Deploying a Static website on Microsoft Azure

The main aim of this project is to host a static website on Azure using a counter function. Throughout this project, Microsoft Azure services will be used, and the counter will be updated whenever someone visits the website.

Setting up Azure Blob storage for static website hosting.

The blob storage has been used as it's highly scalable and cost-effective for storing static files, and mainly one need not worry about managing the VM or the OS. Additionally, the website will be available for people around the globe.

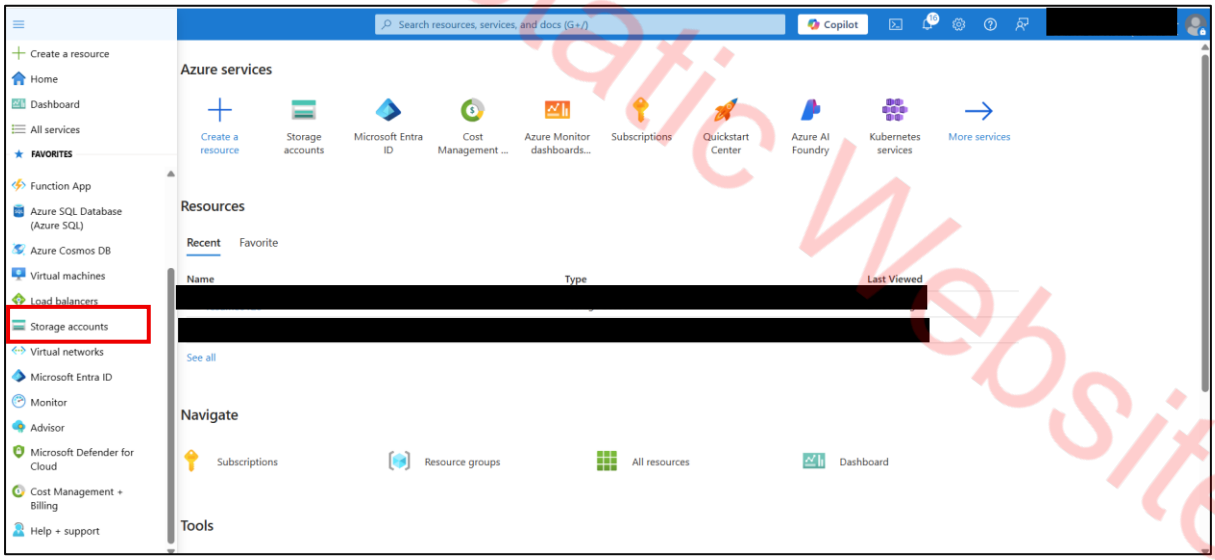
Below are the steps followed during the setup.

Step 1: Design a static website.

Step 2: Create an Azure account, if you don't have one.

Step 3: Create a Blob storage account.

In the left navigation panel, select the storage account.



"Create a storage account", and under the first section you'll see a section called "Preferred storage type" in which there are three types: blob, files and other. We are opting for Blob Storage.

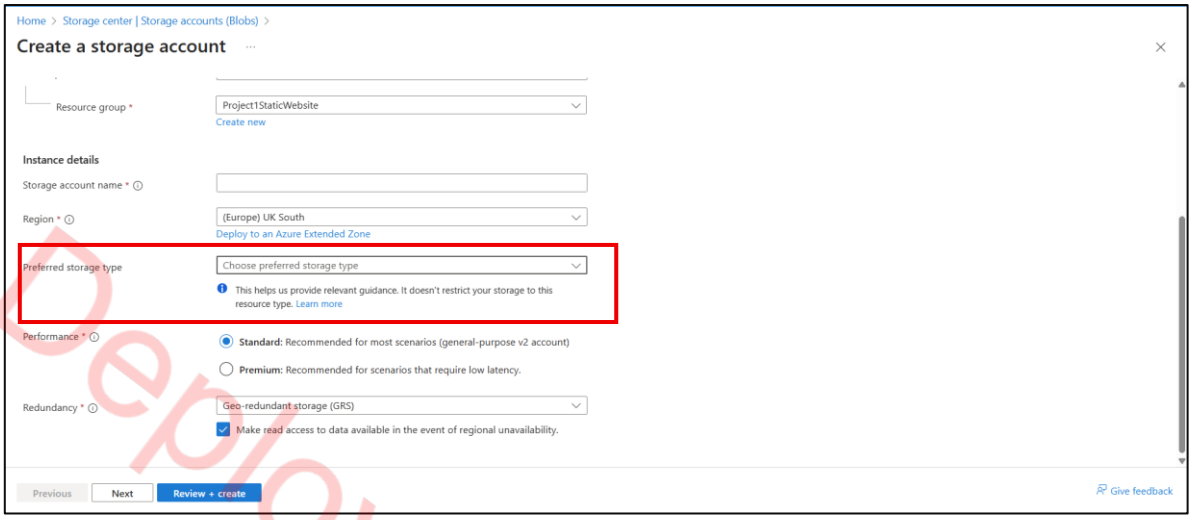
'Blob' means it stores unstructured data such as images, files, audio, video and much more.

