Using .NET(Csharp) Implemented Data Transfer - Azure Blob Storage to Web Application.

Problem Statement: -

- Our Company firstly use one specific server or database server to store files.
- We can upload only specific size of files. If we upload more then server file size then Getting error-file storage has been full.
- When we use database server: -
    - Massive increase in traffic between the web server and the database
    - Slowdown in database performance
    - Massive increase in the size of the database, resulting in a backup headache
    - If you store docs outside of the database you **will** have missing documents and broken links soomer not later. Your backup/restore scenario is a lot more complex: you have no way to ensure that all data is from the same point in time.

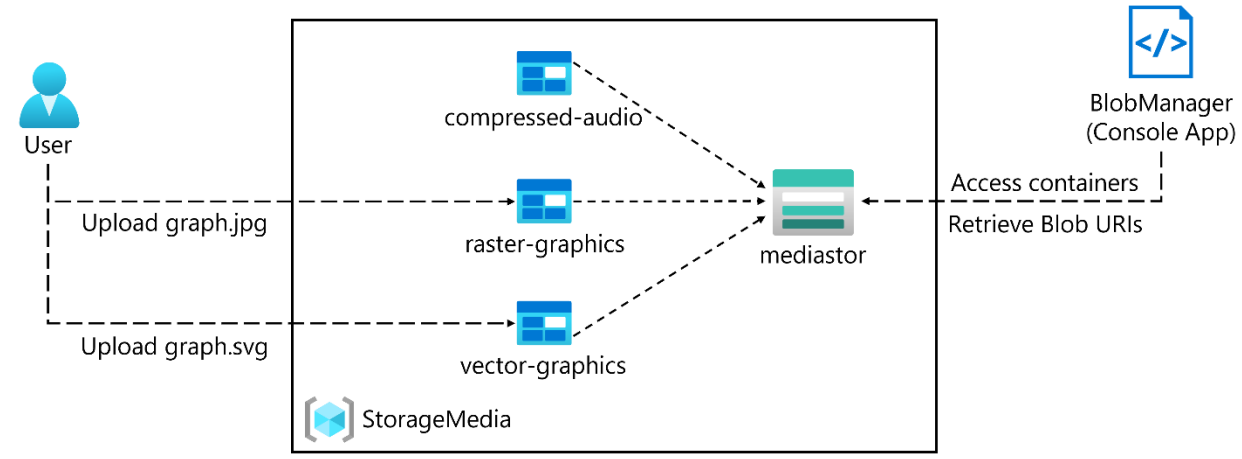
Introduction: -

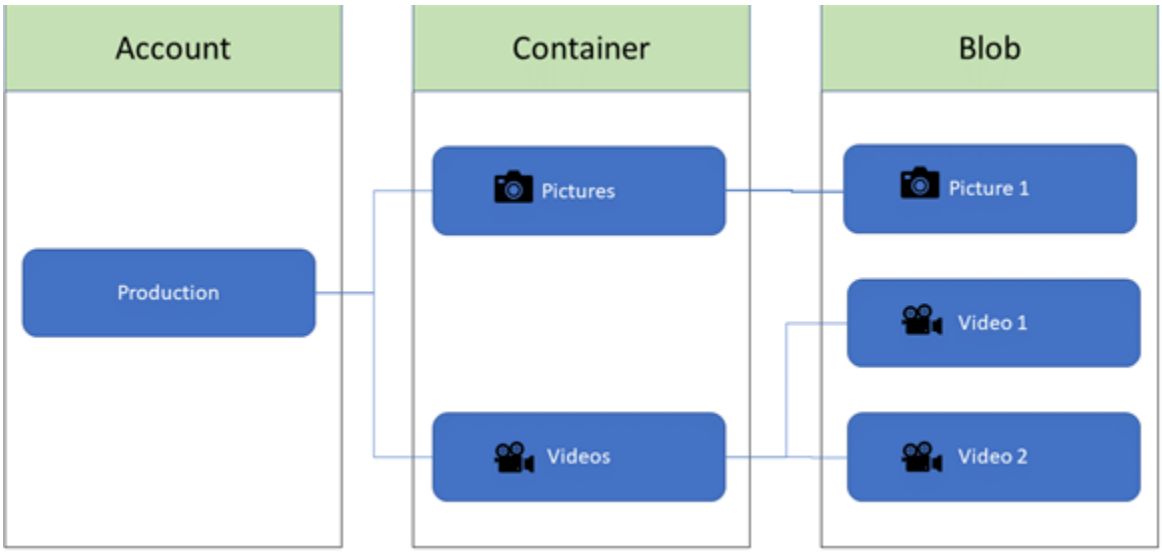
Microsoft offers several options to store data on the cloud. Each option has its unique purpose for serving different business needs. One of the significant capabilities that Microsoft Azure provides is the agility to migrate to different storage options if required.

