
amasty

For more details see how Google Page Speed Optimizer extension works.

Guide for Google Page Speed Optimizer for Magento 2

Improve your store performance with the Google Page Speed Optimizer for Magento 2. Minify code and optimize images without quality loss.

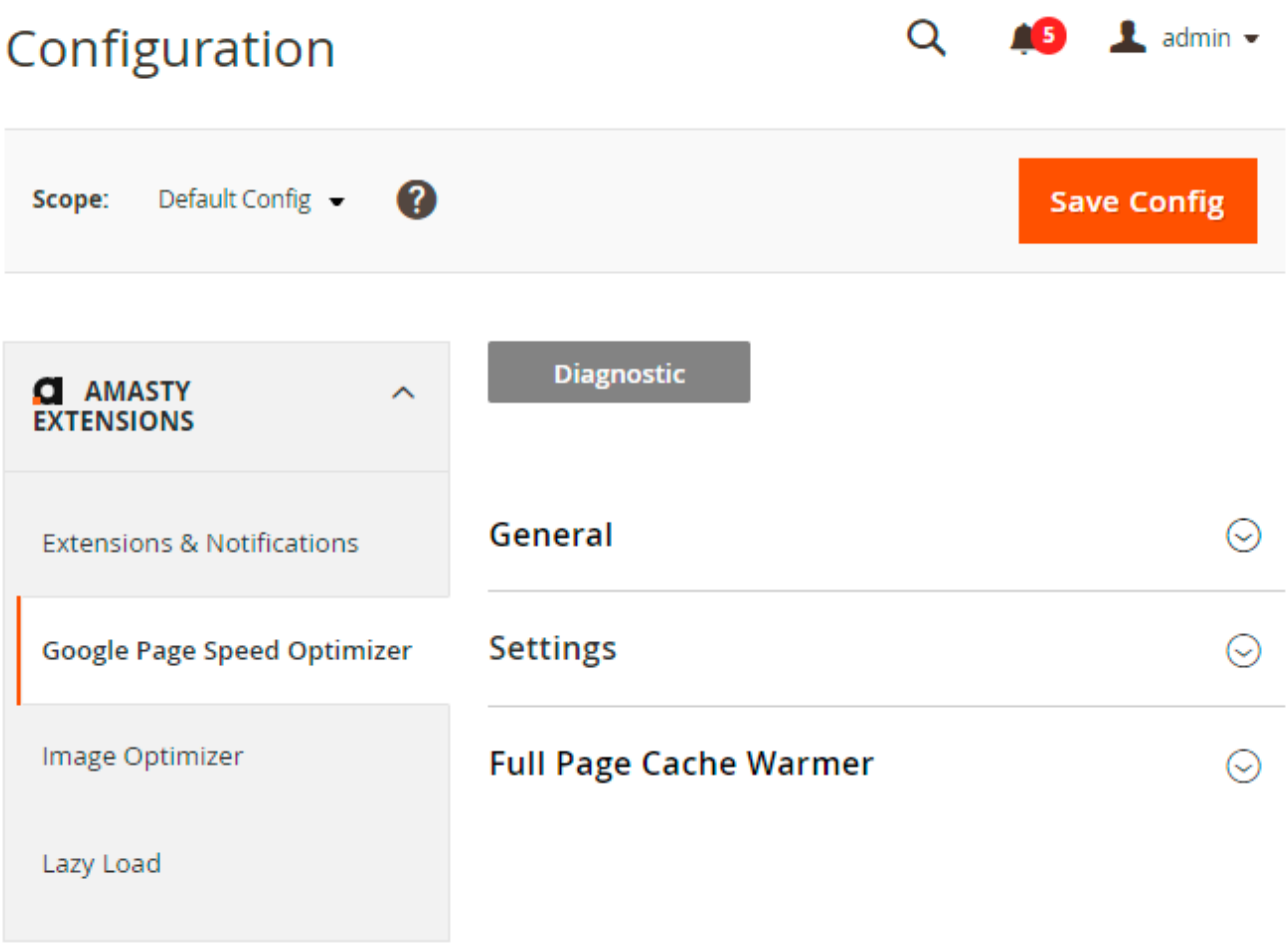
- Automatically reduce images size
- Optimize your database performance
- Minify website code structure
- Improve your webstore pages performance
- Raise your store score in Google PageSpeed Insights up to 99

NEW! Now the extension is compatible with Varnish. You can also add compatibility with 3-rd party extensions which use AJAX.

Find out the general info on how to improve each Google PageSpeed value in our FAQ section.

Diagnostic & Structure

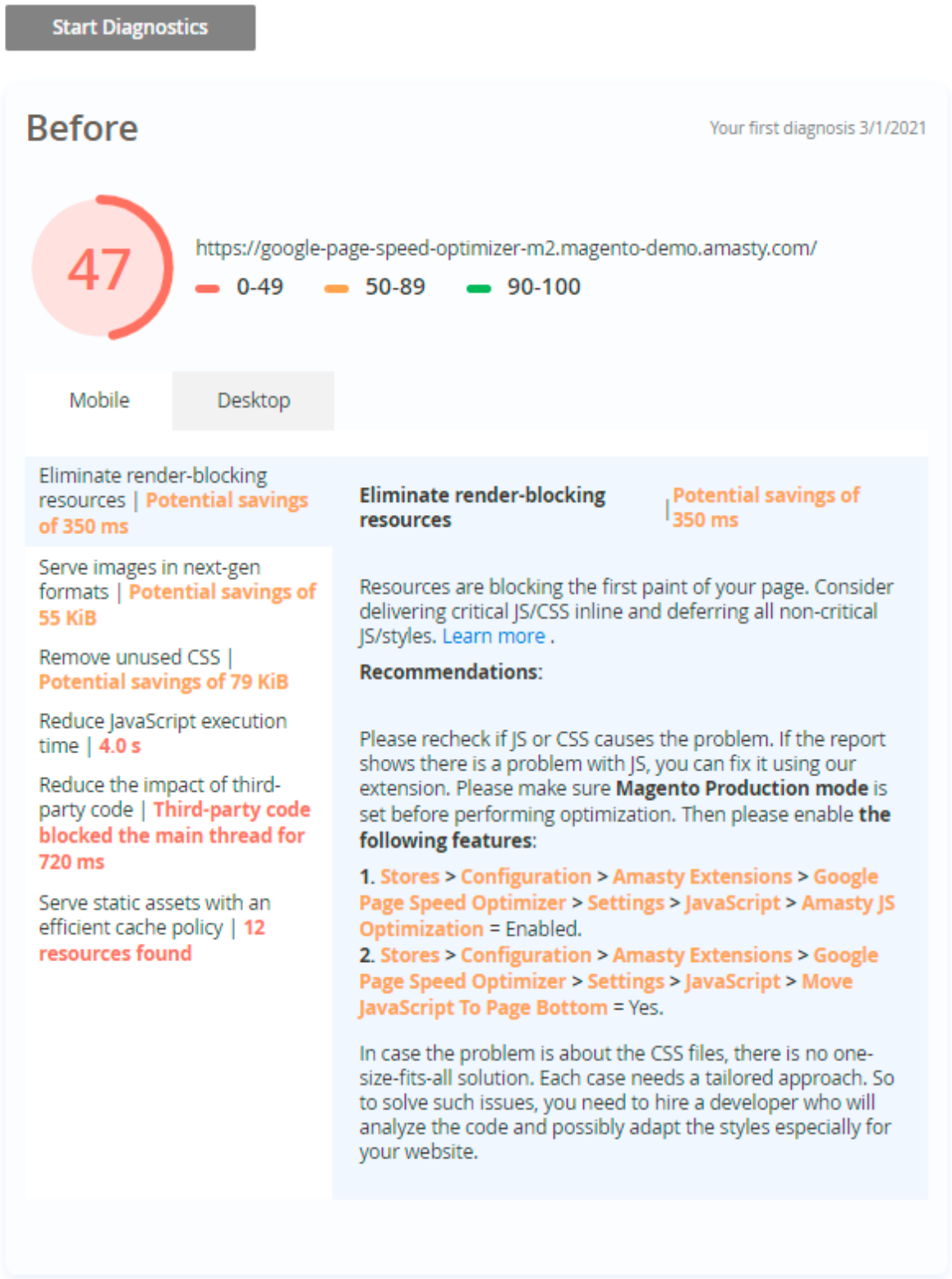
Google Page Speed Optimizer consists of 3 modules: code optimization, image optimization and lazy load. All these extensions should be configured separately. To do this, go to **Stores → Configuration → Amasty Extensions**.