File storage offers fully managed file sharing with the help of the cloud using protocols such as SMB (Server Message Block) and NFS (Network File System).

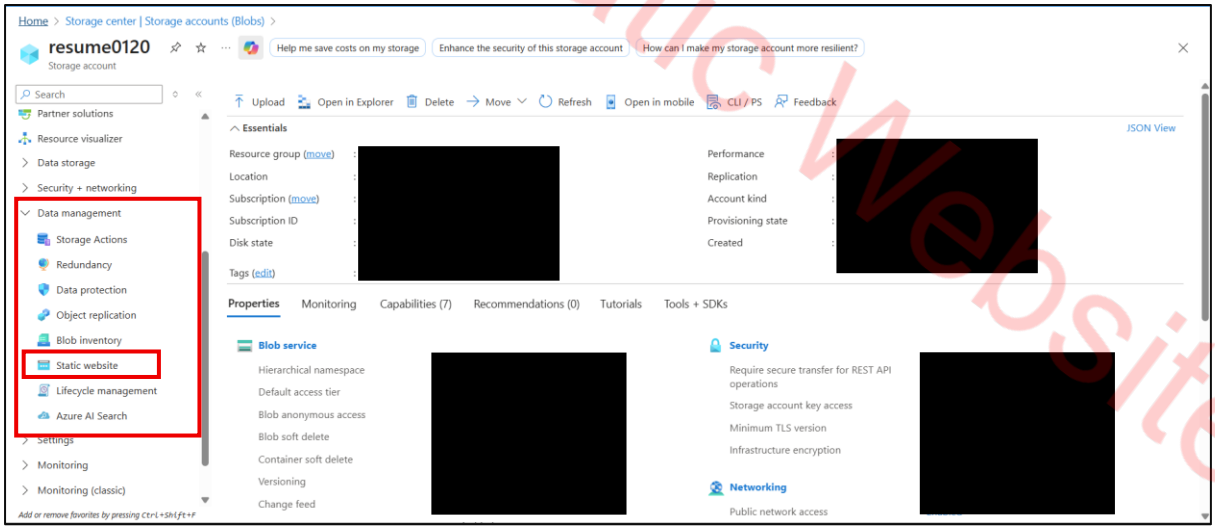
SMB file shares are accessible from Windows, Linux and macOS clients.

NFS file shares are only accessible from the Linux clients.

