Microsoft Azure

Sl No.	Content
1.	Storing file on Azure storage using Drupal module
2.	Custom function to store any file on azure server

1. Aim: - To store Drupal File on Microsoft Azure Blob storage account.

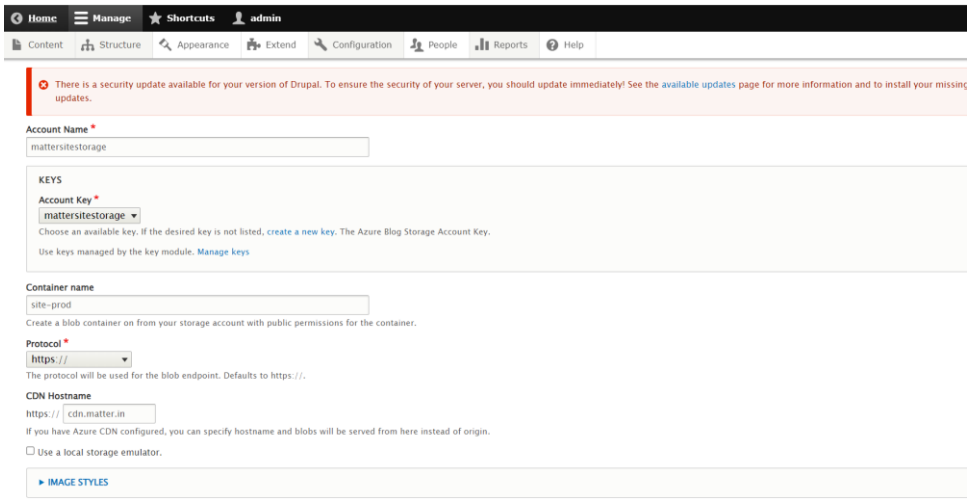
2. Requirement: -

- Account name is required.
- Account key is required.

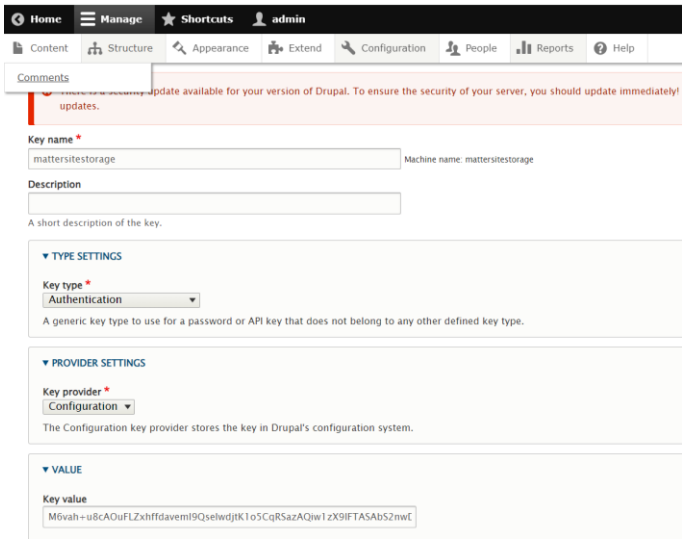
3. Install module named **Azure Blob Storage File System** through composer

composer require drupal/az_blob_fs
drush en az_blob_fs

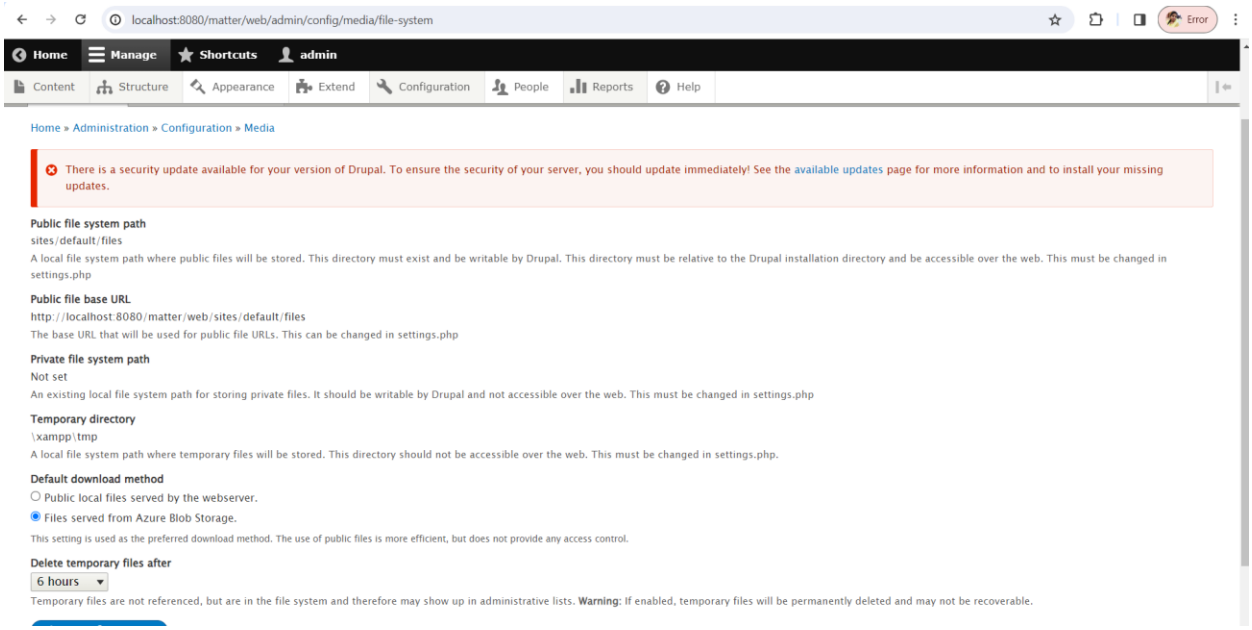
4. After enabling this module go to /admin/config/media/azure-blob-file-system and configure the system.



