A DRUPALHELPS REFERENCE GUIDE

The Ultimate Guide to Media in Drupal 11

Turn the Media Library into a digital asset management system with core, contrib and a little AI.

Rod Martin

DrupalHelps.com · NavigateTomorrow, Inc.

Drupal 11 · November 2026

Contents

1. How to use this guide
2. What changed since the last edition
3. The five-layer model
4. Layer 1: Store
5. Layer 2: Find
6. Layer 3: Shape
7. Layer 4: Govern
8. Layer 5: Automate
9. When you need a dedicated DAM
10. The 12-step build checklist
11. Module reference
12. Sources

How to use this guide

This guide goes with the workshop deck of the same name. The slides show the idea. This document gives you the steps, the settings and the reasons behind them, so you can build it on a real site afterward.

Work through it in order. Each layer depends on the one below it: you can't filter by a field you never created, and AI can't tag from a vocabulary that doesn't exist.

> **Command labels**
>
> Every command block says where it runs. **Local** means your DDEV project. **Server** means production or staging. Run the AVIF check on both, because DDEV's PHP build and your host's PHP build often differ.

> **Versions**
>
> Module versions and requirements are as of November 2026. Check each project page before you install. Contrib moves faster than this guide does.

INTRODUCTION

What changed since the last edition

The goal hasn't changed: build a digital asset management system (DAM) from Drupal core and contributed modules. What changed is the bar. In 2022, "DAM" meant a media type with keywords and an embed button. In 2026 it means editors can search the library, the system shapes every image for every screen, you know where each asset is used and when you lose the right to show it, and AI handles the typing.

Five changes drive this edition.

Change	What it means for your build
CKEditor 5	Media embeds use media view modes. Editors pick a size, and you decide what each size means in Manage display.
Drupal CMS media recipe	One command installs media types, Focal Point, responsive image styles, SVG support, bulk upload and Media File Delete.
Drupal 11.2	Image styles can convert to AVIF with a WebP fallback, in core.
Drupal AI	Alt text, image tags and focal point suggestions come from your AI provider, with an editor approving them.
Entity Usage 5	The Drupal 12 path. It needs Drupal 11.4 or later and a usage rebuild.

Stop doing these

Some advice from the earlier edition is now the slow way, or the wrong one.

Old approach	Do this instead
IEK (Image Effect Kit)	Core image styles, plus Image Effects if you need watermarks. IEK is a Drupal 8-era module.
Asset Injector with 20 lines of padding-hack CSS for video	Four lines of `aspect-ratio` CSS in your theme.

Old approach	Do this instead
Image styles picked inside CKEditor 4	Media view modes in CKEditor 5's Embed media filter.
Typing every keyword by hand	AI suggestions drawn from your vocabulary, approved by an editor.
Building media types from scratch	Apply `drupal_cms_media`, then customize.

Same goal. Higher bar.

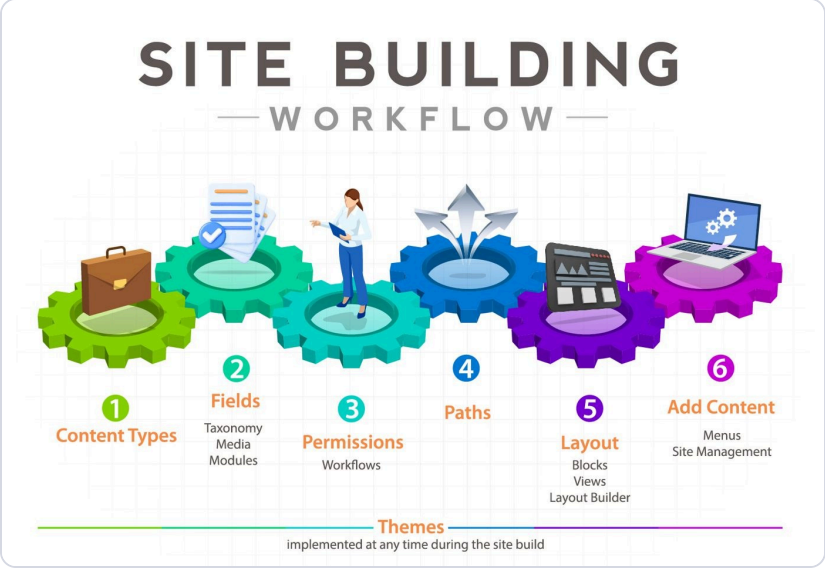
THE MODEL

The five-layer model

A DAM does five jobs. Most Drupal sites do the first one and stop. Each layer in this guide builds on the one below it.