Deploying a Static website on Microsoft Azure

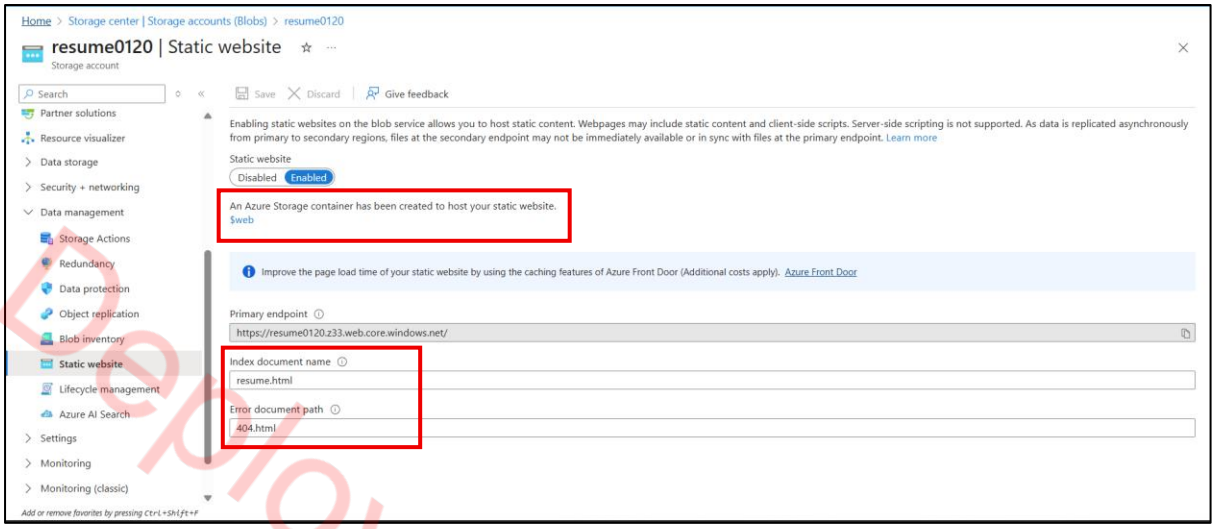


Step 4: Once the storage account is created, there's a section named "Data Management" under which the "Static Website" section that's by default disabled should be enabled.



Upon successfully enabling the static website, one needs to provide an index document name and an error document path.

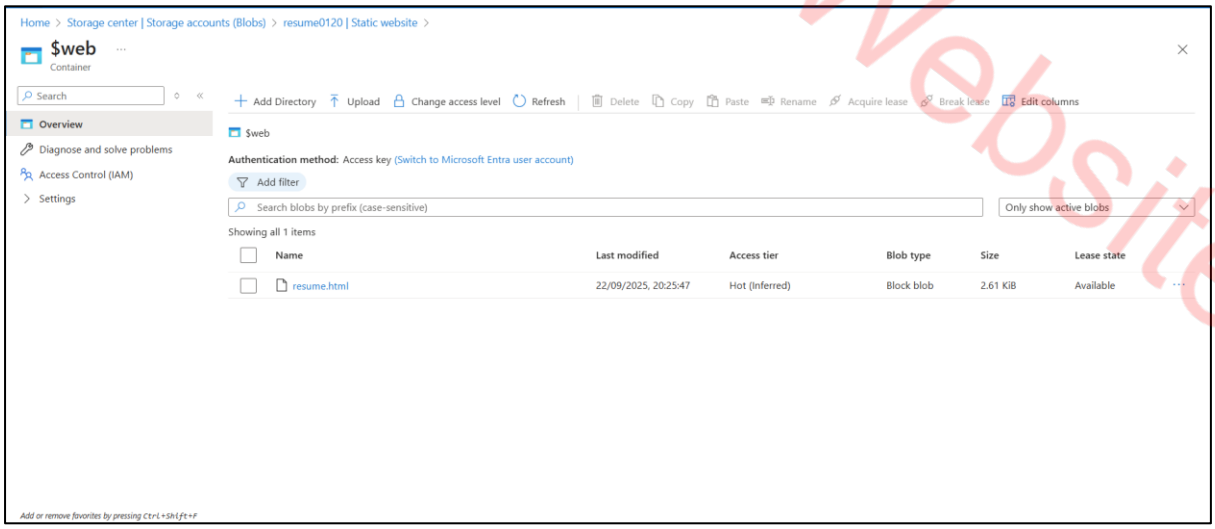
Deploying a Static website on Microsoft Azure



The **Index Document Name** should be the name of the webpage, that the Azure storage returns when a request has been made to the root of the website/webpage. By default, the name should be in lowercase, as the file names used during the web hosting are case-sensitive.

The **Primary Endpoint** is a link that's been used to view/access the webpage.

Additionally, **$web** is a container where the website link needs to be uploaded. When uploading the file, the name of the file should be the same as the index document name. In case if the name is not the same, the website will display 404 error.



Deploying a Static website on Microsoft Azure

Creation of a CosmosDB to store the number of visitors.

As we need to store the number of visitors visited the website, NoSQL is the best approach as its unstructed. This means that we don't need to create a specific table structure. Instead, we just need to store the key value pair. Additionally, the CosmosDB has been built to handle massive amount of data and read/write operations at a global scale.

