How to Guide

Azure Migration Scenario – AWS S3 to Azure Storage

Microsoft

AWS S3 to Azure Storage migration

June 2016

OVERVIEW

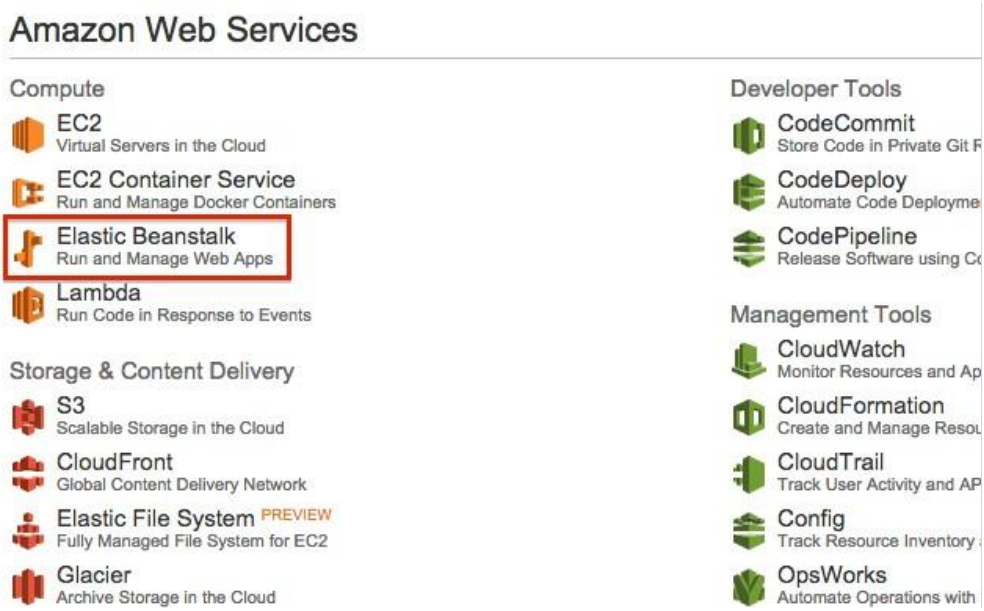
This article describes how to migrate a sample ASP.NET MVC Web Application from Amazon Web Services (AWS) to Azure Cloud Service.

PREREQUISITES

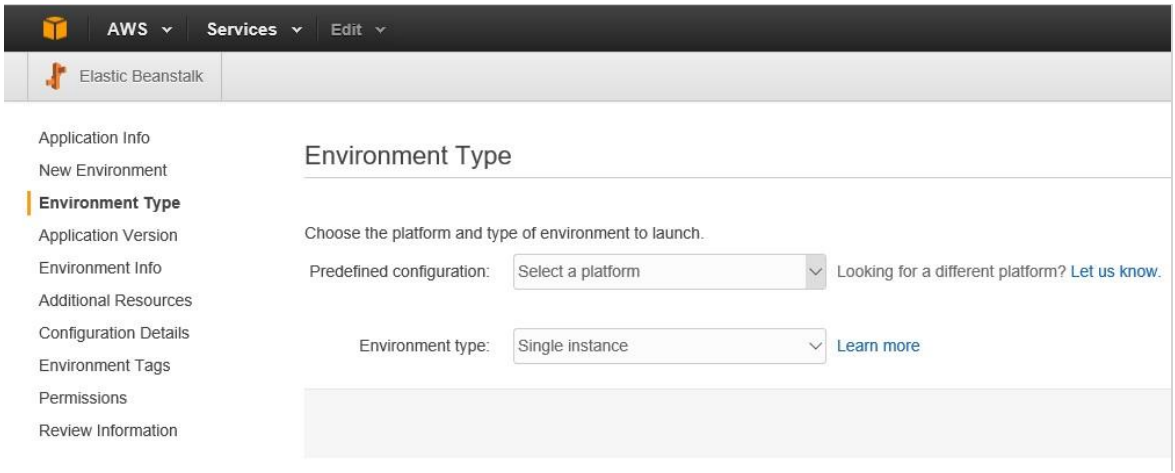
As the starting point, you have an ASP.NET MVC Web Application published to the AWS Elastic Beanstalk service. As the owner of the application you have access to its source code. It is required because you have to reimplement code responsible for the interaction with the Storage Service. Also we assume that your application is live, so you have some data to migrate as well.

If you do not know how to publish your application to the AWS Elastic Beanstalk service, please follow the instructions below:

1. Start from going to the AWS Management Console page - https://aws.amazon.com/console/ - and clicking Sign in to the AWS Console button.
2. Enter your login and password at the opened AWS Elastic Beanstalk service page. After that you will be redirected to the Amazon Web Services page. Find "Compute" area and choose "Elastic Beanstalk" item:



3. At the AWS Elastic Beanstalk dashboard, click on Create New Application to create and configure your application, then enter the Application name and the Description field. Click Next to continue.
4. The next step is to configure the environment - click on Create web server button, you will see the Environment Type page:



Here choose a Predefined configuration (for our example it should be "IIS") and Environment type ("Single instance"). Click Next.

