Milk development and deployment

Setup

Requirements and tools

Operating system: Windows (10)

Source control: Github.

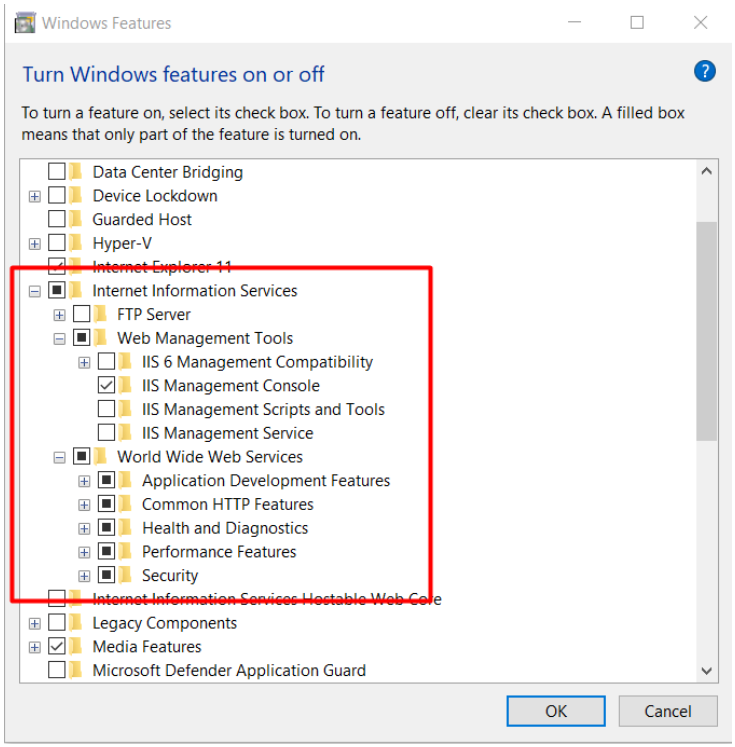
Tools: Visual Studio (2017), Visual Studio Code, git bash, RDP (Remote Desktop Manager), IIS Manager

Languages and technologies: ASP.NET Web Forms (C#), javascript, jQuery, HTML/CSS, nodejs, gulp, SQL