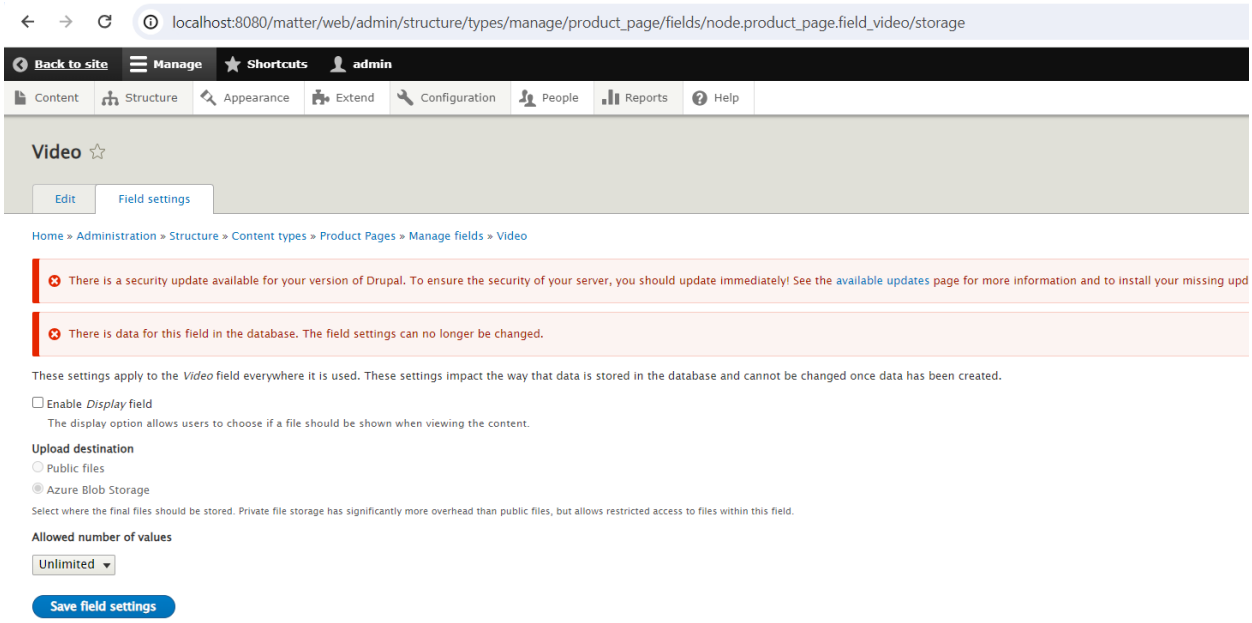
First you need to setup and account in which you will put azure storage account name and key refer below image to setup account and key.



5. Then go to **/admin/config/media/file-system** and set default file system to file served from azure.



Once the setup is done go to the content type and change the image and file upload destination from public files to azure blob. Now whenever you will upload image or file it will upload on Azure server.



Aim:- To create and upload JSON file on Azure server.

Code to update files on Azure Blob using custom theme.

Sometimes, we need to use custom code to update files on azure blob file system. In this tutorial we are creating a custom theme templet this will create a JSON file and save the file on Azure storage.

