

## 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: -

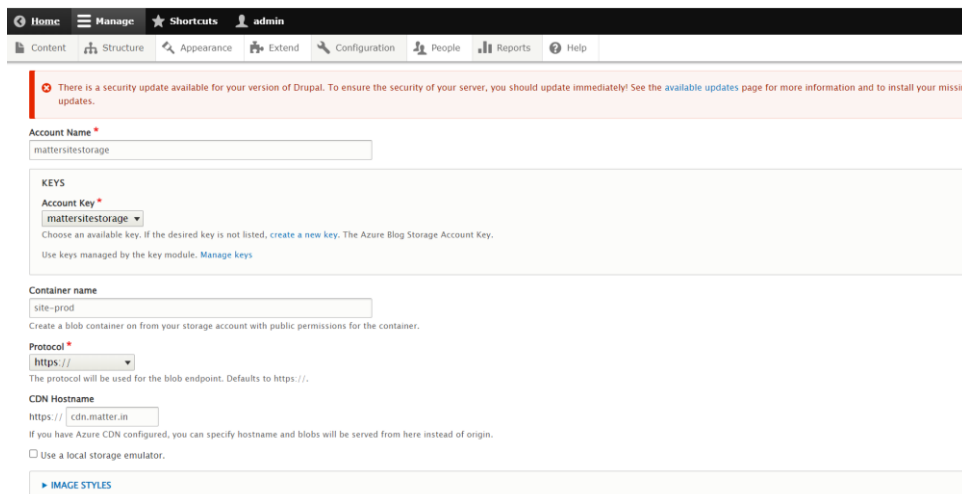
- Account name is required.
- Account key is required.

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.



The screenshot shows the Drupal configuration interface for the 'Azure Blob Storage File System' module. At the top, there's a navigation bar with 'Home', 'Manage', 'Shortcuts', and 'admin'. Below it, a toolbar contains icons for 'Content', 'Structure', 'Appearance', 'Extend', 'Configuration', 'People', 'Reports', and 'Help'. A red message banner at the top states: 'There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the available updates page for more information and to install your missing updates.' The configuration form includes several fields: 'Account Name' (text input with value 'mattersitestorage'), 'Account Key' (dropdown menu with value 'mattersitestorage'), 'Container name' (text input with value 'site-prod'), 'Protocol' (dropdown menu with value 'https://'), and 'CDN Hostname' (text input with value 'https://cdn.matter.in'). A checkbox for 'Use a local storage emulator' is present and unchecked. A link 'Manage keys' is visible below the 'Account Key' dropdown. At the bottom, there's a link 'IMAGE STYLES'.

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.

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Comments

There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

Key name \*

mattersitestorage Machine name: mattersitestorage

Description

A short description of the key.

▼ TYPE SETTINGS

Key type \*

Authentication

A generic key type to use for a password or API key that does not belong to any other defined key type.

▼ PROVIDER SETTINGS

Key provider \*

Configuration

The Configuration key provider stores the key in Drupal's configuration system.

▼ VALUE

Key value

M0vah+u8cAOuFLZxhffdavem19QseIwdjK1o5CqRSazAQw1zX9IFTASAbS2mwL

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

localhost:8080/matter/web/admin/config/media/file-system

Home Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

Home » Administration » Configuration » Media

There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

Public file system path

sites/default/files

A local file system path where public files will be stored. This directory must exist and be writable by Drupal. This directory must be relative to the Drupal installation directory and be accessible over the web. This must be changed in settings.php

Public file base URL

http://localhost:8080/matter/web/sites/default/files

The base URL that will be used for public file URLs. This can be changed in settings.php

Private file system path

Not set

An existing local file system path for storing private files. It should be writable by Drupal and not accessible over the web. This must be changed in settings.php

Temporary directory

\xampp\tmp

A local file system path where temporary files will be stored. This directory should not be accessible over the web. This must be changed in settings.php.

Default download method

☐ Public local files served by the webserver.

☒ Files served from Azure Blob Storage.

This setting is used as the preferred download method. The use of public files is more efficient, but does not provide any access control.

Delete temporary files after

6 hours

Temporary files are not referenced, but are in the file system and therefore may show up in administrative lists. **Warning:** If enabled, temporary files will be permanently deleted and may not be recoverable.

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.

localhost:8080/matter/web/admin/structure/types/manage/product\_page/fields/node.product\_page.field\_video/storage

Back to site Manage Shortcuts admin

Content Structure Appearance Extend Configuration People Reports Help

## Video ☆

Edit Field settings

Home » Administration » Structure » Content types » Product Pages » Manage fields » Video

✖ There is a security update available for your version of Drupal. To ensure the security of your server, you should update immediately! See the [available updates](#) page for more information and to install your missing updates.

✖ There is data for this field in the database. The field settings can no longer be changed.

These settings apply to the *Video* field everywhere it is used. These settings impact the way that data is stored in the database and cannot be changed once data has been created.

☐ Enable *Display* field  
The display option allows users to choose if a file should be shown when viewing the content.

**Upload destination**

☐ Public files

☒ Azure Blob Storage

Select where the final files should be stored. Private file storage has significantly more overhead than public files, but allows restricted access to files within this field.

**Allowed number of values**

Unlimited ▼

Save field settings

## 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+AS2vMQ2g=="
```

## 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;
}

```