05	**Automate**	AI metadata with human review
04	**Govern**	Usage, file replacement, workflow, expiry
03	**Shape**	Focal point, image styles, embeds, video
02	**Find**	Search screens, taxonomy, bulk tools
01	**Store**	Media types and shared fields

Plan media with your fields



The site building workflow. Media sits in step 2, beside taxonomy.

In the site building workflow, media sits in step 2, Fields, next to taxonomy. That's where it belongs: before permissions, before layout, before anyone uploads a file. Retrofitting metadata onto 3,000 existing uploads is the expensive way to build a DAM.

Decide the fields before the first upload.

LAYER 1

Store

Structure decides what the other four layers can do. Start with the recipe, then make three decisions: which media types you need, which fields every asset gets, and which vocabularies organize them.

Start with the recipe

The Drupal CMS media recipe does most of the setup the earlier edition did by hand. It requires Drupal 10.4 or later, or Drupal 11.

Local · DDEV

```
ddev composer require drupal/drupal_cms_media
ddev drush recipe ../recipes/drupal_cms_media
ddev drush cr
```

The `../recipes/` path assumes the standard `drupal/recommended-project` layout, where Composer puts recipes beside `web/`. If your project is laid out differently, point Drush at wherever Composer installed the recipe.

What you get:

- Media types for Image, Document, Local video and Remote video, plus SVG images
- Focal Point cropping on images
- Responsive image styles
- Media Library Bulk Upload
- Media File Delete
- ECA, which you'll use in Layer 4

A recipe applies configuration once. After that, it's your site's configuration. Change anything you like.

Type, field or term?

Ask this before you create anything. The most common mistake I see is a media type per department. That's a term.

Create a…	When	Example
Media type	The source, fields, display or permissions differ.	Logos: SVG source, no cropping, brand team only.
Field	Every asset needs the same fact recorded.	License, credit, rights expiry.
Term	Only the topic or audience differs.	Events, Staff, Campus, Products.

The metadata model

Six fields turn a media library into a DAM. Five go on every media type. Alt text is already part of the image field.

Field	Type	Purpose
Keywords	Entity reference: taxonomy (flat)	Search terms. Many per asset.
Collection	Entity reference: taxonomy (hierarchical)	Browsing by section. One or a few per asset.
Credit / source	Text (plain)	Photographer, agency or origin.
License	List (text)	Owned, Licensed, Creative Commons, Public domain.
Rights expire	Date	When you lose the right to show it.
Alt text	Part of the image field	Set it to required on the Image media type.

Add a shared field

1. Go to **Structure › Media types › Image › Manage fields** and create the field there first.
2. On each other media type, choose **Create a new field**, then **Re-use an existing field**, and pick the same field.
3. Keep the machine name identical on every type, for example `field_keywords`.

Re-using field storage is what makes Layer 2 work. One view can filter Images, Documents and Videos by the same Keywords field only if it's the same field.

One field, every media type, one machine name.

LAYER 2

Find

If an editor can't find an asset in 10 seconds, they upload it again. Now you have two copies, two sets of metadata and two files to replace when the logo changes. Search is what stops the duplicates.

Make the media list a search screen

Both media screens are Views, so you can change them without code.

1. Edit the **Media** view, which powers `/admin/content/media`.
2. Add exposed filters for Keywords, Collection and License.
3. Add Rights expire as a sortable column, so editors can see what's about to expire.
4. Edit the **Media library** view and add the same exposed filters to its widget displays. That's the picker editors see inside the node form, and it needs the filters more than the admin list does.

Organize with taxonomy

Two vocabularies do the organizing. Each answers a different question.

