A statically generated blog example using Next.js and Agility CMS

This example showcases Next.js's Static Generation feature using Agility CMS as the data source.

> `IMPORTANT` - *This example uses Agility CMS's **Page Management** features. This means that the CMS ultimately drives what pages are available and what content is on each page. This enables **Editors** to focus on managing their pages, while allowing you, (the **Developer**) to focus on building UI components for the editors to compose their pages.*

Demo

- **Live**: https://next-blog-agilitycms.vercel.app/
- **Preview Mode**: https://next-blog-agilitycms.vercel.app/?agilitypreviewkey=...

Deploy your own

Once you have access to the environment variables you'll need, deploy the example using Vercel:

Related examples

- WordPress
- DatoCMS
- Sanity
- TakeShape
- Prismic
- Contentful
- Strapi
- Cosmic
- ButterCMS
- Storyblok
- GraphCMS
- Kontent
- Ghost
- Umbraco Heartcore
- Blog Starter
- Builder.io

How to use

Execute `create-next-app` with npm or Yarn to bootstrap the example:

```
npx create-next-app --example cms-agilitycms cms-agilitycms-app
# or
yarn create next-app --example cms-agilitycms cms-agilitycms-app
# or
pnpm create next-app -- --example cms-agilitycms cms-agilitycms-app
```

Configuration

How is this Different from Other CMS Examples?

The key principle behind Agility CMS is that **Editors** should have full control of their pages and what content is on each page without getting into code.

This means you'll not only be defining **Content** for your `Posts` and `Authors`, but you'll also be defining UI Components to compose your pages. This site will consist of a single **Page Template** and a collection of **Modules** that represent the UI components you see on the page.

> ***NOTE*** *-* `Modules` *and* `Page Templates` *in Agility CMS simply correspond to* `React Components` *in your website.*

Once you've gone through the steps below, you'll be able to dynamically manage pages (and what is on them) directly through the CMS without requiring a developer.

Step 1. Create an account and a project on `Agility CMS`

First, create an account on Agility CMS.

After creating an account you'll be asked to create a new project. Use any name of your liking as the **Project Name** and select the **Blank (advanced users)** template to create a blank Agility CMS instance.

Step 2. Create an `Author` Content Definition

From within the Agility CMS Content Manager, navigate to **Settings** > **Content Definitions** and click **New** to create a new **Content Definition**.

- The **Title** should be `Author`. This will also pre-populate **Reference Name** for you.

Next, add these fields via the **Form Builder** tab (you don't have to modify any other settings):

- `Name` - Set **Field Label** to `Name` and **Field Type** to `Text`
- `Picture` - Set **Field Label** to `Picture` and **Field Type** to `Image`

When you are done, click **Save & Close** to save your `Author` content definition.

Step 3. Create a `List` based on your `Author` Content Definition

From within the Agility CMS Content Manager, navigate to **Shared Content** and click the **+ (New)** button, then fill the form like so:

- **Type** should be `Content List`
- **Content Definition** should be **Author**
- **Display Name** should be set to **Authors**. This will also pre-populate **Reference Name** for you.

Step 4. Create a `Post` Content Definition

From within the Agility CMS Content Manager, navigate to **Settings** > **Content Definitions** and click **New** to create a new **Content Definition**.

- The **Title** should be `Post`.

Next, add these fields via the **Form Builder** tab (you don't have to modify any other settings):

- `Title` - Set **Field Type** to `Text`
- `Slug` - Set **Field Type** to `Text`
- `Date` - Set **Field Type** to `Date/Time`
- `AuthorID` - Set **Field Type** to `Number` and enable **Hide field from input form**
- `Author` - Do the following:
  - **Field Type** - `Linked Content`
  - **Content Definition** - `Author`
  - **Content View** - `Shared Content`
  - **Shared Content** - `Authors`
  - **Render As** - `Dropdown List`
  - **Save Value To Field** - `AuthorID`
- `Excerpt` - Set **Field Type** to `Text`
- `Content` - Set **Field Type** to `HTML`
- `Cover Image` - Set **Field Type** to `Image`

When you are done, click **Save & Close** to save your `Post` content definition.