First we will create a JSON file and save the file on server or folder like “C:\path\to\file.png”. To save file on Azure storage we will call function **storageAddFile** . This function will take three parameter azure container name, path of local file name and filename where we store the file. Below code using env file for credentials.

```
/**
 * Convert the data into json format
 */
function convertToJson($filename, $data)
{
   $data = json_encode($data, true);
   //$path = 'C:/xampp/htdocs/matter/frontend/json_data'; //this line will save JSON on Local folder
   $path    = $_SERVER['DOCUMENT_ROOT'] . '/json_data'; //This line will save JSON on server
   $pattern = '/\/web/';
   $path    = preg_replace($pattern, '', $path);
   // To save JSON file in local directory.
   $file   = "$path/$filename";
   $handle = @fopen($file, "w+");
```

```
    if ($handle) {
        fputs($handle, $data);
        fclose($handle);
        header('Cache-Control: must-revalidate, post-check=0, pre-check=0');
        header('Pragma: no-cache');
        header("Content-Type: application/json");
        header("Content-Disposition: disposition-type=attachment; filename=\"$filename\"");
    }

    // To save JSON file in Azure blob.
    storageAddFile(getenv('CDN_FOLDER'), $file, "assets/json/$filename");
}
```

1. Install azure library via Composer

```
{
  "require": {
    "microsoft/azure-storage-blob": "*"
  }
}
```

2. include the autoloader script

```
require_once "vendor/autoload.php";
```

3. Include the namespaces you are going to use.

To create any Microsoft Azure service client you need to use the rest proxy classes, such as BlobRestProxy class:

```
use MicrosoftAzure\Storage\Blob\BlobRestProxy;
use MicrosoftAzure\Storage\Blob\Models\CreateBlockBlobOptions;
```

4. To instantiate the service client you will need a valid connection string in the format

```
AZURE_BLOB_CONNECTION="DefaultEndpointsProtocol=https;AccountName=mattersitestorage;AccountKey=9egiuwwsN6VCb3B/NLZSvraXK+HDpqAQH5cg3KNeouSsDpgj0y18O8HL4YFGD09xx8H3Qanuji68+ASt2vMQ2g=="
```

adds file to the storage. Usage: storageAddFile("myContainer", "C:\path\to\file.png", "filename-on-storage.png")

5. below source code is saving data saving file on the azure server.

```
$connectionString = getenv('AZURE_BLOB_CONNECTION');
define('CONNECTION', $connectionString);
```

```
/**
 * Adds file to the storage. Usage: storageAddFile("myContainer", "C:\path\to\file.png",
"filename-on-storage.png")
 */
function storageAddFile($containerName, $file, $fileName)
{
   $blobClient = BlobRestProxy::createBlobService(CONNECTION);
   $handle     = @fopen($file, "r");
   if ($handle) {
      $options = new CreateBlockBlobOptions();
      $mime    = null;
      try {
         // identify mime type
         $mimes = new \Mimey\MimeTypes;
         $mime  = $mimes->getMimeType(pathinfo($fileName, PATHINFO_EXTENSION));
         // set content type
         $options->setContentType($mime);
      } catch (Exception $e) {
         error_log("Failed to read mime from '" . $file . ": " . $e);
      }
      try {
         if ($mime) {
            $cacheTime = getCacheTimeByMimeType($mime);
            if ($cacheTime) {
               $options->setCacheControl("public, max-age=" . $cacheTime);
            }
         }
         $blobClient->createBlockBlob($containerName, $fileName, $handle, $options);
      } catch (Exception $e) {
         error_log("Failed to upload file '" . $file . "' to storage: " . $e);
      }
      if (is_resource($handle)) {
         @fclose($handle);
      }
      return true;
   } else {
      error_log("Failed to open file '" . $file . "' to upload to storage.");
      return false;
   }
}

/**
 * Get cache time by mime type
 */
```

```
function getCacheTimeByMimeType($mime)
{
   $mime  = strtolower($mime);
   $types = array(
      "application/json" => 604800, // 7 days
      "application/javascript" => 604800, // 7 days
      "application/xml" => 604800, // 7 days
      "application/xhtml+xml" => 604800, // 7 days
      "image/bmp" => 604800, // 7 days
      "image/gif" => 604800, // 7 days
      "image/jpeg" => 604800, // 7 days
      "image/png" => 604800, // 7 days
      "image/tiff" => 604800, // 7 days
      "image/svg+xml" => 604800, // 7 days
      "image/x-icon" => 604800, // 7 days
      "text/plain" => 604800, // 7 days
      "text/html" => 604800, // 7 days
      "text/css" => 604800, // 7 days
      "text/richtext" => 604800, // 7 days
      "text/xml" => 604800, // 7 days
   );
   // return value
   if (array_key_exists($mime, $types)) {
      return $types[$mime];
   }
   return false;
}

/**
 * Removes file from the storage. Usage: storageAddFile("myContainer", "filename-on-
storage.png")
 */
function storageRemoveFile($containerName, $fileName)
{
   // Create blob client.
   $blobClient = BlobRestProxy::createBlobService(CONNECTION);
   try {
      $blobClient->deleteBlob($containerName, $fileName);
   } catch (Exception $e) {
      error_log("Failed to delete file '" . $fileName . "' from storage");
   }
   return true;
}
```