

**Task**<**DeleteArchiveResponse**> res = glacier.DeleteArchiveAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.wrte("Delete ArchiveId: {0}", archiveId);

}

}

**private** **static** string UploadArchive(**AmazonGlacierClient** glacier)

{

**MemoryStream** ms = new **MemoryStream**(Encoding.UTF8.GetBytes("data to archive"));

string treeHash = TreeHashGenerator.CalculateTreeHash(ms);

**UploadArchiveRequest** req = new **UploadArchiveRequest**();

req.VaultName = vaultName;

req.Body = ms;

req.Checksum = treeHash;

**Task**<**UploadArchiveResponse**> res = glacier.UploadArchiveAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.WriteLine("ArchiveId: {0}", res.Result.ArchiveId);

return res.Result.ArchiveId;

}

else

{

return string.Empty;

}

}

**private** **static** void DeleteVault(**AmazonGlacierClient** glacier)

{

**DeleteVaultRequest** req = new **DeleteVaultRequest**();

req.VaultName = vaultName;

**Task**<**DeleteVaultResponse**> res = glacier.DeleteVaultAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.WriteLine("Deleted Vault {0}", vaultName);

}

}

**private** **static** void SetVaultAccessPolicy(**AmazonGlacierClient** glacier)

{

string jsonPolicy = @"{

""Version"":""2012-10-17"",

""Statement"":[

{

""Sid"": ""glacier-perm"",

""Principal"": ""\*"",

""Effect"": ""Allow"",

""Action"": [

""glacier:\*""

],

""Resource"": [

""arn:aws:glacier:us-west-2:" + accountId + @":vaults/" + vaultName + @"""

]

}

]

}";

**SetVaultAccessPolicyRequest** req = new **SetVaultAccessPolicyRequest**();

req.VaultName = vaultName;

req.Policy = new **VaultAccessPolicy**();

req.Policy.Policy = jsonPolicy;

**Task**<**SetVaultAccessPolicyResponse**> res = glacier.SetVaultAccessPolicyAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.Write("Set Vault Access Policy on vault {0}", vaultName);

}

}

**private** **static** void CreateVault(**AmazonGlacierClient** glacier)

{

**CreateVaultRequest** req = new **CreateVaultRequest**();

req.VaultName = vaultName;

**Task**<**CreateVaultResponse**> res = glacier.CreateVaultAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.Write("Created Vault {0} successfully", vaultName);

}

}

**private** **static** void AddTagsToVault(**AmazonGlacierClient** glacier)

{

**AddTagsToVaultRequest** req = new **AddTagsToVaultRequest**();

req.VaultName = vaultName;

req.Tags = new **Dictionary**<string, string>();

req.Tags.Add("cost-center","1234");

req.Tags.Add("stack","production");

**Task**<**AddTagsToVaultResponse**> res = glacier.AddTagsToVaultAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.Write("Created Vault {0} successfully", vaultName);

}

}

**private** **static** void GetVaultAccessPolicy(**AmazonGlacierClient** glacier)

{

**GetVaultAccessPolicyRequest** req = new **GetVaultAccessPolicyRequest**();

req.VaultName = vaultName;

**Task**<**GetVaultAccessPolicyResponse**> res = glacier.GetVaultAccessPolicyAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

Console.Write("Vault Policy for {0}: {1}", vaultName, res.Result.Policy.Policy);

}

}

**private** **static** void ListVaults(**AmazonGlacierClient** glacier)

{

**ListVaultsRequest** req = new **ListVaultsRequest**();

req.Limit = 100;

**Task**<**ListVaultsResponse**> res = glacier.ListVaultsAsync(req);

Task.WaitAll(res);

if (res.IsCompletedSuccessfully)

{

foreach (var vault in res.Result.VaultList)

{

Console.WriteLine("Vault: {0}", vault.VaultName);

}

}

}

}

}

Figure 4 – Program.cs file

This sample code demonstrates a common pattern when developing with the AWS SDK for .NET, which is the use of a client object to represent an AWS service. That client object then exposes functionality via methods, such as the “*CreateVault*” method in this example.

Ensure you make the following changes in the code:

* Replace “*YOUR ACCOUNT ID*” with the account ID of the AWS account you are using.

# Use Glacier!

To run through the Glacier functionality using the console application, use the following command to build (compile) and run the app:

dotnet run

If you want to run the app again without compiling, just pass the --no-build flag like this:

dotnet run --no-build

The console application will run.