Step 5. Create a `Dynamic Page List` based on your `Posts` Content Definition

From within the Agility CMS Content Manager, navigate to **Shared Content** and click the **+ (New)** button, then fill the form like so:

- **Type** should be `Dynamic Page List`
- **Content Definition** should be `Post`
- **Display Name** should be `Posts`. This will also pre-populate **Reference Name** for you

Step 6. Populate Content

Go to **Shared Content**, select the **Authors** list and click the **+ New** button to create a new content item:

- You just need **1 Author content item**.
- Use dummy data for the text.
- For the image, you can download one from Unsplash.

Click on **Save** and **Publish** once you're done.

Next, select the **Posts** list and click the **+ New** button to create a new content item:

- We recommend creating at least **2 Post content items**.
- Use dummy data for the text.
- You can write markdown for the **Content** field.
- For the images, you can download ones from Unsplash.
- Pick the **Author** you created earlier.

For each post content item, you need to click `Publish` after saving. If not, the post will be in the `Staging` state.

Step 7. Define your `Intro` Module

Navigate to **Settings** > **Module Definitions** and click **New** to create a new **Module Definition**.

- Set **Title** to `Intro`
- Set **Description** to `Displays an intro message.`

In this case, we are not adding any fields to control the output or behavior, since the content is actually hard-coded in the template.

Click **Save & Close** to save the definition.

Step 8. Define your `Hero Post` Module

Navigate to **Settings** > **Module Definitions** and click **New** to create a new **Module Definition**.

- Set **Title** to `Hero Post`
- Set **Description** to `Displays the latest Post.`

In this case, we are not adding any fields to control the output or behavior, since the latest post will be used by default and all of the data is associated to the post itself.

Click **Save & Close** to save the definition.

Step 9. Define your `More Stories` Module

Navigate to **Settings** > **Module Definitions** and click **New** to create a new **Module Definition**.

- Set **Title** to `More Stories`
- Set **Description** to `Displays a listing of Posts.`

Next, add the following field:

- `Title` - Set **Field Type** to `Text`

Click **Save & Close** to save the definition.

Step 10. Define your `Post Details` Module

Navigate to **Settings** > **Module Definitions** and click **New** to create a new **Module Definition**.

- Set **Title** to `Post Details`
- Set **Description** to `Displays the details of a Post.`

In this case, we are not adding any fields to control the output or behavior, since the data is associated to the post itself.

Click **Save & Close** to save the definition.

Step 11. Define a `One Column` Page Template

Navigate to **Settings** > **Page Templates** and click **New** to create a new **Page Template**.

- **Name** should be `One Column Template`
- **Digital Channel Type** should be `Website`
- Under **Module Zones** click `+ (New)`
  - Set **Display Name** to `Main Content Zone`, it will populate **Reference Name** for you
  - Click `Save` to apply the `Main Content Zone`

Click **Save & Close** to save the page template.

Step 12. Add a new Page called `home`

Navigate to **Pages** and click the **+ (New)** button in the page tree to create a new **Page**.

- Set **Type** to `Page`
- Set **Page Template** to `One Column Template`
- Set **Menu Text** to `Home` - **Page Title** and **Page Name** fields will be auto-populated.

Click **Save** to create the `/home` page.

Next, let's add the `Intro`, `Hero Post` and `More Stories` modules to the `Main Content Zone` of the `home` page:

- Click the **+ (New)** button on `Main Content Zone` and select `Intro` to add the module to the page
- Click **Save & Close** on the module to return back to the page
- Click the **+ (New)** button on `Main Content Zone` and select `Hero Post` to add the module to the page
- Click **Save & Close** on the module to return back to the page
- Click the **+ (New)** button on `Main Content Zone` and select `More Stories` to add the module to the page
    - Set **Title** to `More Stories`
- Click **Save & Close** on the module to return back to the page

Then click **Publish** on the page in order to publish the page and all of its modules.

Step 13. Add a new Folder called `posts`

Navigate to **Pages** and click the `Website` channel, then click the **+ (New)** button in the page tree to create a new **Folder** in the root of the site:

- Set **Type** to `Folder`
- Set **Menu Text** to `Posts`, **Folder Name** will be auto-populated to `posts`

Click **Save** to create the `/posts` folder.

Important: Click **Publish** on the folder.

Step 14. Add a new Dynamic Page called `posts-dynamic`

Navigate to **Pages** and select the existing `/posts` folder. Click the **+ (New)** button in the page tree to create a new **Dynamic Page** underneath the `posts` page.