You must be wondering why we haven't used SQL database. The main reason is that there's no complex structure that is required in this project. Also, SQL is more costly and require more time for setup and management.

Steps to create a CosmosDB in Azure Portal.

Step 1: Create a CosmosDB account.

Step 2: Enter the requested details such as account name, location, capacity mode and key based authentication method.

In the capacity mode there are two options:

Provisioned Throughput is selected when can predict the workload and consistency of the website.

Serverless is mainly used everywhere as one doesn't need to handle or manage the infrastructure or VM, the cloud provider does that for you. This is opted when there's unpredictable traffic.

Step 3: Review + Create the details. Upon successfully deploying the resource, select "Go To Resource" and in the left side panel select "Data Explorer".

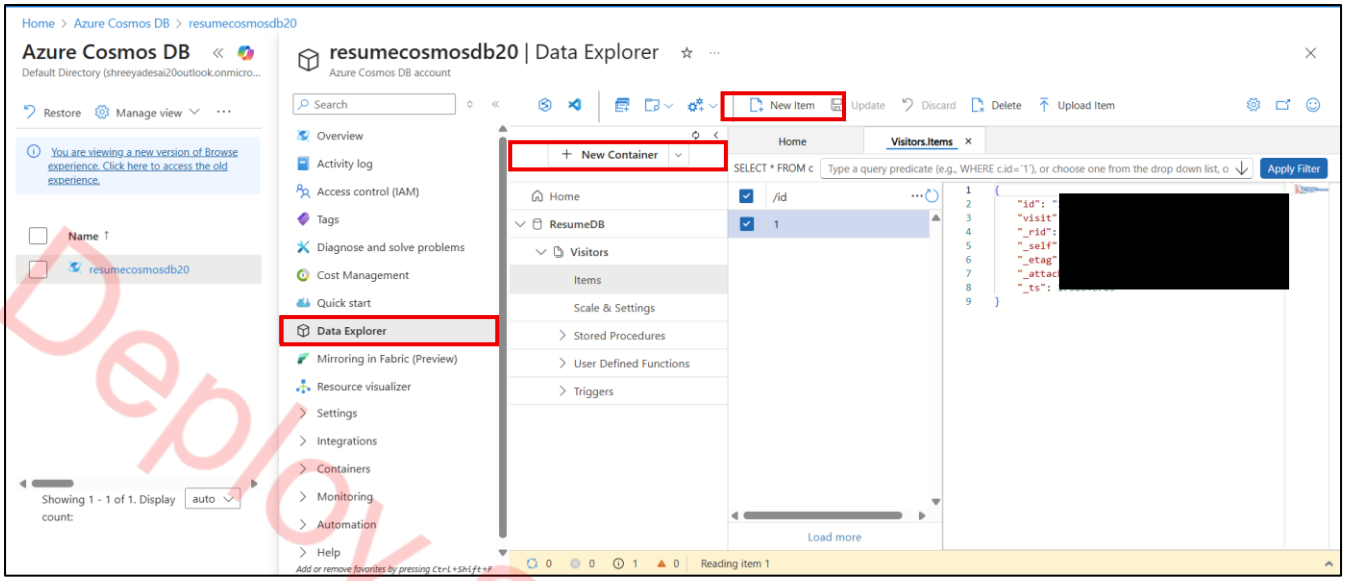
Step 4: In this select the "New Container" where a left side panel will be opened in which you need to set the Database id, Container id and Partition key.

Step 5: Upon creating the container, under the Container id name, you'll see "items", where upon selected select the "New Item" in which default JSON code will be changed to :

```
{
"id":"1",
"visit":"0"
}
```

In this, the id will be constant, and the visit will keep on increasing whenever someone visits the website.

Deploying a Static website on Microsoft Azure



In the image above, I have highlighted the Data Explorer through which the New Container and New item are created.

Deploying a Static website on Microsoft Azure

Create an Azure Function (Serverless API)

Azure Functions is a serverless computing service, meaning that the virtual machine (VM) or server is managed by the cloud provider. You just need to write the code, and Azure will run it as needed. This function will be the main brain of the operation. This will fetch the counter of the visitor, increment it and then update it in CosmosDB.

