# How you can Work with Agility Content

Agility is a headless CMS with traditional capabilities. There are two ways to build websites that integrate with your Agility CMS instance. Each has its own benefits, and special care should be taken to choose which path is right for your project.

1. Use the **Sync Model** - Have published content *pushed* directly to your web server.
2. Use the **Content Fetch API** - *Pull* content on your own terms.

## Sync Model

The **Sync Model** allows a web server to maintain a copy of all published content in a local Content Repository. To facilitate this, you must register each Syncing Web Server in Agility as well as configure the website to be able to accept Sync requests and ability store/read content in its file system.

### Requirements

* The website must be an ASP.NET Framework/Core MVC project
* Must reference the Agility.Web/AspNetCore DLL
* Each website environment must be registered as a Syncing Web Server
* The web app must have access to the local file system

### Benefits

* Highly performant, scalable, and optimized way of accessing content
* No dependencies on Agility CMS, all content is stored and accessed locally

### Sample Use-cases

* You are building a .NET website, you have a team of .NET developers, Visual Studio is what they work in, and you need your site to be able to handle high traffic scenarios *out of the box*
* You have an existing ASP.NET MVC web app and you want to add Agility to it

## Content Fetch API

The **Content Fetch API** allows developers using any programming language to be able to query and use content from Agility. The API is a low-latency, highly available service, and is powered by a global CDN.

### Requirements

As the **Content Fetch API** is a standard REST service, it does not have any specific requirements other than requiring the use of an app token password.

### Benefits

* Use any programming language/framework
* More flexible architecture
* Reduces the complexity of the solution by removing the need for a sync process
* Easier to share content with other teams
* Ability to develop 100% client web apps using the JS SDK
* Open source SDKs for popular frontend frameworks such as React, Vue, etc.

### Sample Use Cases

* You do not use the .NET development stack and prefer to have full control over what and how content is used from the CMS
* You want to build a web app that does not require any server-rendering
* You want to build a JAMStack website
* You need to consume content from a mobile app (iOS, Android) or IoT device
* You want to build a website using the programming language and framework of your choice