- Set **Type** to `Dynamic Page`
- Set **Page Template** to `One Column Template`
- Set **Build Pages From** to `Posts`
- Set **Sitemap Label** to `posts-dynamic`
- Set **Page Path Formula** to `##Slug##`
- Set **Page Title Formula** and **Menu Text Formula** to `##Title##`

Click **Save** to create the `/posts/posts-dynamic` dynamic page.

Next, let's add the `Post Details` and `More Stories` modules to the `Main Content Zone` of the `posts-dynamic` page:

- Click the **+ (New)** button on `Main Content Zone` and select `Post Details` to add the module to the page
- Click the **+ (New)** button on `Main Content Zone` and select `More Stories` to add the module to the page
  - Set **Title** to `More Stories`
- Click **Save & Close** on the module to return back to the `posts-dynamic` page

Then click **Publish** on the page in order to publish the page and all of its modules.

Step 15. Set up environment variables

Copy the `.env.local.example` file in this directory to `.env.local` (which will be ignored by Git):

```
cp .env.local.example .env.local
```

Go to the **Getting Started** section from the menu and click on **API Keys**. You should see a new modal called `Content API Details`, then click in the **Show API Key(s)** button within it.

Then set each variable on `.env.local`:

- `AGILITY_CMS_GUID` should be the **Instance GUID** field
- `AGILITY_CMS_API_FETCH_KEY` should be the **Live API Key** field
- `AGILITY_CMS_API_PREVIEW_KEY` should be the **Preview API Key** field - this is used when the site is in Preview Mode and allows your site to pull the latest content, regardless of whether it is published or not.
- `AGILITY_CMS_SECURITY_KEY` should be the **Security Key** field that can be found in **Settings** > **Global Security** - this is used to communicate between the CMS and your site to validate Preview Mode

Your `.env.local` file should look like this:

```
AGILITY_CMS_GUID=...
AGILITY_CMS_API_FETCH_KEY=...
AGILITY_CMS_API_PREVIEW_KEY=...
AGILITY_CMS_SECURITY_KEY=...
```

Step 16. Run Next.js in development mode

```
npm install
npm run dev

# or

yarn install
yarn dev
```

Your blog should be up and running on http://localhost:3000! If it doesn't work, post on GitHub discussions.

Step 17. Deploy on Vercel

You can deploy this app to the cloud with Vercel (Documentation).

Deploy Your Local Project

To deploy your local project to Vercel, push it to GitHub/GitLab/Bitbucket and import to Vercel.

Important: When you import your project on Vercel, make sure to click on **Environment Variables** and set them to match your `.env.local` file.

Deploy from Our Template

Alternatively, you can deploy using our template by clicking on the Deploy button below.

Step 18. Try preview mode

Now that you've deployed your app to Vercel, take note of the URL of your deployed site. This will be registered in Agility CMS so that when editors click the `Preview` button within Agility CMS, your app is loaded in **Preview Mode**. Learn more about NextJS Preview Mode).

To enable the Preview Mode, you'll need to add your site to **Domain Configuration** in Agility CMS:

- Go to **Settings** > **Domain Configuration**
- Click on the existing channel in the list called `Website`
- Click on the **+ (New)** button to add a new domain:
  - Set **Name** to `Production`
  - Set **Domain URL** to the URL of your production deployment, it should look like `https://<your-vercel-domain>.vercel.app`
  - Enable **Preview Domain**
  - Click **Save** to save your settings

Go to one of your `Posts` and update the title. For example, you can add `[Staging]` in front of the title. Click **Save**, but **DO NOT** click **Publish**. By doing this, the post will be in the staging state.

To enter **Preview Mode**, click the `Preview` button on the details of your `Post`. This redirects you to the `/` page, however you will now be in **Preview Mode** so you can navigate to your `Post` you want to view on the website.

You should now be able to see the updated title. To exit the preview mode, you can click **Click here to exit preview mode** at the top.

> *NOTE - To set up preview on a specific `Post` (as opposed to the `/` page), click on the **Settings** tab of the `Post` list in **Shared Content**. For **Item Preview Page** set it to `~/posts/posts-dynamic` and for **Item Preview Query String Parameter** set it to `contentid`.*