This page describes how to run multiple websites in separate sub-folders on a single domain. In this scenario, you do not need to obtain new domains for each site.

Example - Installing two sites onto a single domain

1. Install a Kentico web project to the following folder: **C:\inetpub\wwwroot\kenticofolder**.
   - In the installer, choose **Custom installation** and **Built-in web server in Visual Studio** (does not create a virtual directory).
2. Open your **Internet Information Services (IIS) Manager** console (*Start -> Control Panel -> Administrative tools -> Internet Information Services (IIS) Manager*).
3. Create a new virtual directory named **kenticoweb**.
   - The name of the virtual directory must be different than the folder where you installed the web project.
   - Set the **Physical path** to a non-website folder in the root of a local disk. Ideally, create an empty folder for this purpose, for example: *c:\empty*
4. Create two IIS applications under **kenticoweb** named **web1** and **web2**.
   - Set the **Physical path** of both applications to the web site folder of the installed project (**C:\inetpub\wwwroot\kenticofolder\CMS** in this example).
   - Set the application pool according to the type that you specified in the installation of the Kentico web project.
5. Open your browser and type in either **http://localhost/kenticoweb/web1** or **http://localhost/kenticoweb/web2**.
6. Sign in to the Kentico administration interface and open the **Sites** application.
7. Install both sites. Set the **Site domain name** fields:
   - Website 1: **localhost/kenticoweb/web1**
   - Website 2: **localhost/kenticoweb/web2**

Now when you go to **http://localhost/kenticoweb/web1**, you will see website 1. If you go to **http://localhost/kenticoweb/web2**, you will see website 2.

Synchronizing global data for the sites

To ensure the synchronization of settings and global objects between the two sites, you need to set up a Web farm environment:

1. Sign in to the Kentico administration interface on one of the sites.
2. Open the **Settings** application.
3. Select the **Versioning & Synchronization -> Web farm** category.
4. Select **Automatic** as the **Web farm mode**.
5. Click **Save**.
6. Add the following key to the *<appSettings>* section of the **web.config** file in the installation directory shared by both websites:

```
<add key="CMSWebFarmSynchronizeFiles" value="false" />
```

The system automatically creates web farm servers for the applications and performs synchronization. The *CMSWebFarmSynchronizeFiles* web.config key disables synchronization of files, which is not needed since the applications already use the same physical folder.

Avoiding file system conflicts

To prevent problems with file conflicts for web analytics log files and locally stored search indexes, you need to configure the application to use a shared file system:

1. Open the Kentico web project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a custom module class.
   - We recommend adding the class into a custom *Class library* project within the Kentico solution.
3. Override the module's **OnInit** method and create a custom storage provider that handles the application's entire file system as shared storage:

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

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

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

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

                // Maps the application's entire file system to the same
location, but with the "shared storage" flag enabled
                var sharedFileSystemProvider = StorageProvider.
CreateFileSystemStorageProvider(isSharedStorage: true);
        sharedFileSystemProvider.CustomRootPath = SystemContext.
WebApplicationPhysicalPath;
        StorageHelper.MapStoragePath("~/", sharedFileSystemProvider);
        }
}
```

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

With a shared file system, the system automatically avoids file name collisions for smart search indexes and web analytics logs.

Your websites should now work without issues.