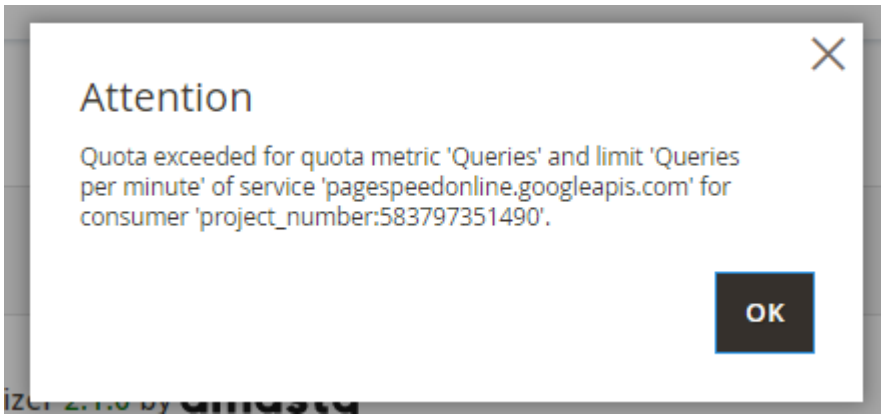
Before starting the optimization, check your website current Google score right from the admin panel using **Diagnostic** option.

Please, mind that the extension automatically gets the URL of a real webstore to check. But if you install the module on a test instance, specify the required URL here:

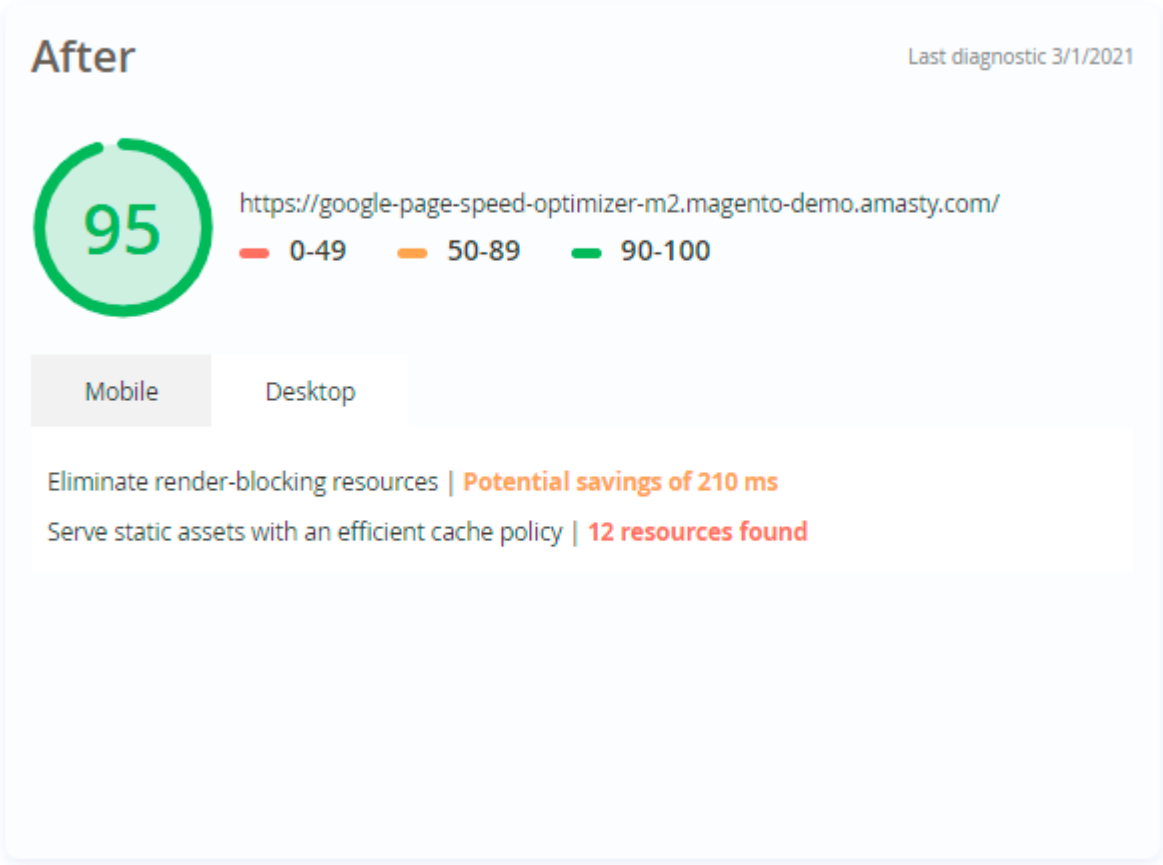
You may see the desktop and mobile page speed scores separately. The informative block also contains Google recommendations.



Note that Google has a limit for checking. If you face the issue below, please wait for some time and restart diagnostics.



Follow the recommendations and check the result once more. You will see the result in the **After** tab.



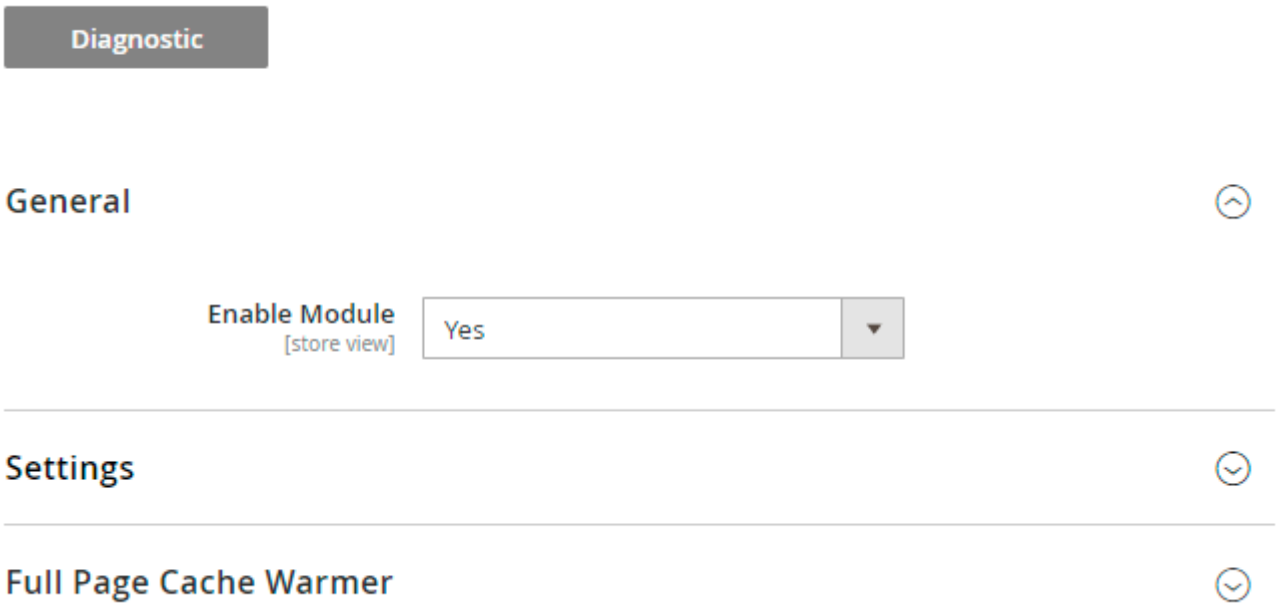
Before starting configuration, please, set **Magento Production** mode instead of Developer or Default. The Production mode is required to get the best performance of this extension.

Google Page Speed Optimizer

This part of the extension contains code optimization and some other settings to make your webstore load faster.

General

Expand the **General** tab.

