Image Optimizer

Magento 2 Module — Complete User Guide

Complete Guide — Every Screen, Every Button, Step by Step

Convert catalog, category and CMS images to modern WebP and serve them through an HTML <picture> element with a JPEG/PNG fallback and native lazy-loading. Optimize on-the-fly, on a cron schedule, or in bulk via CLI — and review every conversion and the disk space saved in a full Optimization Log. Hyvä and Luma compatible, licensed via the eTechFlow portal.

Prepared by: eTechFlow Pvt. Ltd.
Contact: etechflow0@gmail.com
Date: July 2026

Table of Contents

1. Introduction

Image Optimizer converts your Magento store images — product cache, category images, and optional CMS / Page Builder images — into the modern WebP format to shrink file sizes and speed up page loads. On the storefront each image is served through an HTML <picture> element with a JPEG/PNG fallback and native lazy-loading, so modern browsers get smaller WebP files while older browsers still get the original.

Conversion can happen on-the-fly the first time an image is shown, automatically on a cron schedule, or in bulk via the CLI. Every conversion is recorded in the built-in Optimization Log, where you can see the disk space saved. The module is licensed through the eTechFlow portal. This guide covers configuring and using the module.

2. Configuration

All settings live under Stores → Configuration → eTechFlow → Image Optimizer, grouped into General Settings, Frontend Output, Image Coverage and Automatic Optimization.

2.1 General Settings

The core conversion settings.

[Module Enabled] Turns the whole module on or off without uninstalling it.

[WebP Quality (1-100)] Output quality. Higher = larger files with marginal gains past ~85. Default 80 is the photo sweet spot.

[Engine Priority Order] Comma-separated conversion engines tried in order, skipping any unavailable. Recommended: cwebp, imagick, gd (binary first, then PHP extensions).

[Bulk Batch Size] Images processed per CLI batch / cron tick. Default 200; lower it if the process is OOM-killed on very large catalogs.

2.2 Frontend Output & Image Coverage

Frontend Output controls the storefront markup; Image Coverage chooses which image directories the bulk converter walks.

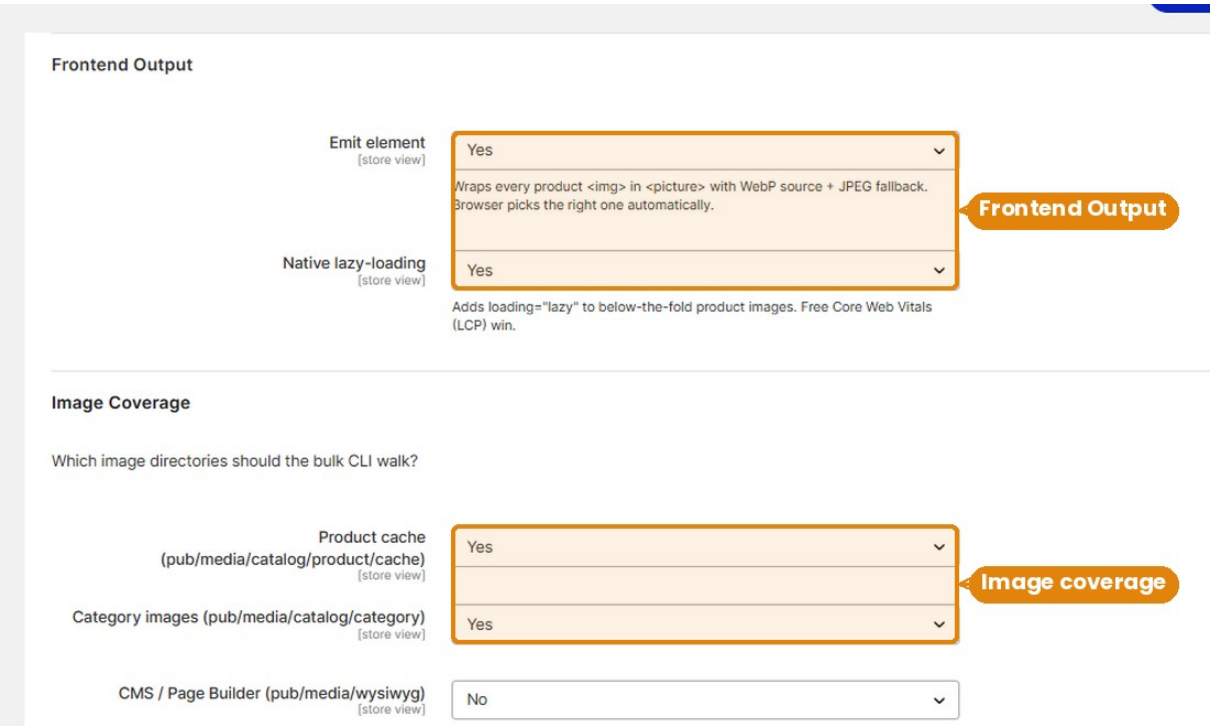

Figure 1 — Frontend Output (Emit <picture> element, Native lazy-loading) and Image Coverage (product cache, category images, CMS / Page Builder).