CMS: Kentico

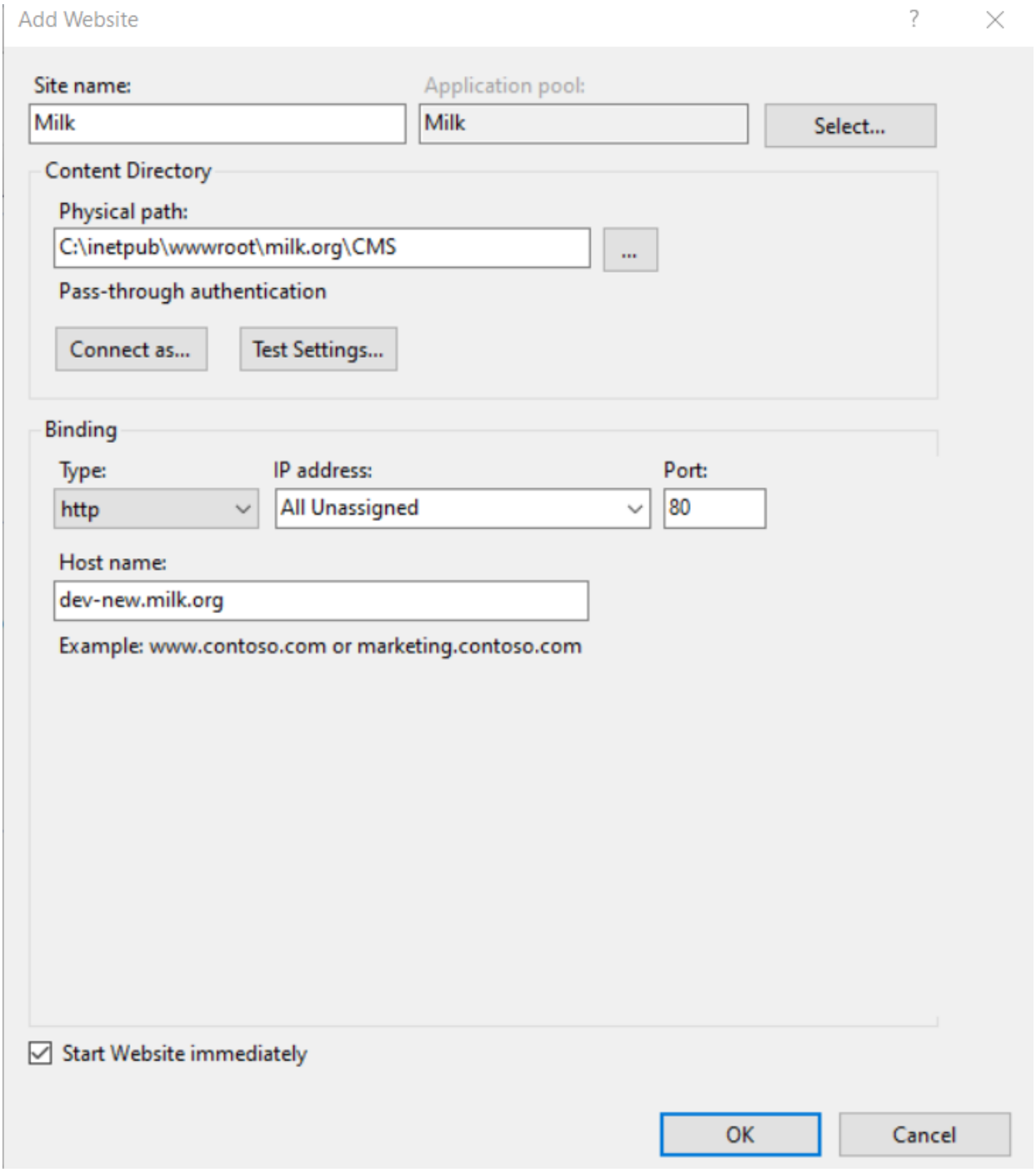
Setup

Enable IIS Services on your Windows computer. Go to *Control Panel -> Programs -> Programs and Features -> Turn Windows features on or off* and make sure IIS Services are turned on as shown on the picture below. Restart your computer if needed.

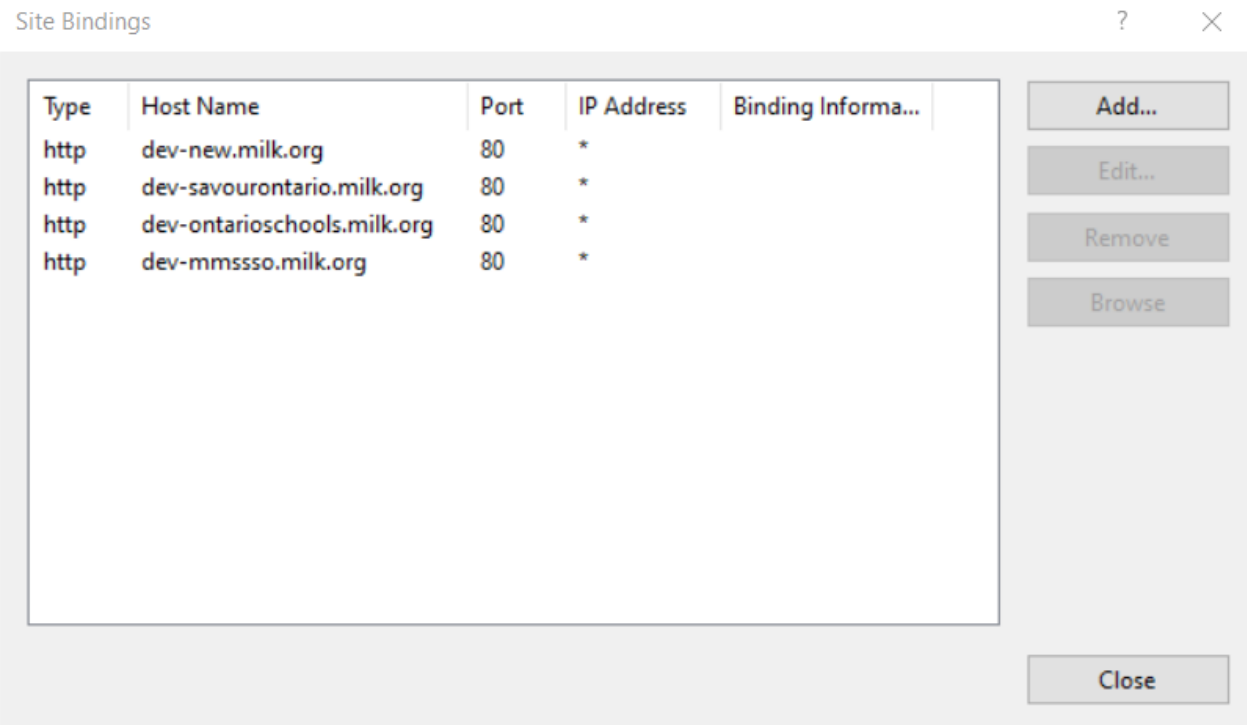


Clone the repository from github. Open git bash in, preferably, *C:\inetpub\wwwroot* folder and clone your repository by typing `git clone https://github.com/mightyrockca/milk.org.git`. You will have to log into github and be assigned to the project in order to be able to clone the repository.

