# Umbraco CMS Build

## Are you sure?

In order to use Umbraco as a CMS and build your website with it, you should not build it yourself. If you're reading this then you're trying to contribute to Umbraco or you're debugging a complex issue.

* Are you about to create a pull request for Umbraco?
* Are you trying to get to the bottom of a problem in your existing Umbraco installation?

If the answer is yes, please read on. Otherwise, make sure to head on over [to the download page](https://our.umbraco.com/download) and start using Umbraco CMS as intended.

**Table of contents**

[Building from source](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "building-from-source)

* [The quick build](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "quick)
* [Build infrastructure](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "build-infrastructure)
  + [Properties](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "properties)
  + [GetUmbracoVersion](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "getumbracoversion)
  + [SetUmbracoVersion](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "setumbracoversion)
  + [Build](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "build)
  + [Build-UmbracoDocs](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "build-umbracodocs)
  + [Verify-NuGet](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "verify-nuget)
* [Cleaning up](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "cleaning-up)

[Azure DevOps](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "azure-devops)

[Quirks](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "quirks)

* [Powershell quirks](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "powershell-quirks)
* [Git quirks](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "git-quirks)

## Building from source

Did you read ["Are you sure"](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "are-you-sure)?

### Quick!

To build Umbraco, fire up PowerShell and move to Umbraco's repository root (the directory that contains src, build, LICENSE.md...). There, trigger the build with the following command:

build/build.ps1

If you only see a build.bat-file, you're probably on the wrong branch. If you switch to the correct branch (v8/contrib) the file will appear and you can build it.

You might run into [Powershell quirks](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/BUILD.md" \l "powershell-quirks).

If it runs without errors; Hooray! Now you can continue with [the next step](https://github.com/umbraco/Umbraco-CMS/blob/v9/contrib/.github/CONTRIBUTING.md" \l "how-do-i-begin) and open the solution and build it.

### Build Infrastructure

The Umbraco Build infrastructure relies on a PowerShell object. The object can be retrieved with:

$ubuild = build/build.ps1 -get

The object exposes various properties and methods that can be used to fine-grain build Umbraco. Some, but not all, of them are detailed below.

#### Properties

The object exposes the following properties:

* SolutionRoot: the absolute path to the solution root
* VisualStudio: a Visual Studio object (see below)
* NuGet: the absolute path to the NuGet executable
* Zip: the absolute path to the 7Zip executable
* VsWhere: the absolute path to the VsWhere executable
* NodePath: the absolute path to the Node install
* NpmPath: the absolute path to the Npm install

The Visual Studio object is null when Visual Studio has not been detected (eg on VSTS). When not null, the object exposes the following properties:

* Path: Visual Studio installation path (eg some place under Program Files)
* Major: Visual Studio major version (eg 15 for VS 2017)
* Minor: Visual Studio minor version
* MsBuild: the absolute path to the MsBuild executable

#### GetUmbracoVersion

Gets an object representing the current Umbraco version. Example:

$v = $ubuild.GetUmbracoVersion()

Write-Host $v.Semver

The object exposes the following properties:

* Semver: the semver object representing the version
* Release: the main part of the version (eg 7.6.33)
* Comment: the pre release part of the version (eg alpha02)
* Build: the build number part of the version (eg 1234)

#### SetUmbracoVersion

Modifies Umbraco files with the new version.

This entirely replaces the legacy UmbracoVersion.txt file. Do not edit version infos in files.

The version must be a valid semver version. It can include a pre release part (eg alpha02) and/or a build number (eg 1234). Examples:

$ubuild.SetUmbracoVersion("7.6.33")

$ubuild.SetUmbracoVersion("7.6.33-alpha.2")

$ubuild.SetUmbracoVersion("7.6.33+1234")

$ubuild.SetUmbracoVersion("7.6.33-beta.5+5678")

#### Build

Builds Umbraco. Temporary files are generated in build.tmp while the actual artifacts (zip files, NuGet packages...) are produced in build.out. Example:

$ubuild.Build()

Some log files, such as MsBuild logs, are produced in build.tmp too. The build directory should remain clean during a build.

**Note: web.config**

Building Umbraco requires a clean web.config file in the Umbraco.Web.UI project. If a web.config file already exists, the pre-build task (see below) will save it as web.config.temp-build and replace it with a clean copy of web.Template.config. The original file is replaced once it is safe to do so, by the pre-packages task.

#### Build-UmbracoDocs

Builds umbraco documentation. Temporary files are generated in build.tmp while the actual artifacts (docs...) are produced in build.out. Example:

Build-UmbracoDocs

Some log files, such as MsBuild logs, are produced in build.tmp too. The build directory should remain clean during a build.