Vocabulary	Shape	Used for	Example terms
Collection	Hierarchical	Browsing by section	Events › 2026 › Rally; Campus › Buildings
Keywords	Flat	Searching	motorcycle, outdoor, summer, rider

The same photo can sit in Events and Campus at once. It's all core taxonomy, filtered in Views or Search API, with no extra module to maintain.

> **Control the vocabulary**
>
> Decide who can create terms. If every editor can add Keywords freely, you'll have "Motorcycle," "motorcycles" and "moto" within a month. Let editors pick terms, and give term creation to one or two people.

Bulk in. Bulk tag. Fix in place.

Job	Module	What it adds
Upload many	Media Library Bulk Upload	A multi-file upload page, with a permission per media type. Installed by the recipe.
Tag many	Views Bulk Operations + Views Bulk Edit	Select 50 assets in the Media view and set Collection and Keywords in one action.
Fix in place	Media Library Edit	An Edit button inside the media picker, so editors fix metadata without leaving the page.

Set up bulk tagging

1. Install Views Bulk Operations and Views Bulk Edit. VBO runs the actions. Views Bulk Edit supplies the one that changes field values.
2. Edit the Media view and add the **Views bulk operations** field.
3. Enable the **Modify field values** action and limit it to Keywords and Collection.
4. Give the permission to the roles that curate the library, not to every editor.

If editors can't find it, they'll upload it again.

LAYER 3

Shape

One upload should serve every size and every format. Editors pick the asset. The system handles the crop, the format and the embed.

Focal Point: set it once, every crop obeys

Focal Point adds a crosshair to the image widget. Drop it on the subject, and every image style that uses the **Focal Point Scale and Crop** effect keeps that spot in frame.



Original upload, focal point on the rider.



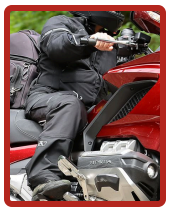
16:9 hero



1:1 card



4:5 avatar



Center crop

The last crop is the same 4:5 avatar without a focal point. You get a knee and a seat, not a face. That's the whole argument for Focal Point.

1. The recipe installs Focal Point and sets the Image widget to use it.
2. In each image style that crops, use **Focal Point Scale and Crop** instead of **Scale and crop**.
3. Train editors to drop the crosshair on the face or the subject, every time.

The AI Image Crop recipe can suggest the focal point for you. See Layer 5.

AVIF first, WebP as the fallback

Drupal 11.2 added a **Convert** option for AVIF with a WebP fallback to image styles. When the server can't write AVIF, Drupal serves WebP instead.

The size difference is real. In the test from the core issue that made AVIF the default for core's image styles, the same derivatives took 6,704 KB as WebP and 4,144 KB as AVIF. That's about 38% smaller.

Check the server first

AVIF needs a PHP GD library built with AVIF support. The Status report warns when it's missing, or you can check directly.

Server · production

```
# Does PHP's GD support AVIF?
drush php:eval "var_dump(function_exists('imageavif'));"

# After editing image styles
drush image:flush --all
```

Run the same check locally with `ddev drush` in front. A `true` in DDEV tells you nothing about production.

Add AVIF to an image style

1. Go to **Configuration › Media › Image styles** and edit a style.
2. Add the **Convert** effect as the last effect and choose AVIF, with WebP as the fallback.
3. Save, then flush that style.
4. Repeat for each style used in a responsive image style.

Pair it with responsive image styles. AVIF makes each file smaller. Responsive styles make sure a phone never downloads the desktop hero.

Media in CKEditor 5

Editors pick a size, not a style. You decide what each size means.

1. Edit the text format at **Configuration › Content authoring › Text formats and editors**.
2. Drag the **Insert Media** button into the toolbar.
3. Enable the **Embed media** filter and choose which media types editors can embed.

4. Under the filter settings, allow the view modes editors may choose, for example Small, Medium, Wide and Full.
5. On the Image media type, open **Manage display** for each of those view modes and point it at a responsive image style.

Change a view mode once and every embed on the site updates.

100% responsive remote video

The earlier edition used Asset Injector and a 20-line padding hack. Modern browsers support `aspect-ratio`, so the fix is 4 lines in your theme.