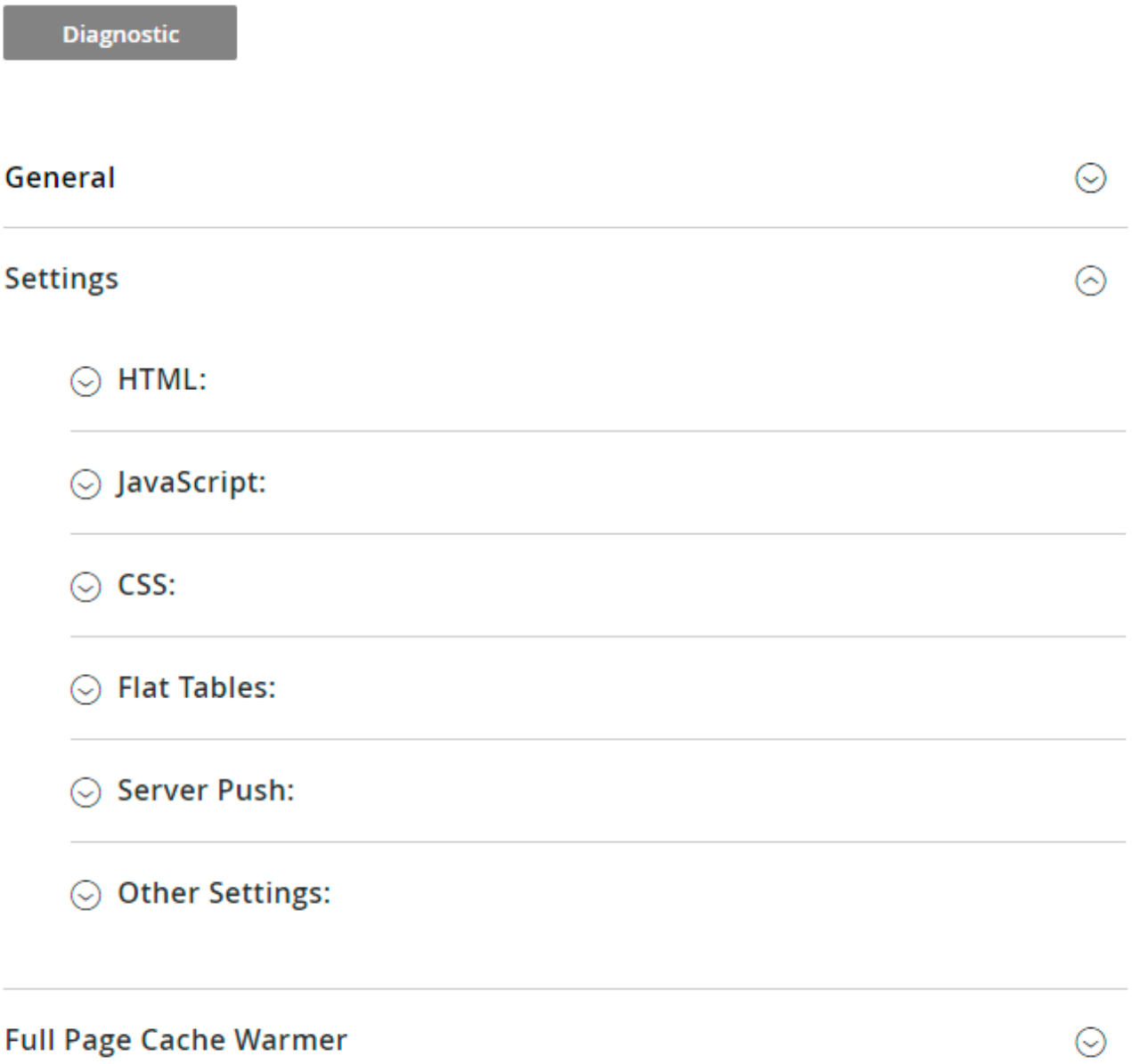


Enable Module - set to *Yes* to activate the module.

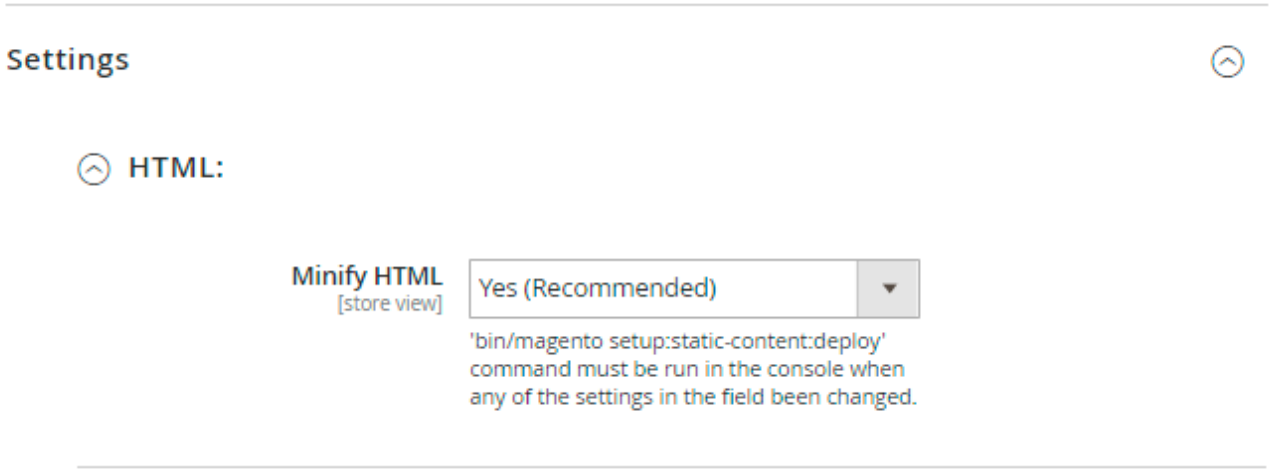
The extension also works on *Magento cloud*.

Settings

Expand this tab to reduce HTML, CSS and JS code size. The extension removes all unnecessary line breaks, tabs, comments and spaces. Also, it includes some advanced settings, such as server push.



HTML Minification



Minify HTML - set the option to *Yes* to enable HTML minification.

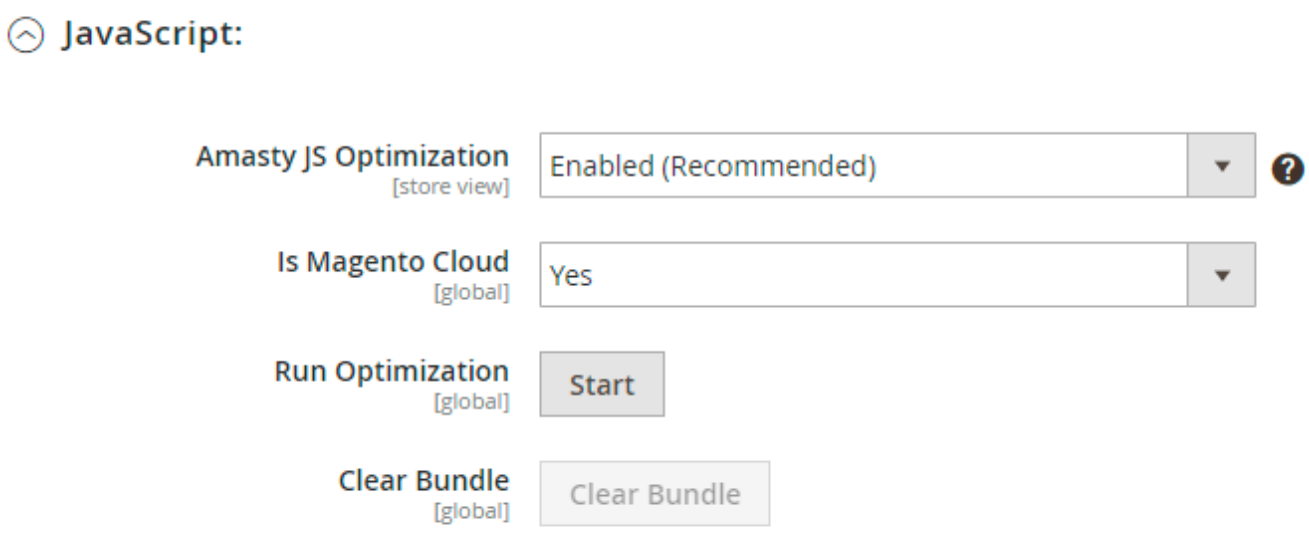
When any of the settings in the field have been changed, execute the following command:

```
bin/magento setup:static-content:deploy
```

JavaScript

In this tab you can create an advanced bundle of JS using **Advanced JS Bundling** feature and JS Minification that will significantly reduce the page size in kilobytes and increase the page load speed. Google rank for both mobile and desktop will be improved.

You will need to use CLI commands to run through the optimization. Please make sure that you have access to the console and know how to do it.
Please create a full backup of your Magento instance and follow the instructions that are displayed in the **Run Optimization** section.



To run the optimization, set the **Amasty JS Optimization** option to *Enabled*.

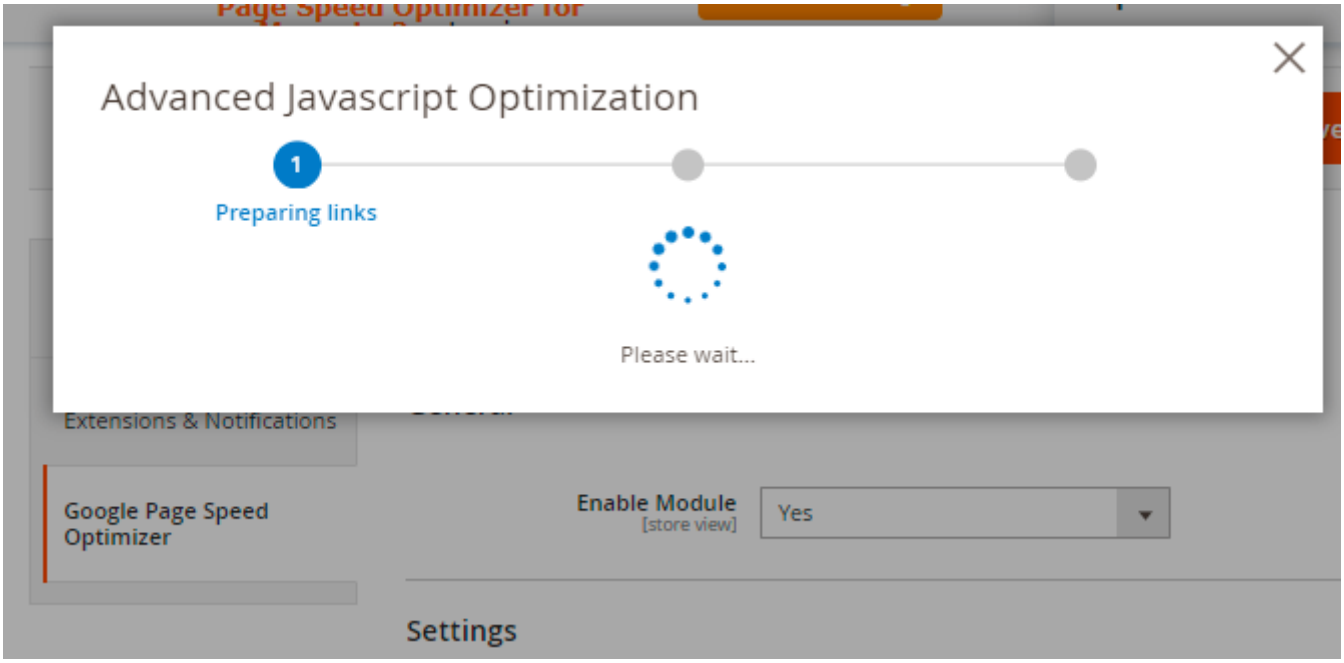
Is Magento Cloud - set to *Yes* if you use Magento Cloud hosting platform.