[Emit element] Yes wraps every product <img> in a <picture> element with a WebP source + JPEG fallback; the browser automatically picks the right one.

[Native lazy-loading] Yes adds loading="lazy" to below-the-fold images — a free Core Web Vitals (LCP) win.

[Product cache (.../catalog/product/cache)] Yes includes resized product images in bulk optimization.

[Category images (.../catalog/category)] Yes includes category images in bulk optimization.

[CMS / Page Builder (.../wysiwyg)] Off by default; turn on if your CMS pages use heavy hero images.

2.3 Automatic Optimization

Keep new images optimized without running the CLI by hand.

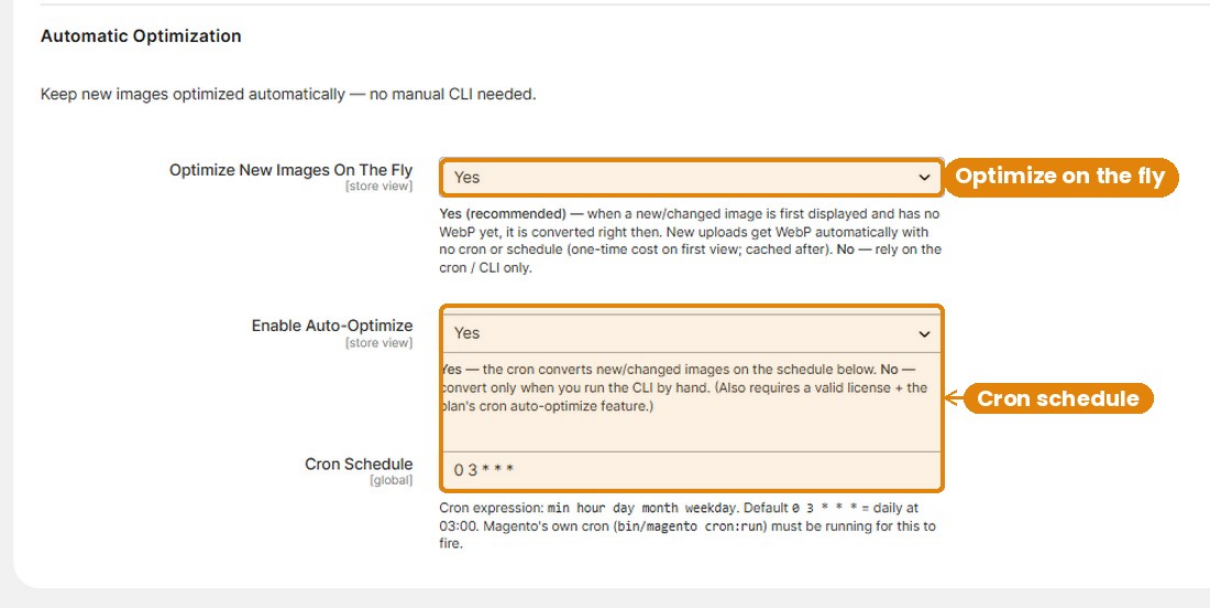

Figure 2 — Automatic Optimization: optimize new images on the fly, enable the cron auto-optimize, and set the Cron Schedule.

[Optimize New Images On The Fly] Yes (recommended) converts a new/changed image to WebP the first time it is displayed (a one-time cost on first view, cached after). No relies on cron/CLI only.

[Enable Auto-Optimize] Yes lets the cron convert new/changed images on the schedule below. Requires a valid licence and Magento's cron running.

[Cron Schedule] A cron expression (min hour day month weekday). Default 0 3 * * * = daily at 03:00. Magento's own cron (bin/magento cron:run) must be running for it to fire.

[Save Config] Saves all configuration groups for the section.

3. Using Image Optimizer

3.1 The Optimization Log

The Optimization Log (eTechFlow menu → Image Optimizer → Optimization Log) records every conversion. The summary tiles across the top show the cumulative totals — Images Optimized, Source Total, WebP Total, Saved On Disk and Avg Savings — so you can see the space saved at a glance.