There are various options available in the azure storage account for storing user data.

- Blob Storage
- File Storage
- Table Storage
- Queue Storage
- Disk

Architecture Diagram :





What is Azure Blob Storage?

Azure Blob Storage is an object storage solution for the cloud. Blob Storage allows you to store a massive amount of unstructured data. The unstructured data need not be of the specific data model.

What is Azure Blob Storage used for?

Azure Blob Storage was designed to serve specific needs. If your business use case needs to store unstructured data like audio, video, images, etc then you should probably go with this option. The objects which are being stored in Blob does not necessarily have an extension. The following points describe the use case scenarios:

- Serving images or documents directly to a browser
- Storing Files for distributed access
- Streaming video and audio
- Writing to log Files
- Storing data for backup, restore, disaster recovery and archiving
- Storing data for analysis by an on-premises or Azure-hosted service

How does BLOB storage work?

Blob Storage comprises of three different types of resources:

1. The **Storage Account**
2. A **container** in the storage account
3. A **blob** in a container

Create a Azure Storage resource using Azure Portal :-

Create a storage account ...

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review + create

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription * : Visual Studio Professional

Resource group * : azuretraining

Create new

Instance details

If you need to create a legacy storage account type, please click here.

Storage account name * :

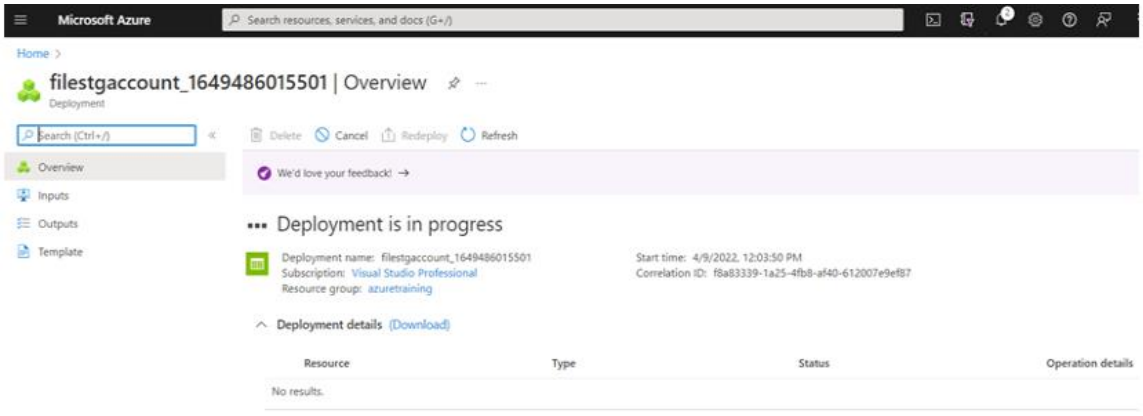
Region * : (US) East US

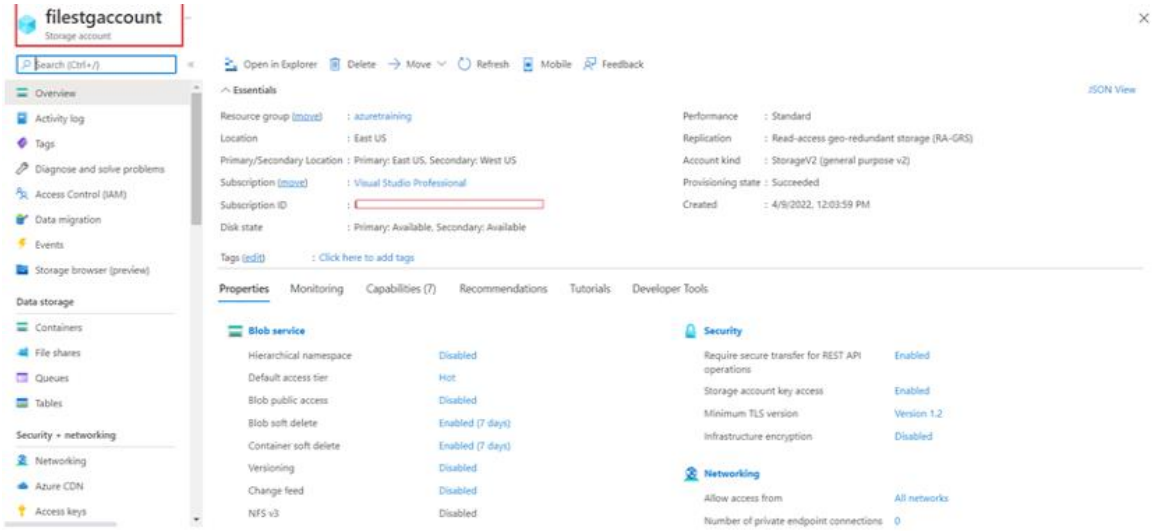
Performance * :