After the project is downloaded, set up local hosting in IIS. Open *Internet Information Services (IIS) Manager* application on your Windows computer and create a website from the cloned repository. Right click on *Sites,* choose *Add website* and configure it as shown below.



Once the website is created, right click on a newly created website and click *Edit Bindings…* and add other bindings like shown in the image.



Navigate to *C:\Windows\System32\drivers\etc* and edit *hosts* file with admin privileges. Add bindings like in the picture.

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#   127.0.0.1       localhost
#   ::1             localhost
127.0.0.1 dev-new.milk.org
127.0.0.1 dev-savourontario.milk.org
127.0.0.1 dev-ontarioschools.milk.org
127.0.0.1 dev-mmssso.milk.org
```

Change the connection string in your *web.config* file.

```
<add name="CMSConnectionString" connectionString="Data Source=milkorg-db-
dev.c7lxjrdhazjl.ca-central-
1.rds.amazonaws.com;Initial Catalog=milkorg;Integrated Security=False;Persist Sec
urity Info=False;User ID=admin;Password=vifMc6rXhF3QX2VT0led;Connect Timeout=120;
Encrypt=False;Current Language=English;" />
```

Development

For the Milk project we use Jira as a Product Management Software. Development process is pretty much straightforward.

New tickets have a *To Do* status. When developer works on a ticket it should be in *In Progress* status, and once it's finished it should be in *Ready for Deployment* status.

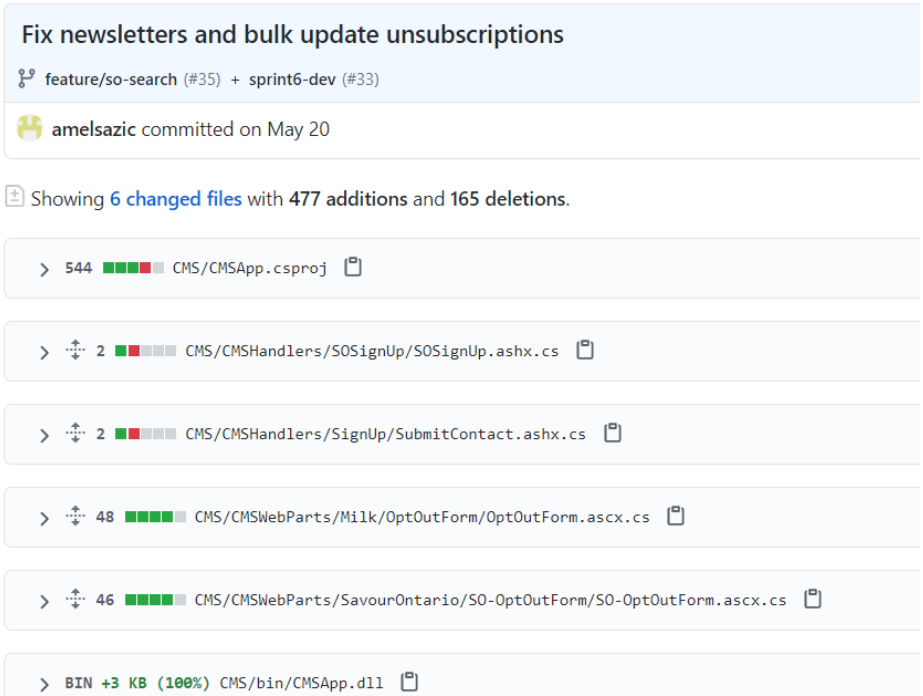
We use Visual Studio for any backend development, meaning pages, web parts etc. Anything in C#. Building the website through Visual Studio creates new CMSApp.dll file which should be included in the commit as it contains compiled code. This is required if any of *.cs* files were changed.

Frontend changes like CSS and js can also be done in Visual Studio, but my preference is Visual Studio Code as it is pretty easy to set up for these type of changes. For some of these we will need nodejs and gulp installed. This is required for bundling *scss* and *js* files in UI projects such as *UIDFOIndustryResources, UIIndustryNews, UISavourOntario.* Development here is done through individual css and js files which are later bundled and published. Instructions for this can be found in *Readme* files for each of these projects.

Source control is done through github, and again my preference for source control in general is Visual Studio Code with its built-in terminal since it has an easy to use interface for this.

The main development branch, currently, is *sprint6-dev* branch.