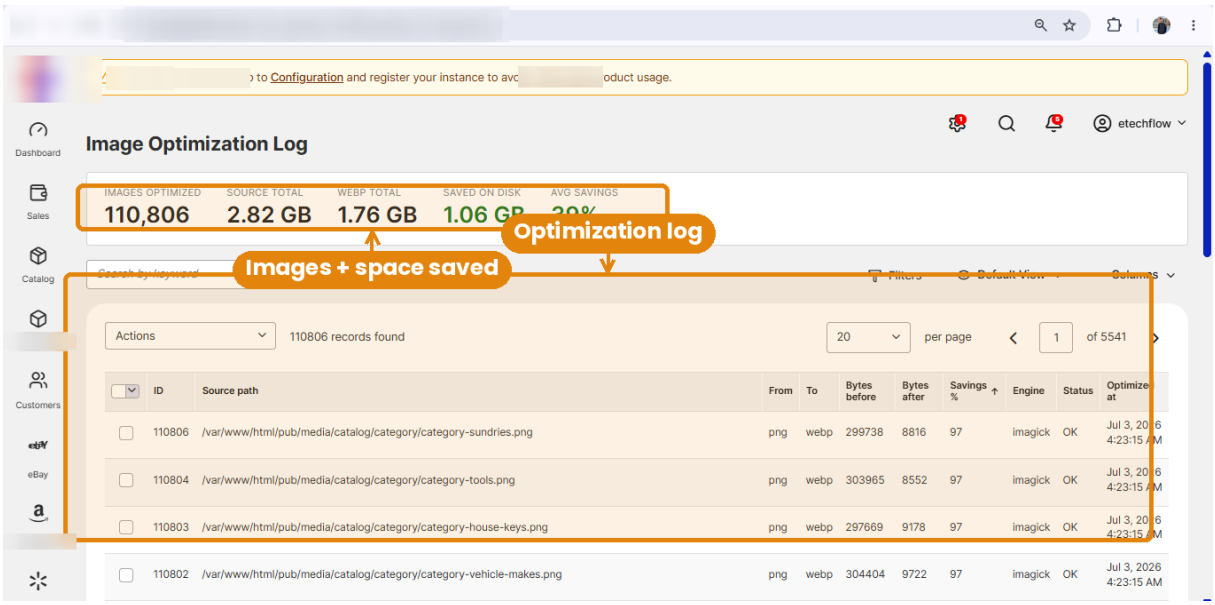

Figure 3 — The Image Optimization Log: summary tiles (here 110,806 images optimized, 1.06 GB saved, 39% average) and the per-image records grid.

Each grid row shows the conversion: ID, Source path, From / To formats (e.g. png → webp), Bytes before, Bytes after, Savings %, the Engine used (e.g. imagick), Status, and the Optimized-at timestamp.

[Search by keyword] Filter the grid by source path or other text.

[Filters / Default View / Columns] Narrow records by field values, save grid views, and show/hide columns.

[Actions] Apply a mass action to selected rows — including deleting log entries and restoring the original images from their WebP conversions.

3.2 Bulk optimize with the CLI

To convert your whole existing catalog in one pass, run the bulk command in your Magento project root (it walks the directories enabled under Image Coverage):

```
php bin/magento etechflow:io:optimize
```

You can re-run it any time; already-converted images are skipped. New/changed images are then kept current automatically by the on-the-fly and cron options in section 7.3.

3.3 Storefront result (WebP delivery)

To confirm images are being served as WebP, open a storefront page, press F12 for DevTools, go to the Network panel and filter by Img. Reload the page and check the image requests.

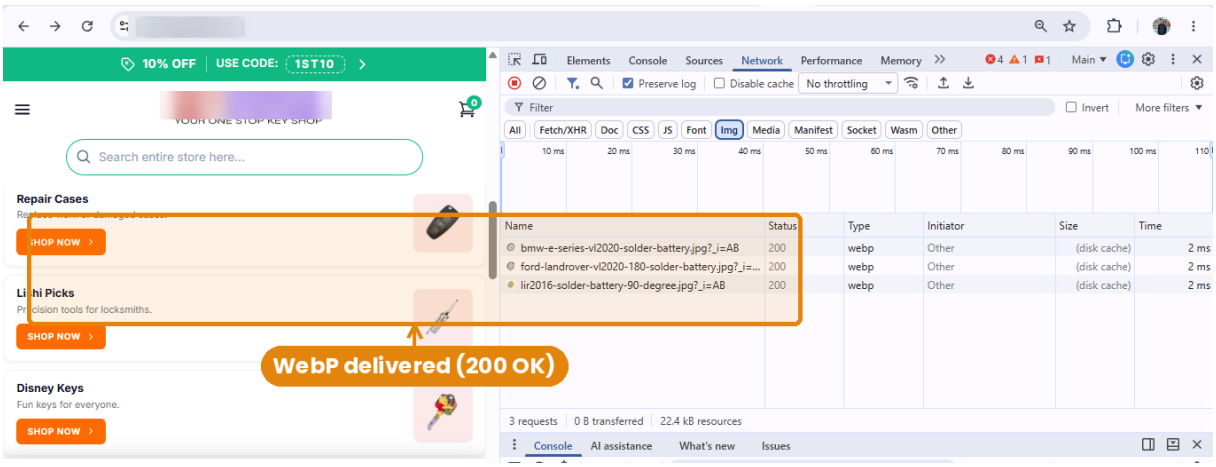

Figure 4 — DevTools Network panel on the storefront: product/category image requests resolve to Type "webp" (Status 200), confirming the conversion and <picture> output are active.

The images look identical to shoppers, but the browser downloads smaller WebP files — faster pages and a better Core Web Vitals score.

4. Support & Contact

For help with Image Optimizer, licensing, or installation, contact eTechFlow support. We respond to licence and configuration questions and can assist with setup on request.

Email: support@etechflow.com

Website: https://etechflow.com