- (•) **Standard:** Recommended for most scenarios (general-purpose v2 account)
- ( ) **Premium:** Recommended for scenarios that require low latency.

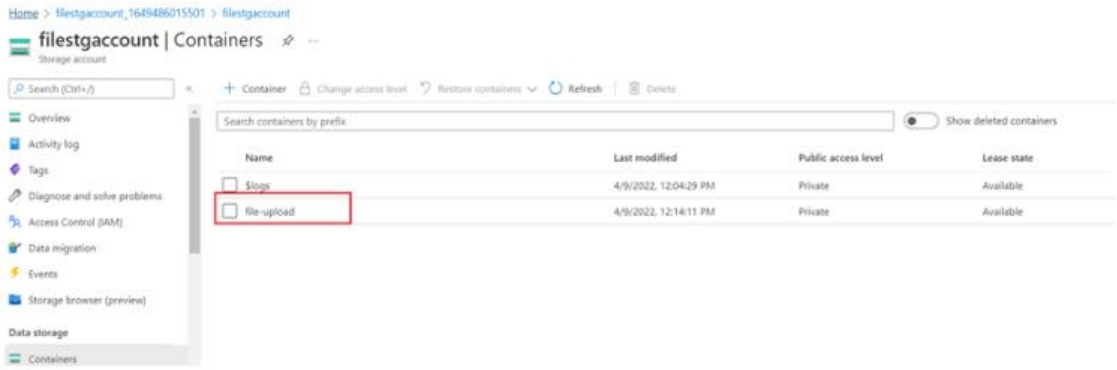
Redundancy * : Geo-redundant storage (GRS)

- [x] Make read access to data available in the event of regional unavailability.





Next step is to create an Azure function that uploads files into the “file-upload” container.