Before: Asset Injector, padding hack

```
.field--name-field-media-oembed-video {
  position: relative; overflow: hidden;
  width: 100%; padding-top: 56.25%;
}
.media-oembed-content {
  position: absolute;
  top: 0; left: 0; bottom: 0; right: 0;
  width: 100%; height: 100%;
}
```

After: your theme's CSS

Theme · css/components/media.css

```
.media-oembed-content {
  width: 100%;
  height: auto;
  aspect-ratio: 16 / 9;
}
```

Add the file to a library in your theme's `*.libraries.yml` and attach it globally or to the media template. It ships with the theme, survives deployments and needs no extra module.

Use the oEmbed Providers module only when you need a provider that isn't in the public oEmbed list, such as a private video host.

CSS belongs in the theme, not in the database.

LAYER 4

Govern

Governance answers three questions: where is this asset used, who approved it, and when do we lose the right to show it. Most media libraries can answer none of them.

Where is this image used?

Entity Usage tracks every reference to a media item and adds a **Usage** tab listing the pages that use it. Turn on the delete warning so editors see that list before they remove an asset that's still in use.

Entity Usage branch	Use it when
2.3	You're on Drupal 10.3 or later, or Drupal 11 below 11.4.
5.0	You're on Drupal 11.4 or later, or planning for Drupal 12. Rebuild usage data after upgrading.

Media File Delete closes the other gap. By default, deleting a media item leaves its file on disk. With Media File Delete, the file goes too. The recipe installs it.

New file, same link

Media Entity File Replace adds a **Replace file** widget to the media edit form. Check **Overwrite original file** and the new upload keeps the original path and filename.

That matters for PDFs. The fee schedule linked from last year's emails and printed QR codes keeps working, and every page that references the media item shows the new version on save.

> **Purge the cache**
>
> Same URL means browsers and CDNs may keep serving the old file. Purge the CDN after a replace, or the "new" PDF is the old one for a while.

Assets have a lifecycle too

Content Moderation applies to media types the same way it applies to content types. Scheduler 2.x supports media entities, so you can schedule an asset to publish or unpublish.



Review before publish

1. At **Configuration › Workflow › Workflows**, edit your editorial workflow.
2. Under **This workflow applies to**, select the media types that need review. Logos and leadership photos are the usual candidates.
3. Give the review transition to the roles that own brand and communications.

Unpublish when rights end

1. Install Scheduler and enable unpublishing on the media types that carry licensed assets.
2. Set the unpublish date to match Rights expire.
3. To fill that date automatically, use ECA, which the recipe installs, to copy Rights expire into Scheduler's unpublish field when the media item is saved. Test this model on your build before you rely on it.

An expired license is a liability, not a setting.

LAYER 5

Automate

Let AI do the typing. Keep people in charge of the meaning. Metadata that gets added on upload is metadata that exists. Metadata that's planned for "Friday afternoon" doesn't.

The pipeline

Step	Tool	Result
1. Upload	Media Library	Editor adds an image.
2. Alt text	AI Image Alt Text	A suggested description in the alt field.
3. Tags and description	AI image classification recipe	Terms chosen from your Keywords vocabulary.
4. Focal point	AI Image Crop recipe	A suggested focal point on the subject.
5. Review	The editor	A person checks it and saves.

Set it up

1. Install the AI module and one provider module. Store the API key with the Key module and select the provider under **Configuration › AI**.
2. Install AI Image Alt Text and choose the provider and model in its settings.
3. Apply the AI image classification recipe and point it at your Keywords vocabulary.
4. Apply the AI Image Crop recipe for focal point suggestions.
5. Backfill old images with the **AI Image Bulk Alt Text** submodule. It writes suggestions for images that have no alt text.

Four guardrails

Push back on these before you switch anything on.

Guardrail	Why
Your vocabulary, not AI's	Tag from terms you curate. Free-form AI tags turn a library into tag soup within a month.
Context makes alt text	The same photo needs different alt text on a news story and a staff page. AI sees the image, not the page it's on.
A person approves	Use the generate button, review, then save. Auto-publishing AI text on a public site is a compliance risk.
Know where images go	Every image is sent to your AI provider. Pick one the client has approved, and get that in writing.