Click **Start**.

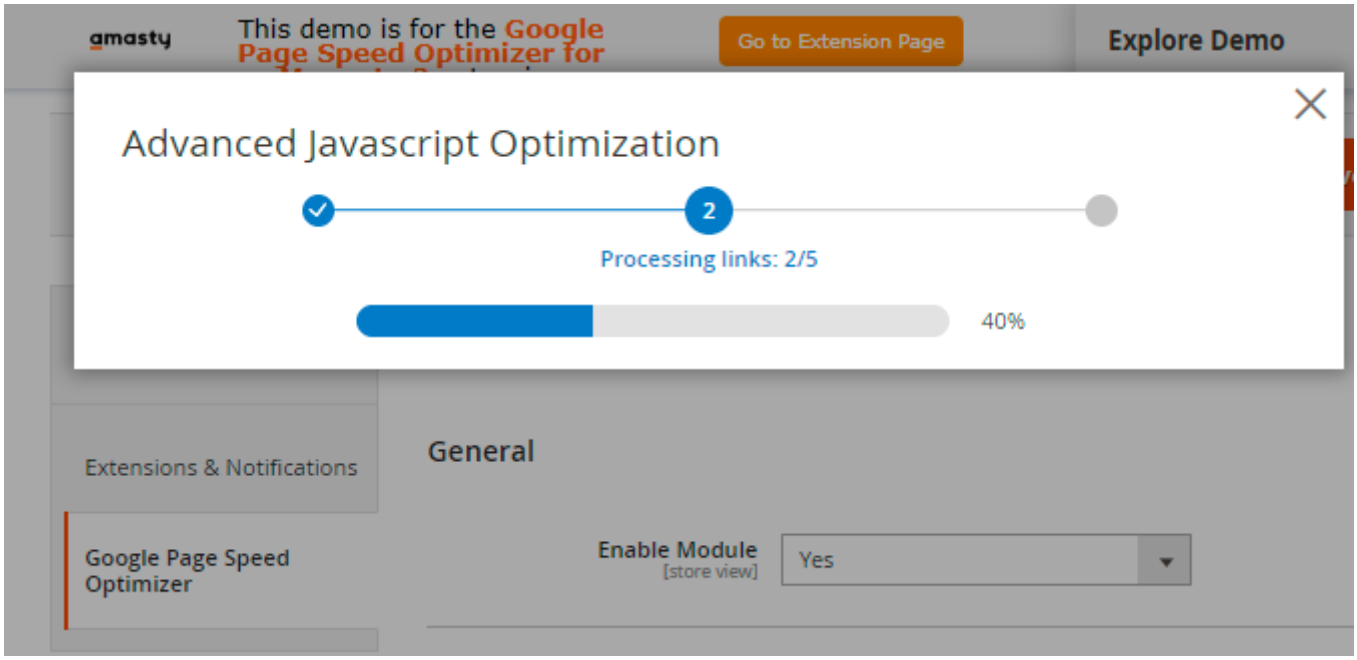
Easily **Clear Bundle** if needed.

Automatic Optimization

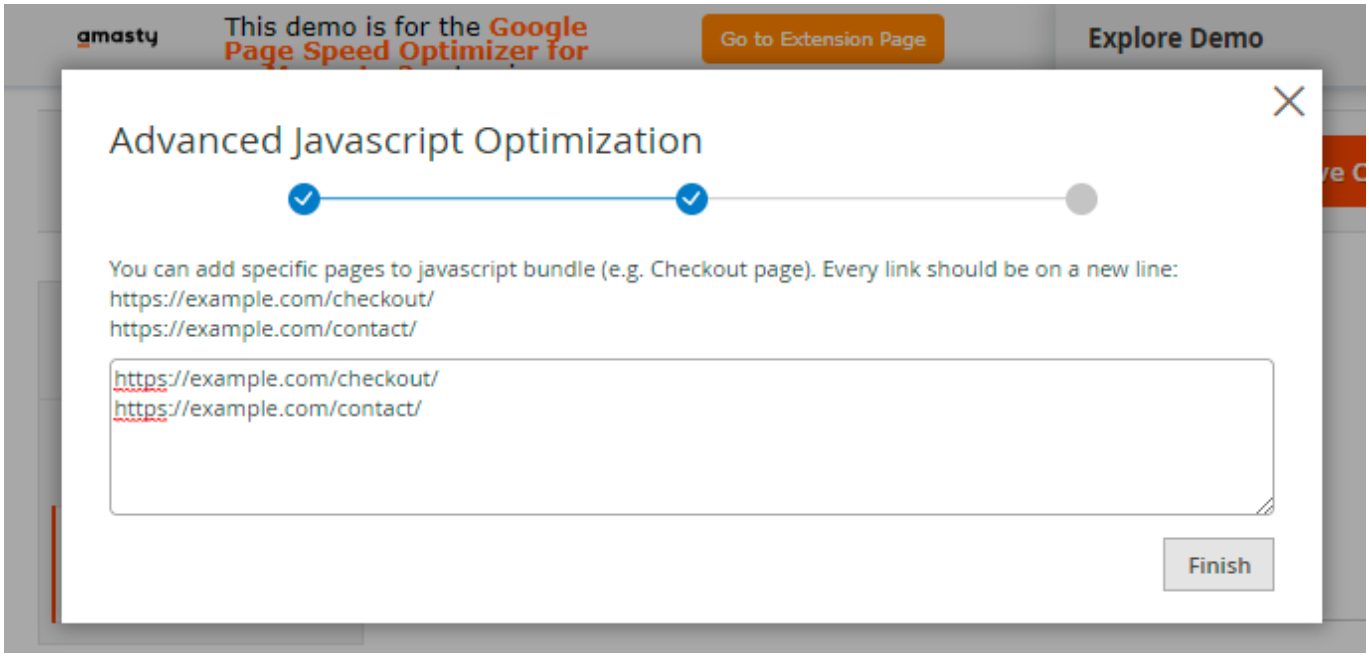
Step 1. After clicking the **Start** button, the extension starts preparing all the links for optimization.



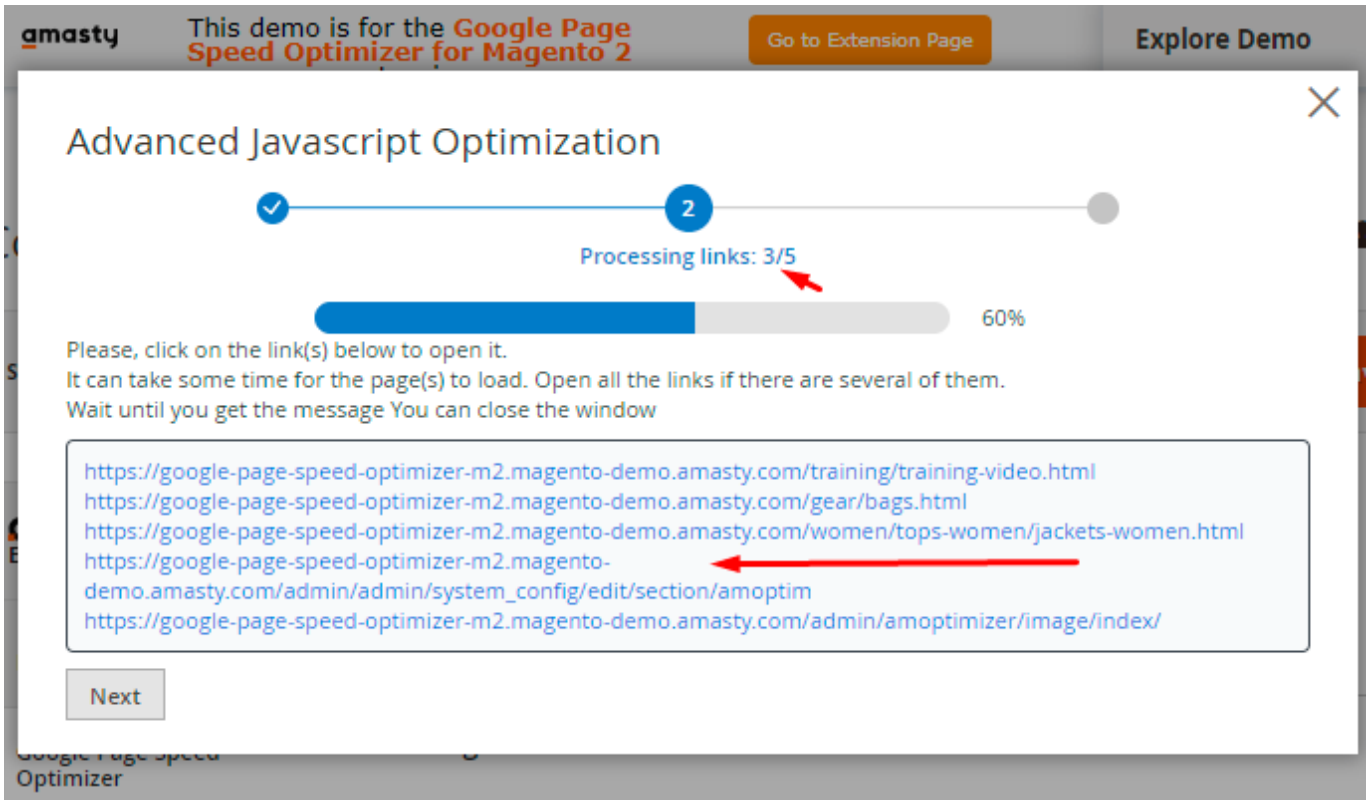
Step 2. Links processing progress is displayed on the bar.