While working on changes, it helps keeping track of changed files to ease up the deployment process. The changed files also can be found on github by going through commits.



Any non code related changes are done through CMS. Kentico knowledge required.

Keeping track of changed objects or content is prefferable as it will ease up the deployment process.

Database changes are usually not neccessarry as this is mainly done through CMS interface.

Deployment

The tickets that are in *Ready for Deployment* status need to be moved to the server.

We have one development server. Developer sets up local website and uses a shared database from the server.

Currently there are three servers to help with the smooth transition of new releases.

QA server is usually the first one that the new changes are deployed to and is used for initial testing of the changes.

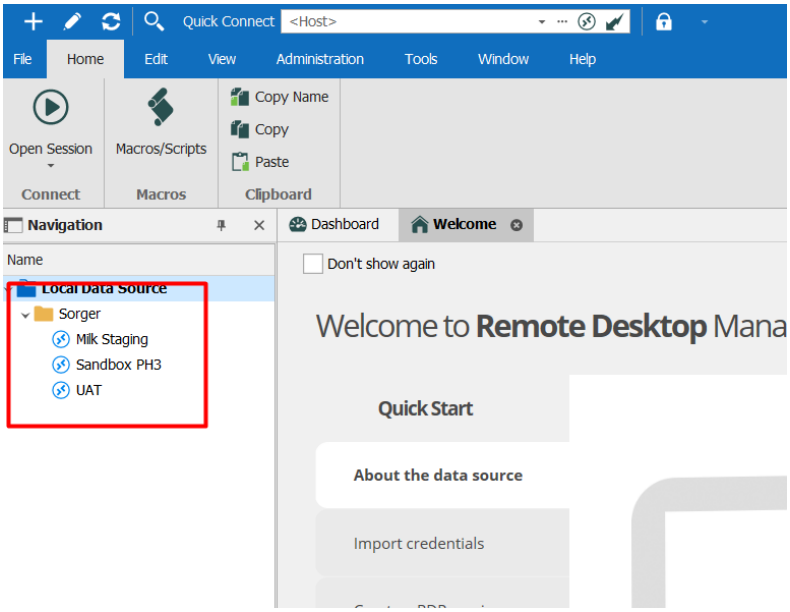
UAT server is the one we typically use for training as well as preparing content for production deployment.

Production server is the one that is live and should be as bug free as possible with latest content available for end users.

Code deployment

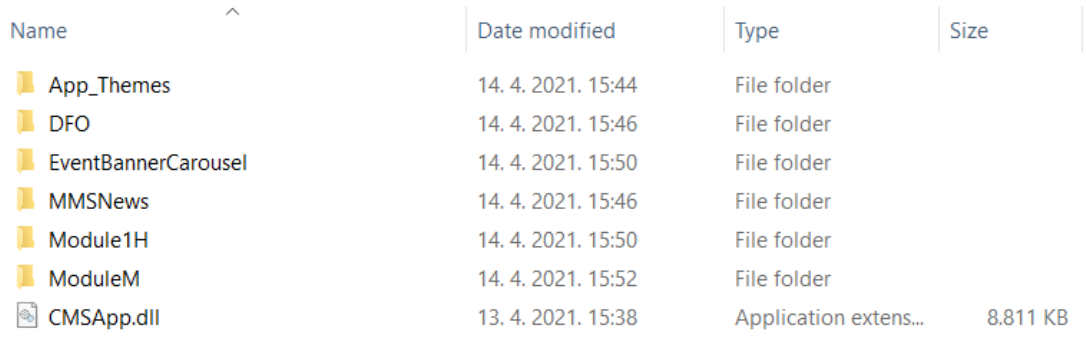
Once the ticket is in *Ready for Deployment* status, it's time to deploy the changes to the QA server in order for them to be tested by QA Engineers.