Code :

```
{
    "IsEncrypted": false,
    "Values": {
        "AzureWebJobsStorage": "<replace your blob storage connection
key here>",
        "ContainerName": "file-upload", // Container name
        "FUNCTIONS_WORKER_RUNTIME": "dotnet"
    }
}
```

```
using Azure.Storage.Blobs;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.Extensions.Logging;
using System;
using System.IO;
using System.Threading.Tasks;
namespace FileUploadFunction {
    public static class FileUpload {
        [FunctionName("FileUpload")]
        public static async Task < IActionResult > Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "post", Route =
null)] HttpRequest req, ILogger log) {
            string Connection =
Environment.GetEnvironmentVariable("AzureWebJobsStorage");
            string containerName =
Environment.GetEnvironmentVariable("ContainerName");
            Stream myBlob = new MemoryStream();
            var file = req.Form.Files["File"];
            myBlob = file.OpenReadStream();
            var blobClient = new BlobContainerClient(Connection,
containerName);
            var blob = blobClient.GetBlobClient(file.FileName);
            await blob.UploadAsync(myBlob);
            return new OkObjectResult("file uploaded successfylly");
```

```
        }
    }

}
```

```
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Models;
using Azure.Identity;

// TODO: Replace <storage-account-name> with your actual storage account name
var blobServiceClient = new BlobServiceClient(
        new Uri("https://<storage-account-name>.blob.core.windows.net"),
        new DefaultAzureCredential());

//Create a unique name for the container
string containerName = "quickstartblobs" + Guid.NewGuid().ToString();

// Create the container and return a container client object
BlobContainerClient containerClient = await
blobServiceClient.CreateBlobContainerAsync(containerName);

// Create a local file in the ./data/ directory for uploading and downloading
string localPath = "data";
Directory.CreateDirectory(localPath);
string fileName = "quickstart" + Guid.NewGuid().ToString() + ".txt";
string localFilePath = Path.Combine(localPath, fileName);

// Write text to the file
await File.WriteAllTextAsync(localFilePath, "Hello, World!");

// Get a reference to a blob
BlobClient blobClient = containerClient.GetBlobClient(fileName);

Console.WriteLine("Uploading to Blob storage as blob:\n\t {0}\n",
blobClient.Uri);

// Upload data from the local file
await blobClient.UploadAsync(localFilePath, true);

Console.WriteLine("Listing blobs...");

// List all blobs in the container
await foreach (BlobItem blobItem in containerClient.GetBlobsAsync())
{
    Console.WriteLine("\t" + blobItem.Name);
}
```

```
// Download the blob to a local file
// Append the string "DOWNLOADED" before the .txt extension
// so you can compare the files in the data directory
string downloadFilePath = localFilePath.Replace(".txt", "DOWNLOADED.txt");

Console.WriteLine("\nDownloading blob to\n\t{0}\n", downloadFilePath);

// Download the blob's contents and save it to a file
await blobClient.DownloadToAsync(downloadFilePath);

// Clean up
Console.Write("Press any key to begin clean up");
Console.ReadLine();

Console.WriteLine("Deleting blob container...");
await containerClient.DeleteAsync();

Console.WriteLine("Deleting the local source and downloaded files...");
File.Delete(localFilePath);
File.Delete(downloadFilePath);


Console.WriteLine("Done");
```

Challenges Faced :

Integration complexity: Syncing Azure Blob Storage with a .NET application requires careful integration.

Data consistency: Ensuring data consistency between the application and Blob Storage can be challenging.

Scalability considerations: Handling growing data volumes while maintaining performance is a challenge.

Error handling and logging: Robust error handling and logging mechanisms are necessary for troubleshooting.

Security and access control: Managing access control and security during synchronization can be complex.

Business Benefit :-

- **Perfect for small businesses and established enterprises** — Azure is designed for every business regardless of size, from the local bakery to multi-national corporations. It's easily scalable to meet your IT demands and operates on a pay-as-you-go pricing model to meet any budget. Since businesses can launch and store internal and external applications in the cloud, it also saves on in-house IT costs, including hardware and maintenance.
- **Complements and expands your current IT infrastructure** — Azure allows your IT personnel to focus on your business without having to worry about in-house capabilities or maintaining equipment that is over-taxed or underused. The platform makes it fast and easy to deploy your current apps with little to no downtime. An integrated development environment reduces the learning curve, allowing teams to master the platform quickly. Additionally, the platform has a footprint in more countries than Google or Amazon, providing faster content delivery while optimizing the user experience. Azure is scalable to grow with your company, and you pay for only what you need.
- **Leading the way with IaaS and PaaS** — At the forefront of IaaS and PaaS, Azure offers rapid deployments. The hybrid cloud environment allows companies to select whether they operate autonomously or utilize a public cloud. You are also able to decide the level at which you are connected to the internet, if at all. Meet all your IT and service needs without having to maintain the underlying infrastructure.
- **Security, compliance, and disaster recovery** — Microsoft understands the importance of security and has designed Azure to stay ahead of the competition when it comes to protecting your data. Azure has many compliance certifications and is a top choice of high-risk industries such as health care and government to provide cloud services. Both the platform and end users are protected. Additional services such as multi-factor authentication and sophisticated disaster recovery abilities that can restore data in a matter of hours further address business needs.

- **Industry-specific applications** — Due to the high-risk and sensitive nature of certain industries, Azure has designed specific applications to address unique needs. Government, health care, manufacturing, and financial services benefit greatly from Azure's many features, including offline cloud services, individualized security needs, simplified compliance, and modernized customer apps.

References:

- **Azure RBAC**
  action: Microsoft.Storage/storageAccounts/blobServices/containers/blobs/write (for writing to an existing blob)
  or Microsoft.Storage/storageAccounts/blobServices/containers/blobs/add/action (for writing a new blob to the destination)
- Microsoft.Storage/storageAccounts/blobServices/containers/blobs/read

- by Priyanka Dingankar
Senior Full Stack Developer