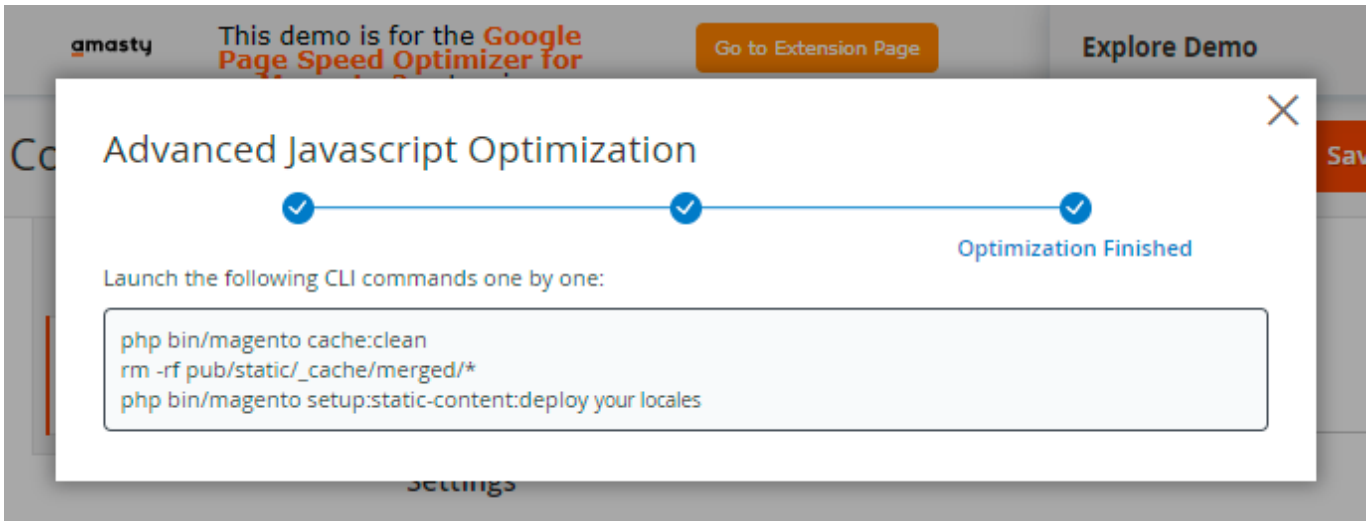
When all the links from the bundle are successfully processed, you can add specific pages to JavaScript bundle. Keep in mind that each link should be placed on a new line.



If any errors are found, the processing will be stopped and the error message will appear. You may check the particular link that caused a failure. For example, 3/5 links has been successfully processed and then the message appears. It means that the 4th link contains some problems. Easily click on it to check if it is valid or not and correct the link or page itself if needed.



Step 3. When optimization is finished, you will see the appropriate message and a list of commands that should be executed.



'Is Magento Cloud' Usage

Previously, the setting 'Is Magento Cloud' was created for Magento Cloud usage. But, to avoid manual deployment, you can also use this setting even if you don't use the Cloud. Follow the steps below to set up an automatic deploy:

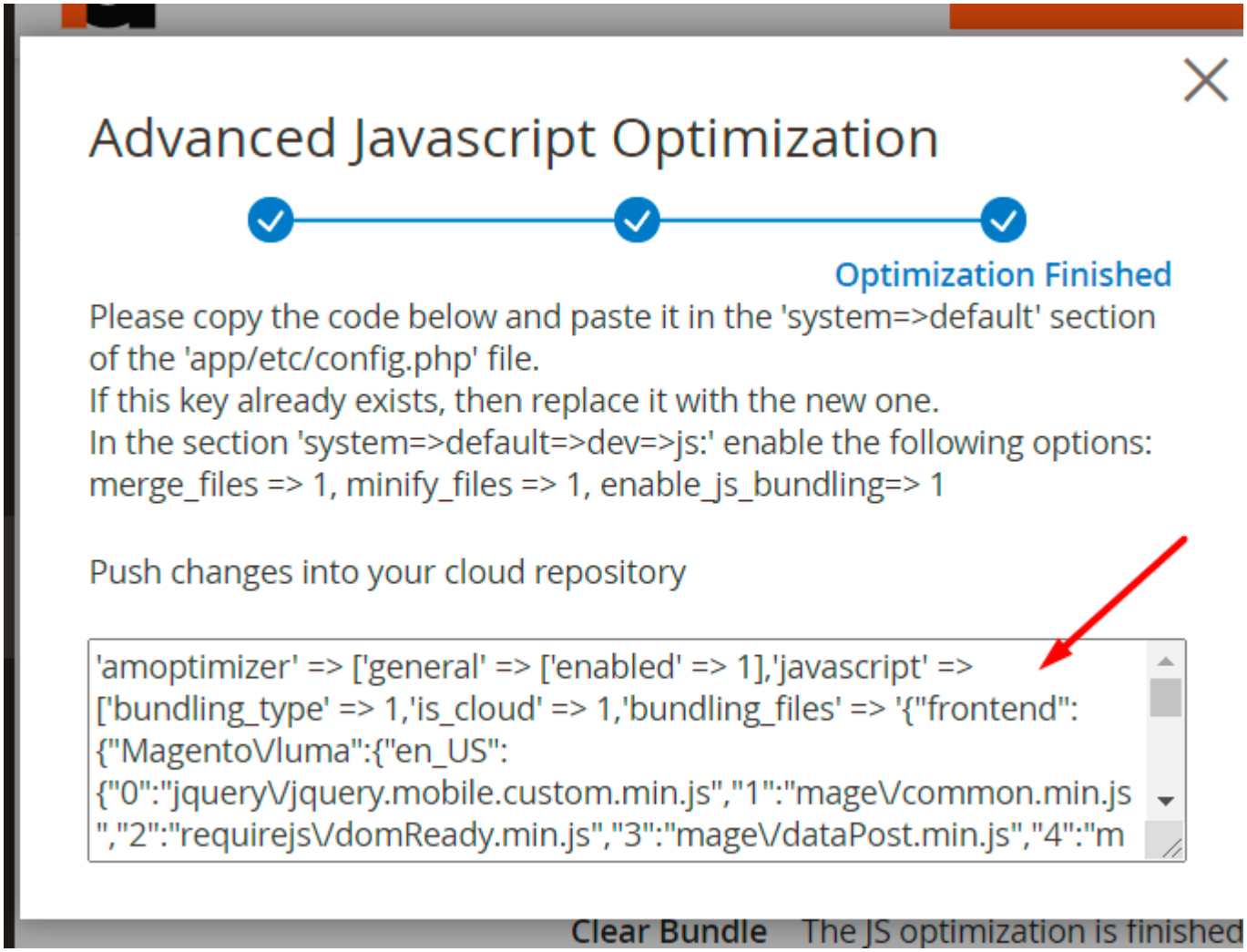
1. Set 'Is Magento Cloud' option to '*Yes*'

2. Find the **app/etc/config.php** file. There you will see the list of installed modules. To create an optimization file, you will need the following massive:

```
'system' => [
        'default' => [
            'dev' => [
                'template' => [
                    'minify_html' => 1
                ],
                'js' => [
                     'merge_files' => 1,
                     'enable_js_bundling' => 1,
                     'minify_files' =>1
                ]
            ]
        ]
    ]
```

