In some situations, it may be beneficial to store files of an onpremise website on a **Microsoft Azure Blob Storage** account rather than on a local disk.

- For example, if your server has a limited storage capacity and you need to save large amounts of file data. Using Microsoft Azure may be more convenient than upgrading your server, especially if the increased requirements are only temporary.
- The same applies when running Kentico in **Azure Web Apps**, where use of blob storage is optional (unlike with Cloud Services).

There are two approaches for connecting an Azure storage to your project:

- Store only media files on Azure storage **(recommended)**
  - We recommend that you map only media library folders to the Azure storage. This allows you to store large media files on a shared storage and optionally use an Azure CDN for this data, while avoiding issues related to local smart search indexes and Web analytics functionality.
- Map the whole file system to Azure storage
  - We do not recommend mapping the whole file system to the Azure storage because of possible problems with local smart search indexes and Web analytics functionality. These features use data that is not suitable for shared file systems.

Mapping media files or the file system or storage accounts enforcing HTTPS connections

Storage accounts can be configured to communicate exclusively over HTTPS via the Security transfer required setting available in the Azure portal. In such cases, additional configuration of the environment is required. See Mapping files to storage accounts enforcing HTTPS connections for details.

File name case

Unlike standard Windows file systems, the Microsoft Azure Blob storage is case-sensitive. To ensure consistent behavior, Kentico automatically converts all file and folder names to lower case when processing files on Azure storage.

Configuring Kentico to store only media files on Azure storage

To map media library folders to Azure storage, use the API and the built-in Azure storage provider:

1. Add the following keys to the *appSettings* section of your project's **web.config** file. Specify the storage account name and primary access key:

```
<add key="CMSAzureAccountName" value="StorageAccountName" />
<add key="CMSAzureSharedKey" value="PrimaryAccessKey" />
```

To locate the values of these keys on Microsoft Azure:

a. Open the Azure Management Portal in a browser and sign in.
b. Open **Storage accounts**.
c. Select your storage.
d. Switch to the **Access keys** tab.
   - Use the **Storage account name** and one of the provided access key values.

Important

Do NOT set the **CMSExternalStorageName** key in your web.config (remove the key if it is present). Setting the key's value to *azure* maps the project's entire file system to the Azure storage, which is not recommended. See the Configuring Kentico to map its file system to Azure storage section for more information about this scenario.

2. Open the Kentico web project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
3. Create a custom module class.
   - Either add the class into a custom project within the Kentico solution (recommended) or directly into the Kentico web project (into the **CMSApp** project for *web application* installations, into the **App_Code** folder for *web site* installations).

For basic execution of initialization code, you only need to register a "code-only" module through the API. You do NOT need to create a new module within the **Modules** application in the Kentico administration interface.

4. Override the module's **OnInit** method and perform the following:
   - Create a new instance of the Azure storage provider.
   - (Optional) Specify the target container using the **CustomRootPath** property of the provider.
   - (Optional) You can specify whether you want the container to be publicly accessible using the **PublicExternalFolderObject** property of the provider. *True* means the container is publicly accessible.
   - Map a directory to the provider. This is the directory that you want to store in the container.

```
using CMS;
using CMS.Base;
using CMS.DataEngine;
using CMS.IO;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(CustomInitializationModule))]

public class CustomInitializationModule : Module
{
        // Module class constructor, the system registers the module under the
name "CustomInit"
        public CustomInitializationModule()
                : base("CustomInit")
        {
        }

        // Contains initialization code that is executed when the application
starts
        protected override void OnInit()
        {
                base.OnInit();

                // Creates a new StorageProvider instance for Azure
                var mediaProvider = StorageProvider.CreateAzureStorageProvider();

                // Specifies the target container
                mediaProvider.CustomRootPath = "mymediacontainer";

                // Makes the container publicly accessible
                mediaProvider.PublicExternalFolderObject = true;

                // Maps a directory to the provider
                StorageHelper.MapStoragePath("~/MySite/Media/", mediaProvider);
        }
}
```

5. **Save** the file. If your project is installed in the web application format, rebuild the solution.

The system now stores files from the *~/MySite/Media/* folder in the *mymediacontainer* on the Azure storage. See the Media library notes section on this page for additional information about media libraries when using Azure storage.

Mapping media folders on instances with multiple sites

By default, each site has its own separate media folder in the project's file system: *~/<site code name>/media*

To map all media folders to Azure storage on Kentico instances with multiple sites, we recommend using a shared root folder for media libraries:

1. Open the **Settings** application in Kentico.
2. Navigate to the **Content -> Media** settings category.
3. Set the **Media libraries folder** to a custom folder (globally), for example: ~/SharedMedia
4. Enable the **Use site-specific subfolders for custom media libraries folder** setting to ensure that individual library folders are separated by site.
5. Click **Save**.
6. Move any existing media library files to the new location.

You can then map the shared media folder to your Azure storage (using the API described above).

If you wish to use the default media folder locations, you need to map each site's media folder in the code – for each site, create and configure a separate *StorageProvider* instance and call the *StorageHelper.MapStoragePath* method for the corresponding media folder.

Using Azure storage for multiple projects

We do not recommend using a single shared storage for multiple projects (for example production and testing instances), because the projects would overwrite each others' files.