When there were code changes, we need to manually deploy the files that were changed. Log into remote QA server via RDP protocol. For this we can use Windows built-in *Remote Desktop Connection* application, enter username and password and we are in. We can also use *Remote Desktop Manager* tool (my preference), as we can set up multiple servers and connect to them in one click.



Once we are connected to the server it's time to deploy the files. The way we usually do this is create a new folder as a deployment package and copy all the changed files from the local website into the folder. It helps keeping the same file structure as the original one as it will be easier to replace them on server.

First copy all the files and/or folders that contain the changes that need to be deployed and make a folder for it. For exampe:



Name	Date modified	Type	Size
App_Themes	14. 4. 2021. 15:44	File folder	
DFO	14. 4. 2021. 15:46	File folder	
EventBannerCarousel	14. 4. 2021. 15:50	File folder	
MMSNews	14. 4. 2021. 15:46	File folder	
Module1H	14. 4. 2021. 15:50	File folder	
ModuleM	14. 4. 2021. 15:52	File folder	
CMSApp.dll	13. 4. 2021. 15:38	Application extens...	8.811 KB

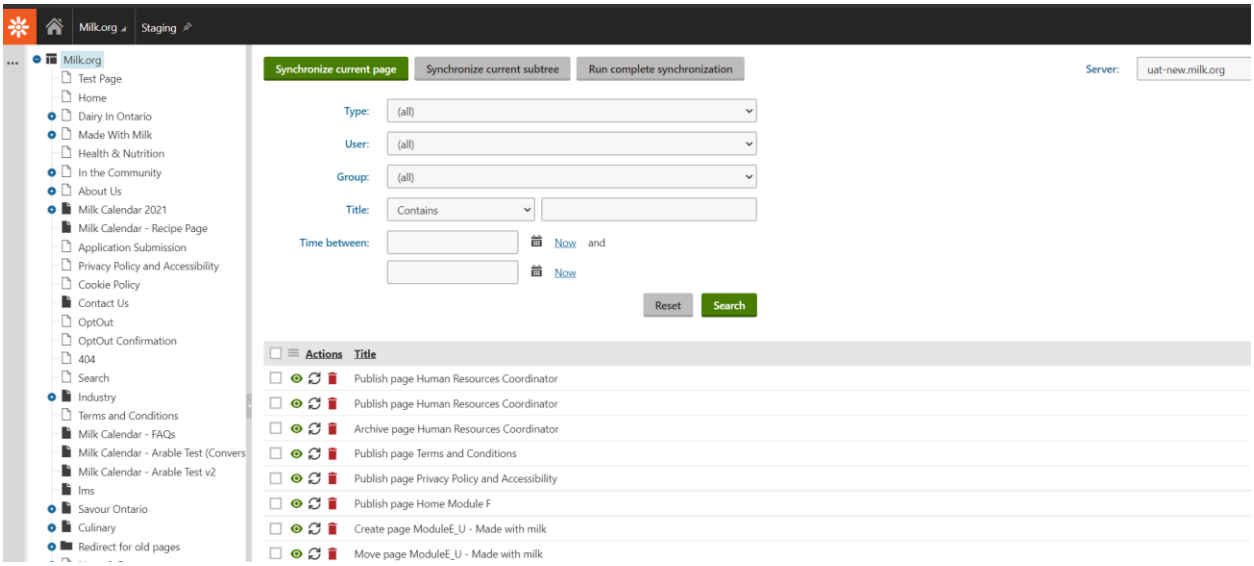