#### Verify-NuGet

Verifies that projects all require the same version of their dependencies, and that NuSpec files require versions that are consistent with projects. Example:

Verify-NuGet

### Cleaning up

Once the solution has been used to run a site, one may want to "reset" the solution in order to run a fresh new site again.

At the very minimum, you want

git clean -Xdf src/Umbraco.Web.UI/App\_Data

rm src/Umbraco.Web.UI/web.config

Then, a simple 'Rebuild All' in Visual Studio will recreate a fresh web.config but should be quite fast (since it does not really need to rebuild anything).

The clean Git command force (-f) removes (-X, note the capital X) all files and directories (-d) that are ignored by Git.

This will leave media files and views around, but in most cases, it will be enough.

To perform a more complete clear, you will want to also delete the content of the media, views, scripts... directories.

The following command will force remove all untracked files and directories, whether they are ignored by Git or not. Combined with git reset it can recreate a pristine working directory.

git clean -xdf .

For git documentation see:

* git [clean](https://git-scm.com/docs/git-clean)
* git [reset](https://git-scm.com/docs/git-reset)

## Azure DevOps

Umbraco uses Azure DevOps for continuous integration, nightly builds and release builds. The Umbraco CMS project on DevOps [is available for anonymous users](https://umbraco.visualstudio.com/Umbraco%20Cms).

DevOps uses the Build-Umbraco command several times, each time passing a different target parameter. The supported targets are:

* pre-build: prepares the build
* compile-belle: compiles Belle
* compile-umbraco: compiles Umbraco
* pre-tests: prepares the tests
* compile-tests: compiles the tests
* pre-packages: prepares the packages
* pkg-zip: creates the zip files
* pre-nuget: prepares NuGet packages
* pkg-nuget: creates NuGet packages

All these targets are executed when Build-Umbraco is invoked without a parameter (or with the all parameter). On VSTS, compilations (of Umbraco and tests) are performed by dedicated DevOps tasks. Similarly, creating the NuGet packages is also performed by dedicated DevOps tasks.

Finally, the produced artifacts are published in two containers that can be downloaded from DevOps: zips contains the zip files while nuget contains the NuGet packages.

During a DevOps build, some environment UMBRACO\_\* variables are exported by the pre-build target and can be reused in other targets and in DevOps tasks. The UMBRACO\_TMP environment variable is used in Umbraco.Tests to disable some tests that have issues with DevOps at the moment.

## Quirks

### PowerShell Quirks

There is a good chance that running build.ps1 ends up in error, with messages such as

The file ...\build.ps1 is not digitally signed. You cannot run this script on the current system. For more information about running scripts and setting execution policy, see about\_Execution\_Policies.

PowerShell has Execution Policies that may prevent the script from running. You can check the current policies with:

PS> Get-ExecutionPolicy -List

Scope ExecutionPolicy

----- ---------------

MachinePolicy Undefined

UserPolicy Undefined

Process Undefined

CurrentUser Undefined

LocalMachine RemoteSigned

Policies can be Restricted, AllSigned, RemoteSigned, Unrestricted and Bypass. Scopes can be MachinePolicy, UserPolicy, Process, CurrentUser, LocalMachine. You need the current policy to be RemoteSigned—as long as it is Undefined, the script cannot run. You can change the current user policy with:

PS> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

Alternatively, you can do it at machine level, from within an elevated PowerShell session:

PS> Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy RemoteSigned

And then the script should run. It might however still complain about executing scripts, with messages such as:

Security warning - Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning message. Do you want to run ...\build.ps1? [D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"):

This is usually caused by the scripts being blocked. And that usually happens when the source code has been downloaded as a Zip file. When Windows downloads Zip files, they are marked as blocked (technically, they have a Zone.Identifier alternate data stream, with a value of "3" to indicate that they were downloaded from the Internet). And when such a Zip file is un-zipped, each and every single file is also marked as blocked.

The best solution is to unblock the Zip file before un-zipping: right-click the files, open Properties, and there should be a Unblock checkbox at the bottom of the dialog. If, however, the Zip file has already been un-zipped, it is possible to recursively unblock all files from PowerShell with:

PS> Get-ChildItem -Recurse \*.\* | Unblock-File

### Git Quirks

Git might have issues dealing with long file paths during build. You may want/need to enable core.longpaths support (see [this page](https://github.com/msysgit/msysgit/wiki/Git-cannot-create-a-file-or-directory-with-a-long-path) for details).