However, you can use the **mediaProvider.CustomRootPath** property (as described on this page) to map each project to a different container. This way, each project has its own section of the Azure storage and overwriting of files does not occur.

Configuring Kentico to map its file system to Azure storage

Warning: Mapping the Kentico project's entire file system to Azure storage is only recommended when running on Azure Cloud Services. In other environments, this type of setup may introduce issues with locally stored search indexes and Web analytics functionality and we recommend that you only map media library folders.

If you do wish to map the entire file system, you can set up additional mappings back to the local file system for the search index and web analytics folders – see Mapping the Web analytics storage folder to the server file system for details. Also see Troubleshooting local search indexes on Azure for more information.

1. Add the following key into the *appSettings* section of your project's **web.config** file.

```
<add key="CMSExternalStorageName" value="azure" />
```

   This key maps the entire file system of your application to the Azure storage.

2. Add the following keys to specify the storage account name and primary access key:

```
<add key="CMSAzureAccountName" value="StorageAccountName" />
<add key="CMSAzureSharedKey" value="PrimaryAccessKey" />
```

   To locate the values of these keys on Microsoft Azure:

   a. Open the Azure Management Portal in a browser and sign in.

b. Open **Storage accounts**.
c. Select your storage.
d. Click **Manage access keys** on the bottom panel.
    - Use the **Storage account name** and one of the provided access key values.

Additional website settings

When configuring this type of storage, keep in mind that the website itself must be configured to store files in the file system rather than in the database only. In **Settings -> System -> Files** enable the **Store files in file system** option.

It is also recommended to enable **Redirect files to disk** in **Settings -> System -> Performance**. This means that files will be requested from the Azure Storage account rather than from the database (if possible).

Also see the Media library notes on this page for additional information about specifics of media libraries when using Azure storage.

Mapping files to storage accounts enforcing HTTPS connections

Some external storage accounts may enforce HTTPS communication via the Security transfer required setting configurable in the Azure portal. When mapping the file system or media library files to such accounts, you must include the CMSAzureBlobEndPoint configuration key in the application's **web.config** file. The key's value needs to contain the full endpoint URL of the external storage account and explicitly use the *https* protocol:

```
<appSettings>
        ...
        <add key="CMSAzureBlobEndPoint" value="https://<StorageAccountName>.blob.core.
windows.net" />
</appSettings>
```

Media library notes

Using the Azure Blob storage as an external storage for your project has some specific effects on media libraries.

Storing too many files in one media library folder

Note that storing a large number of media files in a single folder can significantly affect the performance of user interface when editing files in the Media library application. The performance of the website, however, is not affected. See Media library limitations when storing files in an external storage for details.

Using permanent links when CMSAzurePublicContainer key is set to true

If all following conditions are true:

- you use the Azure Blob storage as an external storage
- and you set the CMSAzurePublicContainer key to *true*
- and you want to use content staging

then **use Permanent links** when linking to media files in media libraries.

The reason is that when using the CMSAzurePublicContainer key, direct file links contain the name of the storage in the URL. These links are not updated when staging files to another server, which is typically connected to a different storage. As a result, staged links incorrectly target files from the source server's storage, or may become broken if the files are not exactly mirrored on the staging servers. Permanent links do not contain the name of the storage, and automatically target the correct storage for each server.

Optional web.config settings for Azure storage

Key	Description	Sample value
CMSAzureTempPath	The system uses the specified folder to store temporary files on a local disk, for example when transferring large files to or from the storage account.	`<add key="CMSAzureTempPath" value="C:\AzureTemp" />`
CMSAzureCachePath	Specifies a folder on a local disk where files requested from the storage account are cached. This helps minimize the amount of blob storage operations, which saves time and resources.	`<add key="CMSAzureCachePath" value="C:\AzureCache" />`

CMSAzureBlobEndPoint	Sets the endpoint used for the connection to the blob service of the specified storage account. If you wish to use the default endpoint, remove the setting completely from the appropriate files.	`<add key="CMSAzureBlobEndPoint" value="http://127.0.0.1:10000/devstoreaccount1" />`
CMSAzurePublicContainer	Indicates if the blob container used to store the application's file system is public. If true, it will be possible to access files directly through the URL of the appropriate blob service, for example: ***http://<StorageAccountName>.blob.core.windows.net/cmsroot/corporatesite/media/imagelibrary/logo.png*** When you set this key to true, please see the Media library notes section.	`<add key="CMSAzurePublicContainer" value="true" />`

CMSAzureCDNEndpoint	URL of the HTTP endpoint of a Azure Blob storage CDN. **Note:** If you set the CMSAzureCDNEndpoint key, you also need to set the blob storage container to public - **<add key="CMSAzurePublicContainer" value="true" />**.	`<add key="CMSAzureCDNEndpoint" value="kentico123.vo.msecnd.net" />`
CMSDownloadBlobTimeout	Specifies the timeout interval in minutes for importing files from Azure Blob storage into Kentico. The default value is *1.5* minutes. Increase the interval if you encounter problems when importing large (about 2GB) files.	`<add key="CMSDownloadBlobTimeout" value="50" />`