Zip the folder and move it to the server. Once it is on the server, unzip the folder. It is prefferable to make a copy of the same files that are changed on the server as a backup. Once all this is done, simply copy the files into the appropriate locations on the server. The file structure must remain the same. The easiest way to find the currently active website folder is through IIS. Open IIS Manager, right click on the website and select *Explore.* This will open up File Explorer in the folder that we are deploying to.

Once all the files are replaced it's time to deploy CMS changes.

Content deployment

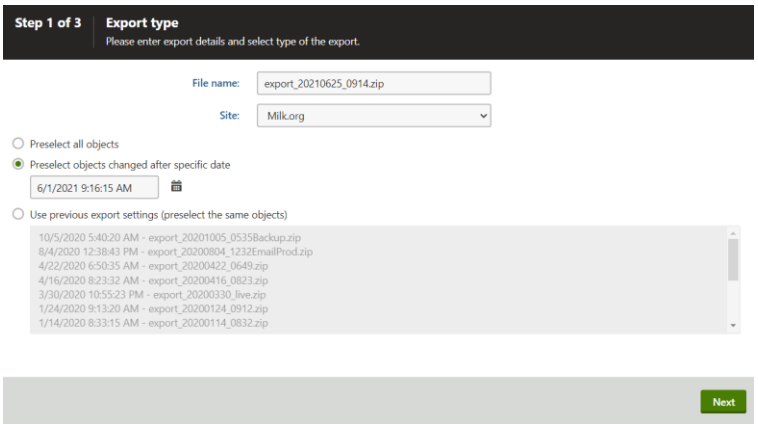
Content deployment means two things. Deploying content in terms of pages, and deploying kentico objects that are part of development.

Pages are most easily deployed through Kentico's staging feature. Select the server you want to deploy to in the Server dropdown. To deploy pages simply select pages, or page trees you want to sync to the server and click *Synchronize current page* or *Synchronize current subtree.*

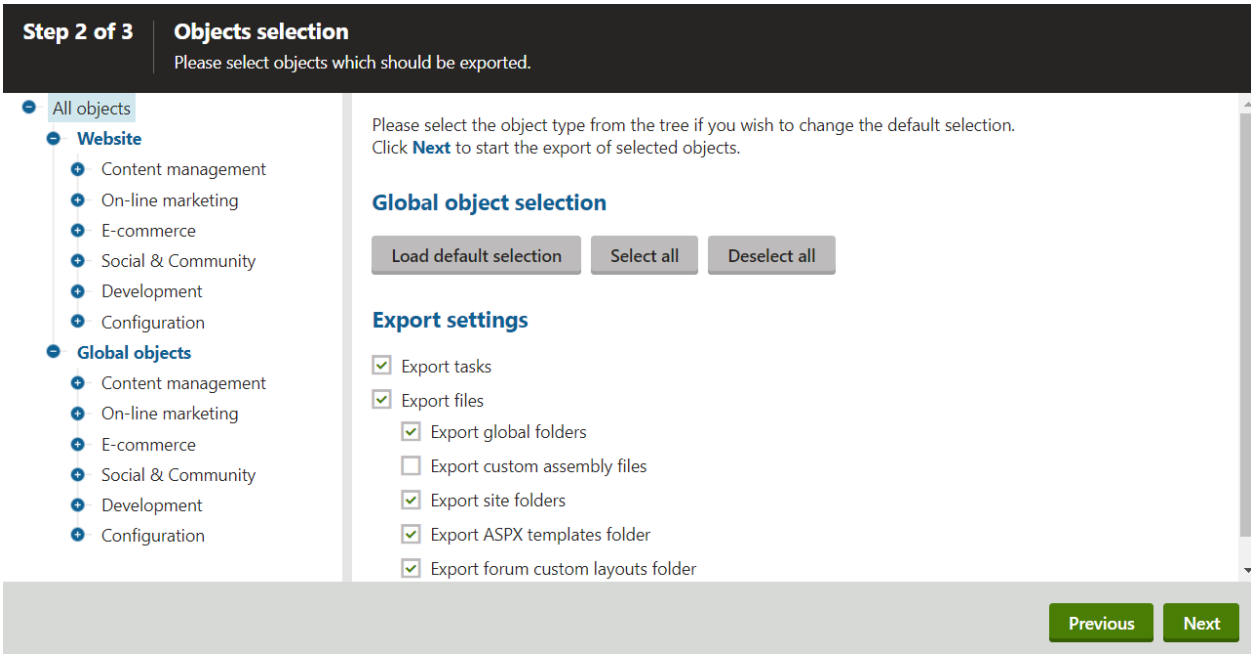


For other Kentico objects we typically use *Import/Export* feature in *Sites* application. These object can also be synced through staging feature if it's configured to track the changes.