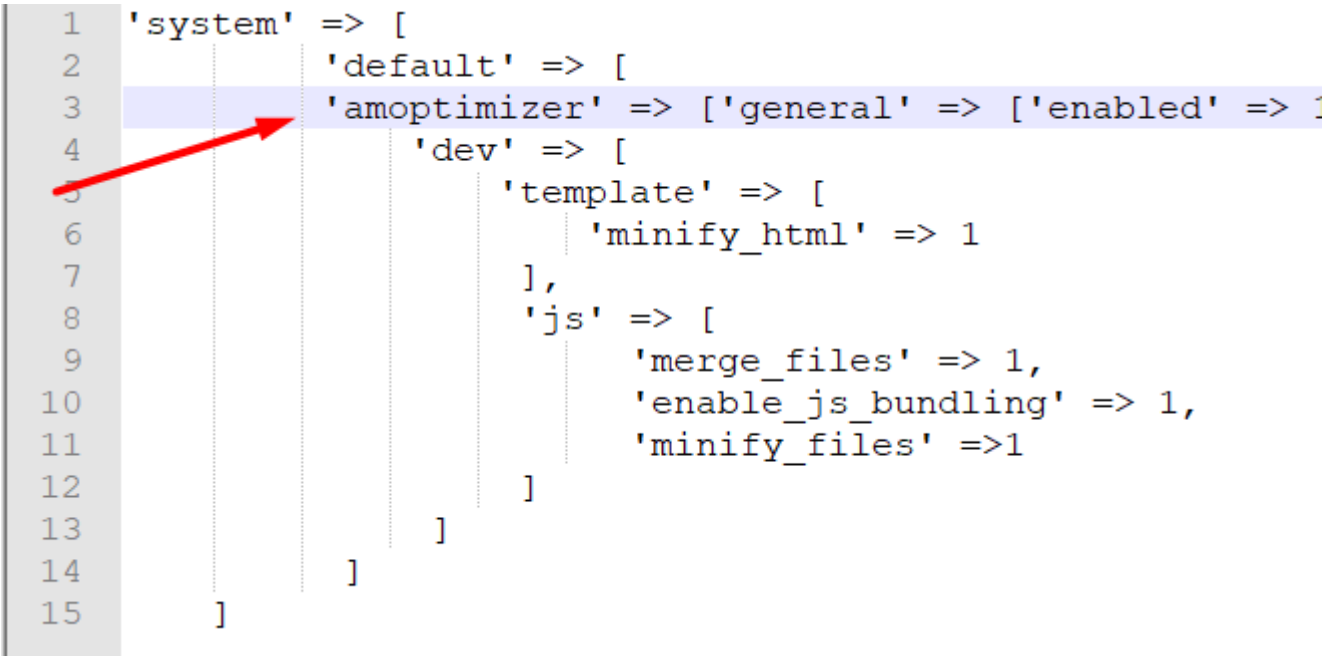
3. Go back to your admin panel and click **Run Optimization → Start**.

4. The extension will generate the code that should be put into cloud repository:



You can see the detailed instructions for generating in the Advanced Javascript Optimization.

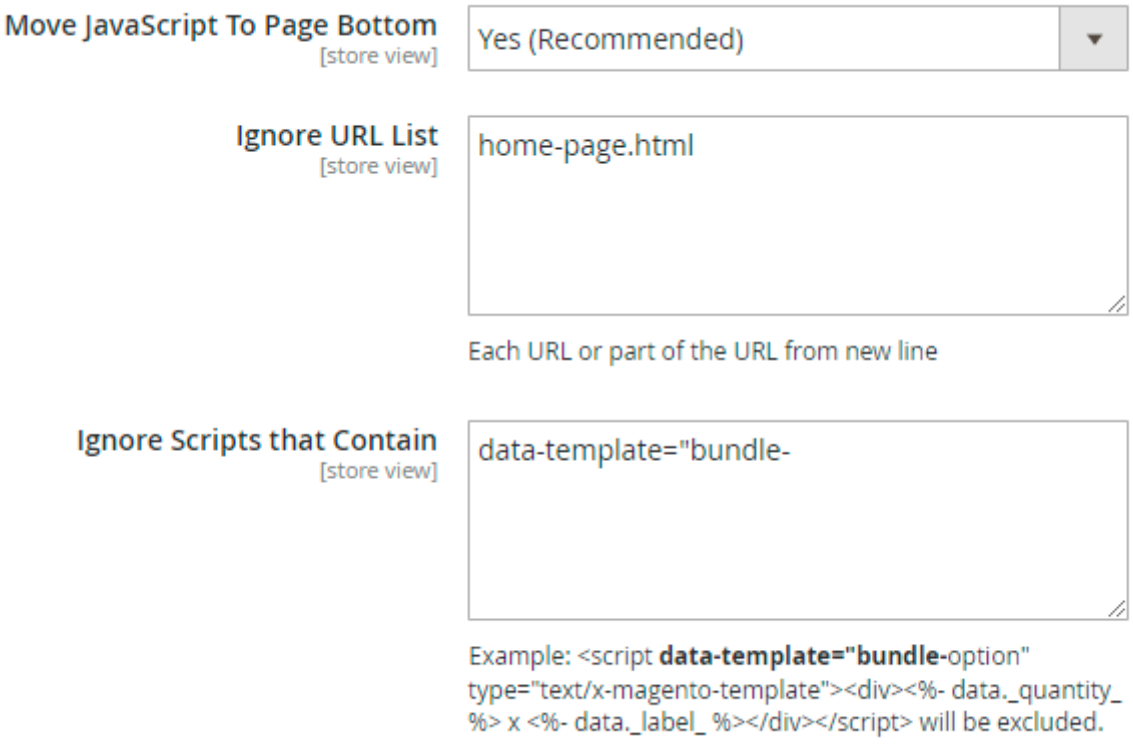
5. Copy the generated code and paste it after 'default' arrow in **app/etc/config.php**:



```
'system' => [
        'default' => [
        'amoptimizer' => ['general' => ['enabled' => 1
            'dev' => [
                'template' => [
                    'minify_html' => 1
                ],
                'js' => [
                    'merge_files' => 1,
                    'enable_js_bundling' => 1,
                    'minify_files' =>1
                ]
            ]
        ]
    ]
```

6. Save the *config.php* file.

Additional JS Options



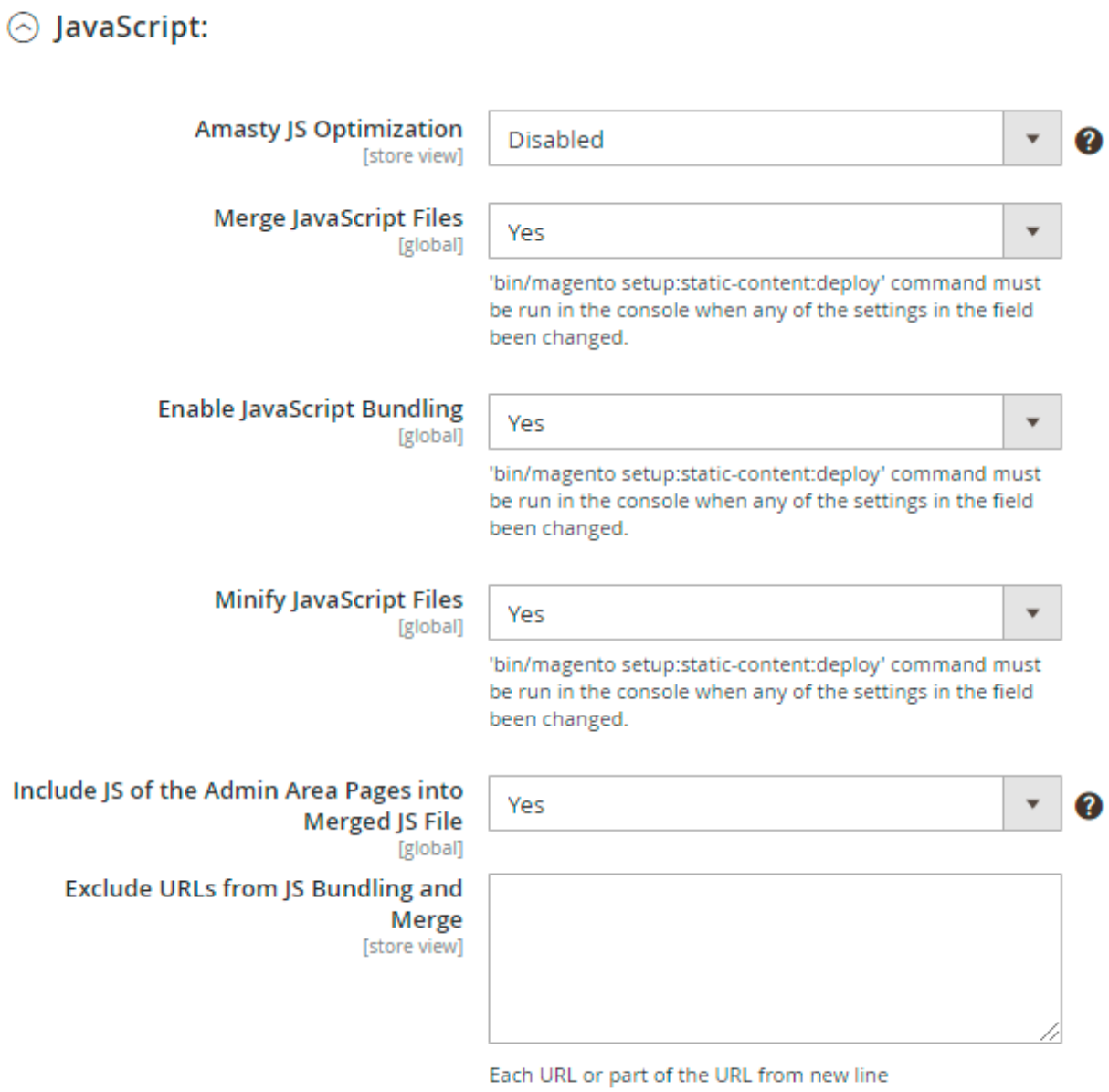
Move JavaScript To Page Bottom option helps to load all the important content before loading JS. The functionality improves pages loading time.