The function works as follow:

Fetches the visitor count.
Increments It.
Updates in the ComosDB.

Steps to create an Azure Function App:

Step 1: In the Azure portal, either search for Azure Function App or, from the left panel, select the Function app.

Step 2: Click on Create, and you'll see a variety of hosting plans, such as flex consumption, consumption, functions premium, app service & container app environment. Select the consumption plan, as it supports creating or editing the functions, editing the code and running the same. However, in the Flex Consumption, one can't create or edit functions in the portal; one must develop them locally and then deploy them in the functions app.

Step 3: Select your runtime stack (I opted for python), version and region that's near to you.

Step 4: Click on 'Review + Create' and then 'Create'.

The Function can be created in two ways :

1. Create through the Azure Portal.
2. Create locally and deploy

1. **Create through the azure portal.**

Step 1: Once the function app is created, go to its page.

Step 2: Select the Function and create.

Step 3: Opt for the Http trigger and give it a name. Select the "Functions" as the authorization level.

Step 4: Select create.

2. **Steps locally and deploy.**

The file structure is very important firstly.

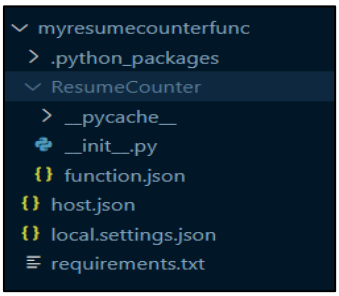


Figure 1: File Structure

Step 1:

The myresumecounterfunc is an Azure Function App that comprises three files: host.json, requirements.txt, and primarily local.settings.json. The local.settings.json handles the keys through which the code would be tested locally.

Step 2:

The ResumeCounter is the function where the main logic is developed. The **init**.py handles the logic of adding the visitors, in which the keys are required. To retrieve the endpoint and key for CosmosDB, navigate to the Azure portal and select the CosmosDB instance you have created. In the left panel of the CosmosDB interface, select the "Settings" section. Under this section, you will find the "Keys" option, which allows you to access both the endpoint and the keys.

Step 3:

The next step is to open the PowerShell as Administrator on your system and follow the below commands:

npm install -g azure-functions-core-tools@4 --unsafe-perm true

This command is mainly used to install the azure functions core tools, which are essential for developing and running the Azure function locally.

Step 4: Verify if the Azure function is installed.

func –version

Upon entering the command, you'll see the version of the func.

Step 5: Create a folder for the Azure Function App, if in case you haven't done this on the portal.

Initialize the Azure Function project (func init . –python).

Step 6: Create a new Http Trigger Function.

func new --name ResumeCounter --template "HTTP trigger" --authlevel "function"

Step 7: Write your code inside the files.

Step 8: Test locally

func start

where, you'll get a URL. Click on the URL and the browser will show a message "Visit:1". This indicates that you are going right but if incase you receive other message there would be an error in the code.

Step 9: Deploy using the command:

func azure functionapp publish myresumecounterfunc

Step 10:

Navigate to the Portal, select the created Function App. The left panel will have a section named "Setting", under which "Environment Variables" needs to be selected.

Upon selecting, you'll see two options namely App setting and connection string. Select the App setting as they are just key-value pairs. Add the CosmosDB endpoint and cosmos key along with their respective values.

Kindly note that the naming conventions should be proper such as Cosmos_Endpoint and Cosmos_Key.

NOTE: You can also create files normally using create folder and files. Then add the code to the associated file. Keep in mind that the file names are correct as any misspelled can cause you error.

Deploying a Static website on Microsoft Azure

Update the HTML file to call the API.

This is the final step where the HTML file connects to the Azure functions.

Step 1: Add a script to the code where an Azure_Function_URL is required.

Step 2: To get the URL:

Navigate to the Portal and select the Azure function that's created.

Under this, you'll see the recently created function; whereupon clicking, select the "Get Function URL".

In this you'll see three keys, out of which you must select the function key. The reason for option this key is its safe to expose it to the frontend and its scoped to a specific function. Never use the Master key and Host key as its not safe to expose.

Step 3: Once the file is updated, upload it to the $web (Storage account > Data Management > Static website) and then when any visitor visits the website the counter will be updated.