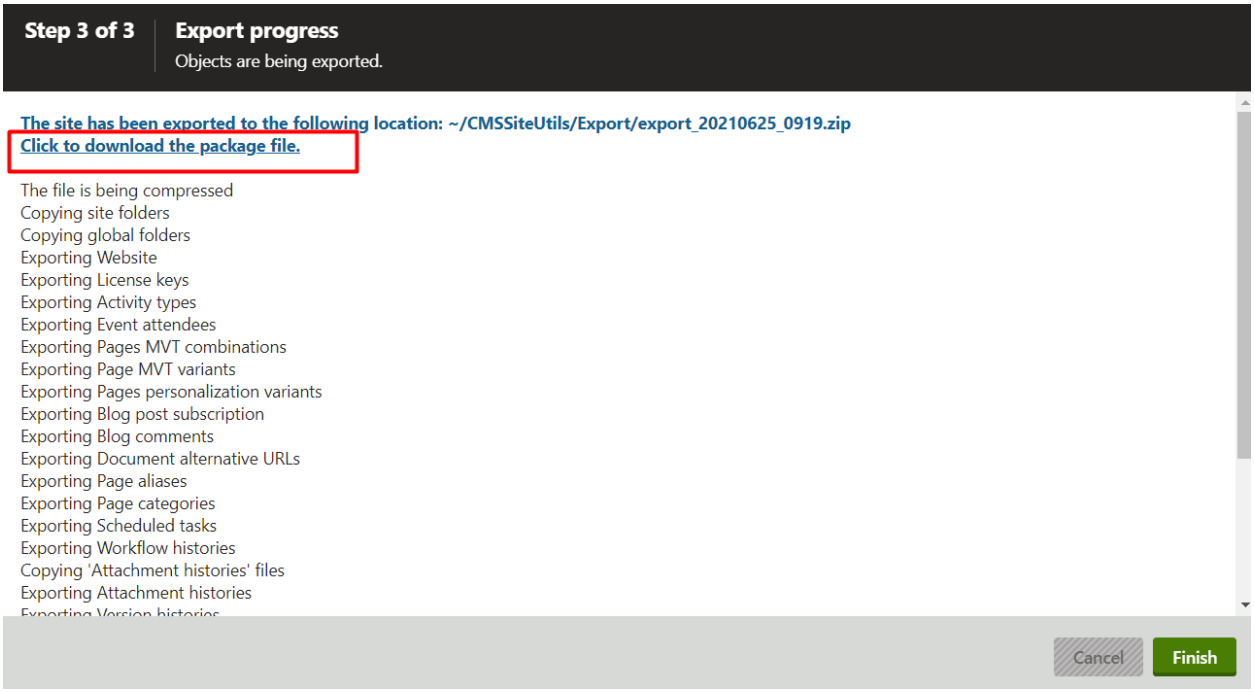
To export objects go to *Sites* and click *Export*. File name is generated, although it can be changed. Select the site you want to export objects for, and select one of the three selection options depending on what you need. I usually choose the second one to select only objects changed after some date.



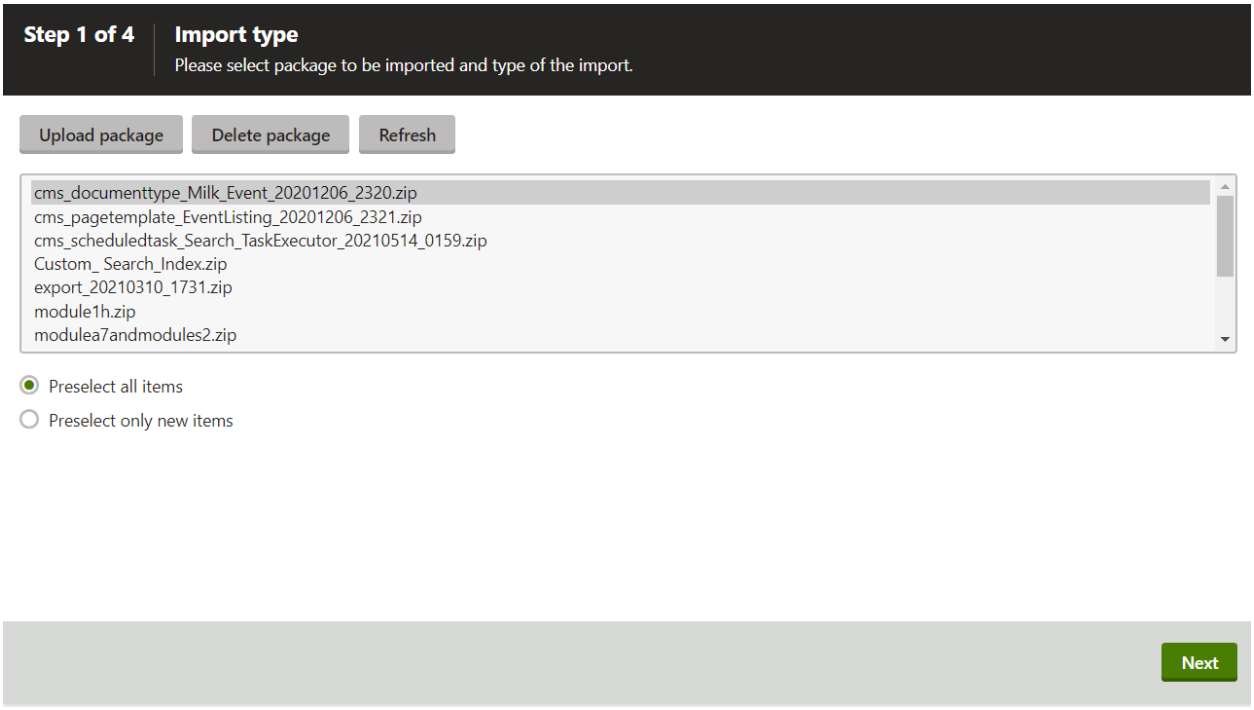
On *Step 2* check if everything you need is selected and click *Next*.



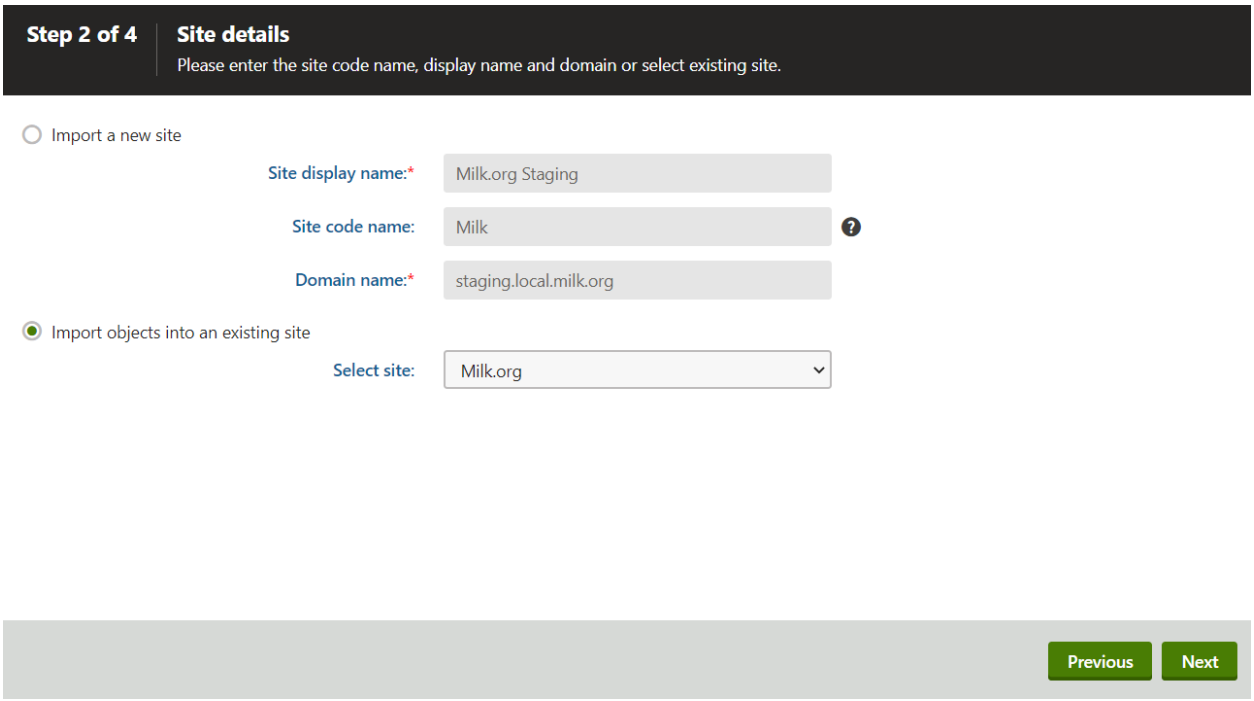
Once the export file is generated click *Click to download the package file*, download the file and click *Finish*.



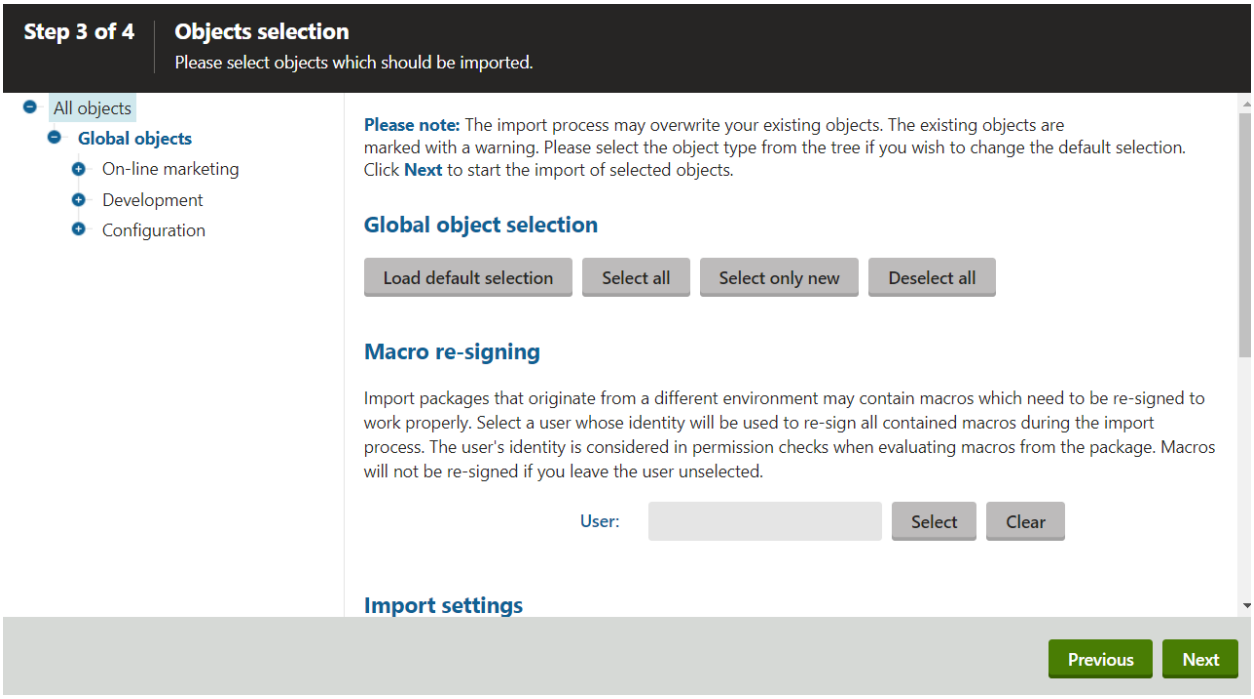
The downloaded file is ready to be imported on the target server. Go to *Sites* application and click *Import site or objects.* Click *Upload package* and select the file we created previously. Preselect all items as, during export, we only selected the ones that were changed.



On Step 2 select *Import object into an existing site* and select the site you want to import objects to.



On Step 3 check if everything you need is there and selected and click *Next*.



With this we are done with the deployment. On the *System* application click *Clear cache,* and then *Restart application* to make sure the sites are running with the latest changes.