Ignore URL List - provide URLs of the pages on which JS shouldn't be moved to bottom.

Ignore Scripts that Contain - you may also specify particular scripts and their parts if you want them to exclude from moving.

The sample of *Moving to Footer* feature:

If **Amasty JS Optimization** is *Disabled*, you can customize the way of code optimization. Besides Moving JS To Page Bottom, there are 4 additional options to reduce your JS code. You can also exclude particular URLs and scripts.



Merge JavaScript Files - select *Yes* to put all JS files into one. As a result, the number of queries will be reduced.

Enable JavaScript Bundling - enable this option to combine all JS files into few bundles and download them for each page. This helps to save time by reducing the number of server requests.

Notice that as browsers download the bundles synchronously, page rendering is blocked until all bundles finish downloading.

Minify JavaScript Files - set to *Yes* to enable JS minification.

Include JS of the Admin Area Pages into Merged JS File - enable this option if you want to optimize the admin area as well.

Optimizing the admin area pages sometimes lead to malfunctions and doesn't affect the Google rating.

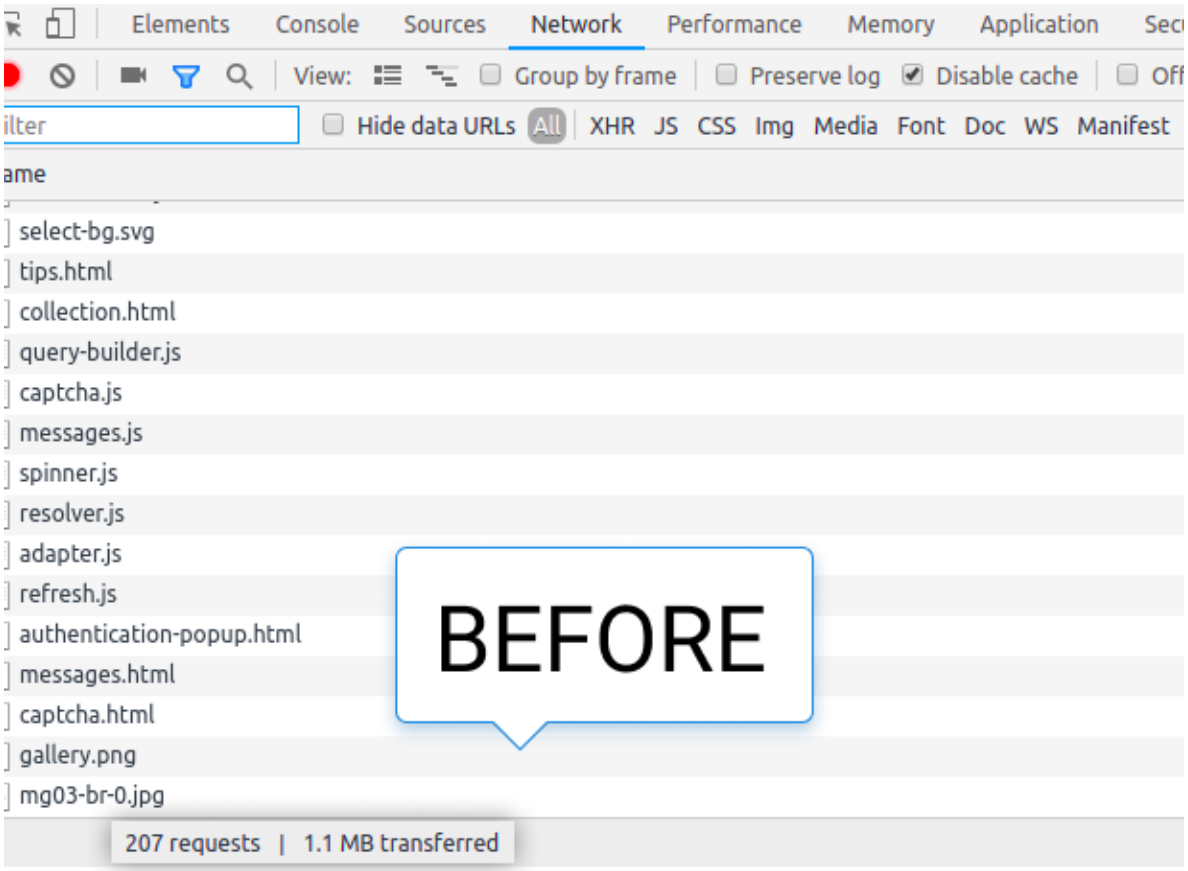
Exclude URLs from JS Bundling and Merge - specify particular URLs that shouldn't be merged or minified. Start each URL or part of the URL from new line.

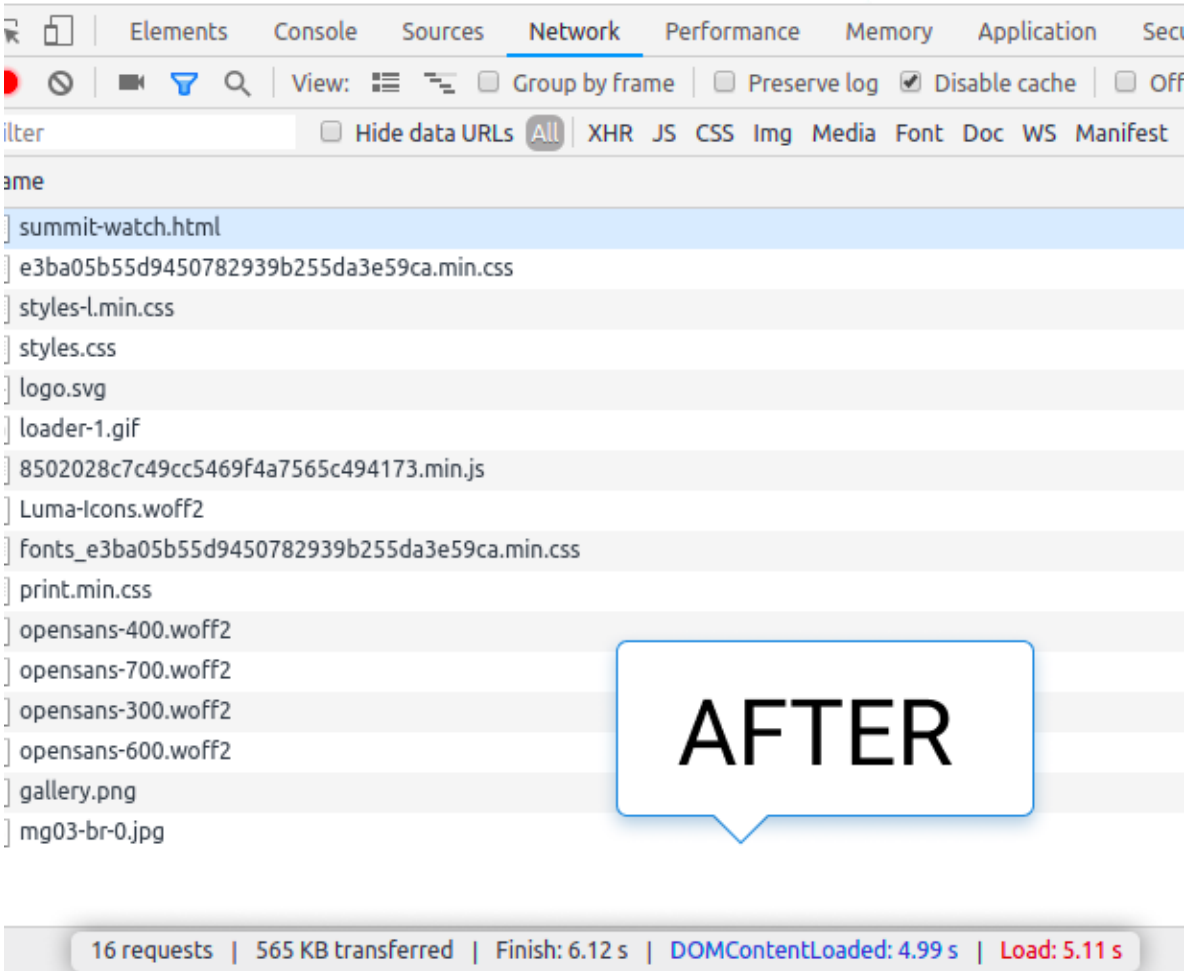
Don’t forget to run the command

```
bin/magento setup:static-content:deploy
```