AI drafts the metadata. A person owns it.

KNOW YOUR LIMITS

When you need a dedicated DAM

Everything in this guide is enough for most organizations. These are the signs it isn't:

- Several sites or brands must share one library.
- Print, video and design teams work outside Drupal.
- Rights are negotiated per channel, region or agency.
- Approvals and versions matter more than web delivery.

Then connect a dedicated DAM to Drupal instead of replacing Drupal's media. Acquia DAM, for one, shows up as a tab inside the Media Library, so editors keep the same picker. Until you hit those signs, don't buy one.

THE BUILD

The 12-step build checklist

This replaces the 10-step list from the earlier edition. Work top to bottom.

- [ ] Apply the `drupal_cms_media` recipe.
- [ ] Add shared fields on every media type: Keywords, Collection, Credit, License, Rights expire. Make alt text required.
- [ ] Turn the Media view and the Media library view into search screens.
- [ ] Set up bulk upload, bulk tagging and Media Library Edit.
- [ ] Use Focal Point Scale and Crop in every cropping image style.
- [ ] Add AVIF with a WebP fallback, after checking the server supports it.
- [ ] Embed media in CKEditor 5 with view modes mapped to responsive image styles.
- [ ] Make remote video responsive in the theme.
- [ ] Track usage with Entity Usage, and delete files with Media File Delete.
- [ ] Replace files without breaking links with Media Entity File Replace.
- [ ] Add moderation and expiry to media with Content Moderation and Scheduler.
- [ ] Add AI alt text, tags and focal points, with an editor reviewing each one.

REFERENCE

Module reference

Module	Source	Status	Role
Media, Media Library	Core	Keep	The foundation
Image (Convert to AVIF)	Core	Keep	AVIF with WebP fallback, 11.2+
Responsive Image	Core	Keep	Breakpoint-based image styles
CKEditor 5	Core	Keep	Media embeds with view modes
Content Moderation, Workflows	Core	Keep	Review before publish
Focal Point	Contrib	Keep	Subject-aware cropping
Media Library Bulk Upload	Contrib	Keep	Multi-file upload
Media Entity File Replace	Contrib	Keep	Replace a file, keep the URL
Entity Usage	Contrib	Keep	Usage tab and delete warnings
oEmbed Providers	Contrib	If needed	Custom oEmbed providers only
Media File Delete	Contrib	Add	Delete the file with the media
Scheduler	Contrib	Add	Scheduled publish and unpublish
Views Bulk Operations, Views Bulk Edit	Contrib	Add	Bulk tagging
Media Library Edit	Contrib	Add	Edit inside the picker
AI, AI Image Alt Text	Contrib	Add	AI provider and alt text
AI image classification, AI Image Crop	Recipes	Optional	Tags and focal point suggestions
IEK	Contrib	Drop	Replaced by core image styles
Asset Injector (for video CSS)	Contrib	Drop	Replaced by theme CSS

REFERENCE

Sources

Versions, requirements and figures in this guide were checked against these pages in November 2026.

- Core issue, AVIF conversion with WebP fallback: drupal.org/i/3510582
- Change record, AVIF default for core image styles (source of the 6,704 KB and 4,144 KB figures): drupal.org/node/3524206
- Drupal CMS media recipe: packagist.org/packages/drupal/drupal_cms_media
- Entity Usage: drupal.org/project/entity_usage
- Entity Usage 5.0 and Drupal 12, The Drop Times: thedroptimes.com/node/72293
- Media Entity File Replace: drupal.org/project/media_entity_file_replace
- Media File Delete: drupal.org/project/media_file_delete
- Scheduler documentation, media support: project.pages.drupalcode.org/scheduler
- AI Image Alt Text: drupal.org/project/ai_image_alt_text
- AI image classification recipe: drupal.org/project/ai_recipe_image_classification
- AI Image Crop recipe: packagist.org/packages/drupal/ai_recipe_image_crop
- Focal Point: drupal.org/project/focal_point

DRUPALHelps

Training, guides and help: **drupalhelps.com**