5. Choose "Upload your own" radiobutton and select your project archive at the next page called Application Version ("*AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS*" (c)). Please make sure that your file meets the requirements (see AWS Documentation https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/usingfeatures.deployment.source.html). Click Next. Wait while the application version is uploading.
6. The next step is to enter your environment information: name, URL, description. Click Next. At the new page mark the checkbox next to Create this environment inside a VPC. Click Next.
7. Please leave Configuration Details, Environment Tags and Permissions pages unaltered. Turn to VPC Configuration page and select the first availability zone listed by checking the box under the EC2 column. Click Next.
8. Then review your environment configuration on the next screen and click Launch to deploy your application. Launching may take a few minutes.

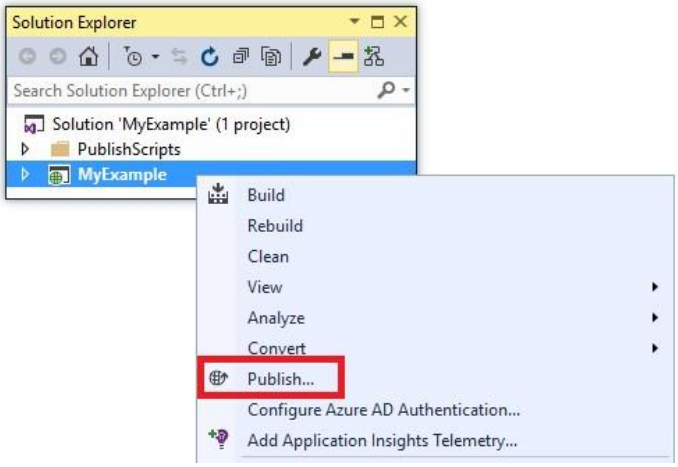
GOALS

1. Migrate the existing ASP.NET MVC Web Application from AWS to Azure.
2. Implement the new code to use the Azure Storage instead of the AWS S3.
3. Migrate the existing data from the AWS S3 to the Azure Storage.

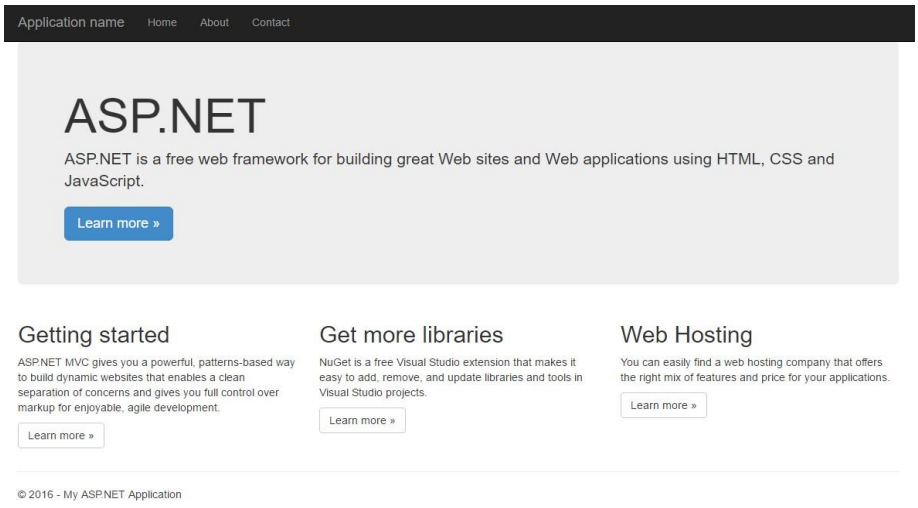
ASP.NET MVC WEB APPLICATION MIGRATION

Since you have a source code for the application you can create the Azure WebApp and publish your existing project there using standard procedure in Visual Studio. There are the instructions how to publish an ASP.NET MVC Web Application to Azure WebApp via Visual Studio:

1. When you have already created the project (we are using a template), right-click the project, and choose Publish:



2. In a few seconds, the Publish Web wizard appears. The wizard opens to a publish profile that has settings for deploying the web project to the new web app. The publish profile includes a user name and password for deployment. These credentials have been generated for you, and you don't have to enter them. The password is encrypted in a hidden user-specific file in the Properties\PublishProfiles folder.
3. At the opened window on the Profile tab choose "Microsoft Azure App Service" target.
4. At the opened window App Service fill Subscription and View fields, choose an App Service to publish.
5. On the Connection, Settings and Preview tabs enter the needed data and click Publish. Wait until you see the "Publish Succeeded" message. There will be the link to your web application, click it and ensure that the project has deployed successfully:



You can monitor and manage your web app in the Azure portal - https://portal.azure.com/.

BUSINESS LOGIC MIGRATION

The most interesting part here is the Storage Service interaction layer implementation. Best practices recommend using a generic interface to manage data in the storage. If you design the solution this way – you will be able to change the service provider very easily. Here is the example of the interface, which allows you to perform basic typical operations such as Upload, Enumerate and Delete (there is no need to get content because you have the URI to download it directly from storage):

```
public interface IImageService
{
    Task UploadImageAsync(string name, Stream inputStream);

    Task<IEnumerable<ImageModel>> GetImagesAsync();

    Task DeleteImageAsync(string name);
}
```

In the case of AWS S3 Service we are going to use standard AWSSDK.S3 NuGet package provided for easy access to S3 API. The implementation of the *IImageService* interface looks like:

```
public async Task UploadImageAsync(string name, Stream inputStream)
{
    await CreateBucketIfNotExistsAsync(ImageBucketName);

    var request = new PutObjectRequest
    {
        BucketName = ImageBucketName,
        Key = name,
        ContentType = "image/gif",
        InputStream = inputStream
    };

    await Client.PutObjectAsync(request);
}


public async Task<IEnumerable<ImageModel>> GetImagesAsync()
{
    IEnumerable<ImageModel> result = null;

    if (await IsBuckedExistAsync(ImageBucketName))
    {
        var request = new ListObjectsRequest
        {
            BucketName = ImageBucketName
        };

        var response = await Client.ListObjectsAsync(request);

        result = response.S3Objects.Select(o => new ImageModel {Name = o.Key});
    }

    return result;
}


public async Task DeleteImageAsync(string name)
{
    if (await IsBuckedExistAsync(ImageBucketName))
    {
        var request = new DeleteObjectRequest
        {
            BucketName = ImageBucketName,
            Key = name
        };

        await Client.DeleteObjectAsync(request);
    }
}
```

Here is the specific implementation of the interface for the Azure Blob Storage Service (using WindowsAzure.Storage NuGet package):

```
public async Task UploadImageAsync(string name, Stream inputStream)
{
    // Get the container
    var blobContainer = await GetCloudBlobContainer(ContainerName);

    // Get block blob
    var blockBlob = blobContainer.GetBlockBlobReference(name);

    // Save block blob
    await blockBlob.UploadFromStreamAsync(inputStream);

    // Set content type
    blockBlob.Properties.ContentType = "image/gif";
    await blockBlob.SetPropertiesAsync();
}


public async Task<IEnumerable<ImageModel>> GetImagesAsync()
{
    IEnumerable<ImageModel> result = null;

    if (await IsBlobExistAsync(ContainerName))
    {
        var blobContainer = await GetCloudBlobContainer(ContainerName);
        var blobs = blobContainer.ListBlobs().ToArray();

        result = blobs
            .OfType<CloudBlockBlob>()
            .Select(b => new ImageModel { Name = b.Name });
    }

    return result;
}


public async Task DeleteImageAsync(string name)
{
    if (await IsBlobExistAsync(ContainerName))
    {
        // Get the container
        var blobContainer = await GetCloudBlobContainer(ContainerName);

        // Get block blob
        var blockBlob = blobContainer.GetBlockBlobReference(name);

        // Delete block blob
        await blockBlob.DeleteAsync();
    }
}
```

As you can see both implementations are similar with the only difference in the libraries used to access the Storage Client. You can find the link to download the whole solution for the VS2015 at the end of the article.

EXISTING DATA MIGRATION

The migration of data from the Amazon S3 to the Azure Storage can be produced in the following ways based on the amount of data you have:

I. minimal data

You can copy the data manually: download information from the AWS S3 manually and upload to the Azure Storage manually (maybe even using your Web Application).

II. a lot of data

You can implement an application which will receive the list of the files from the AWS S3, download them and upload to the Azure Storage. Please note that you will have additional expenses for the network traffic on both sides: Amazon and Azure. There are several existing tools in the internet (console and web) that you can use. Just keep in mind that you have to provide the secret keys to a tool, meaning you should make sure that your secret data will not be compromised.

- Azure Copy: https://azure.microsoft.com/en-us/documentation/articles/storage-use-azcopy/
- AWCopy: https://github.com/cicorias/AWCopy
- CloudBerry Cloud Migrator: http://www.cloudberrylab.com/cloud-migrator.aspx
- Code sample of the tool to copy the S3 object: http://gauravmantri.com/2012/06/14/how-to-copy-anobject-from-amazon-s3-to-windows-azure-blob-storage-using-copy-blob/
- Code sample of the tool to copy the S3 bucked: http://gauravmantri.com/2012/06/15/how-to-copy-abucket-from-amazon-s3-to-windows-azure-blob-storage-using-copy-blob/

III. a huge amount of data (data volume is larger than 100GB)

You can use the method described above but it may not be cost effective, as it causes a lot of traffic between the AWS S3 and the Azure Storage. The alternative way is the Import/Export Service in Amazon and Azure:

- Amazon Import|Export Service: https://aws.amazon.com/importexport/
- Azure Import|Export Service: https://azure.microsoft.com/en-us/documentation/articles/storage-importexport-service/