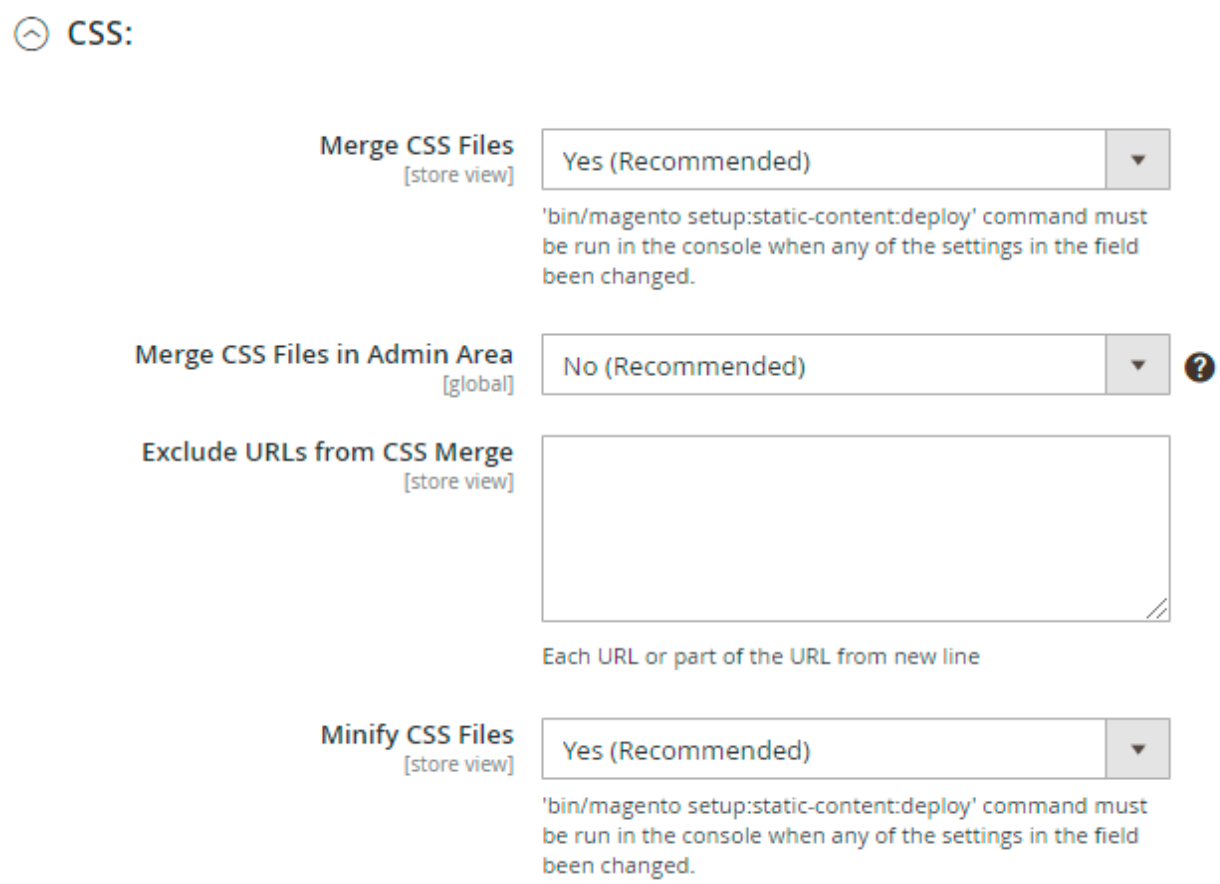
in the console after any changes in this block of settings have been made.

This is how the bundling feature works:


Elements
Console
Sources
Network
Performance
Memory
Application
View:
Group by frame
Preserve log
Disable cache
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
select-bg.svg
tips.html
collection.html
query-builder.js
captcha.js
messages.js
spinner.js
resolver.js
adapter.js
refresh.js
authentication-popup.html
messages.html
captcha.html
gallery.png
mg03-br-0.jpg
207 requests | 1.1 MB transferred
BEFORE


Elements
Console
Sources
Network
Performance
Memory
Application
View:
Group by frame
Preserve log
Disable cache
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
summit-watch.html
e3ba05b55d9450782939b255da3e59ca.min.css
styles-l.min.css
styles.css
logo.svg
loader-1.gif
8502028c7c49cc5469f4a7565c494173.min.js
Luma-Icons.woff2
fonts_e3ba05b55d9450782939b255da3e59ca.min.css
print.min.css
opensans-400.woff2
opensans-700.woff2
opensans-300.woff2
opensans-600.woff2
gallery.png
mg03-br-0.jpg
16 requests | 565 KB transferred | Finish: 6.12 s | DOMContentLoaded: 4.99 s | Load: 5.11 s
AFTER

CSS

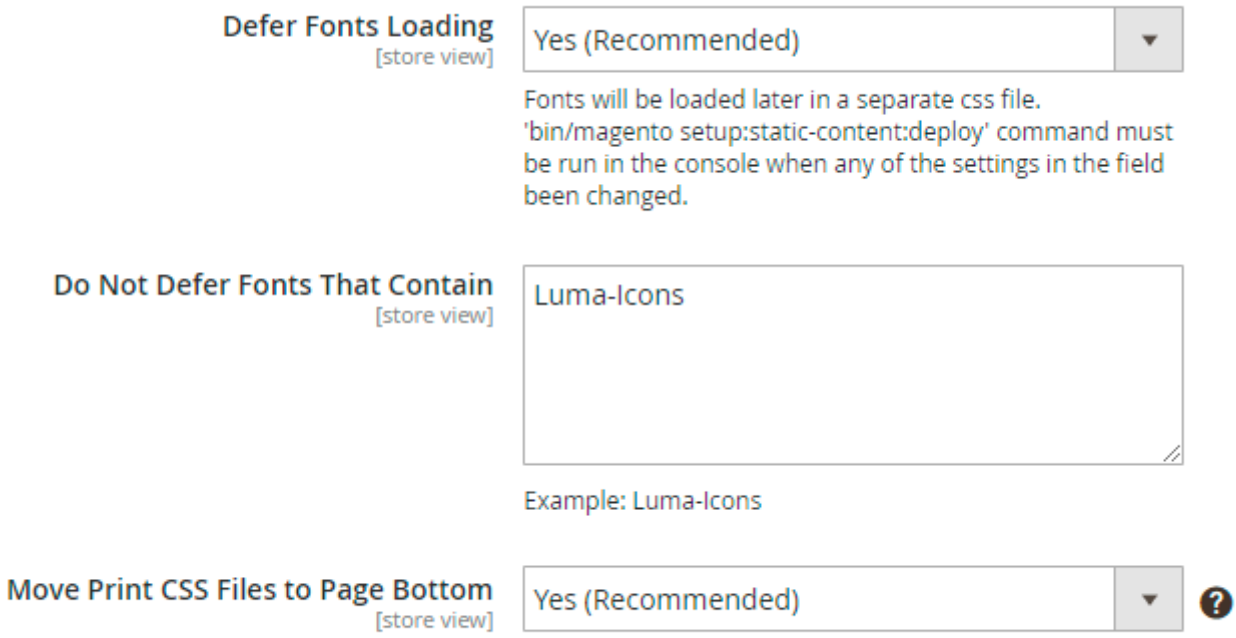


Merging CSS Files works the same way as merging JS files.

Merge CSS Files in Admin Area - set *No* to forbid CSS files merging in the admin area. It won't affect your Goggle rating.

Exclude URLs from CSS Merge if needed.

Minifying CSS Files works the same way as minifying JS and HTML.



If you set **Defer Fonts Loading** to *Yes*, the browser will load the page with system fonts and replace them with your fonts when they are loaded. It makes store pages load faster.

The following command must be run in the console when any of the settings in the field been changed:

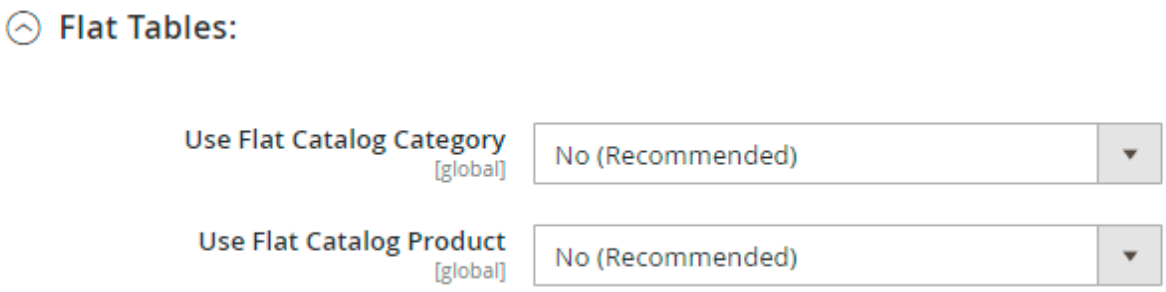
```
bin/magento setup:static-content:deploy
```

You may exclude some fonts from deffering in the **Do Not Defer Fonts That Contain** option.

As in the JS tab, you can also **Move Print CSS Files to Page Bottom** to make your page render faster.

Flat Tables

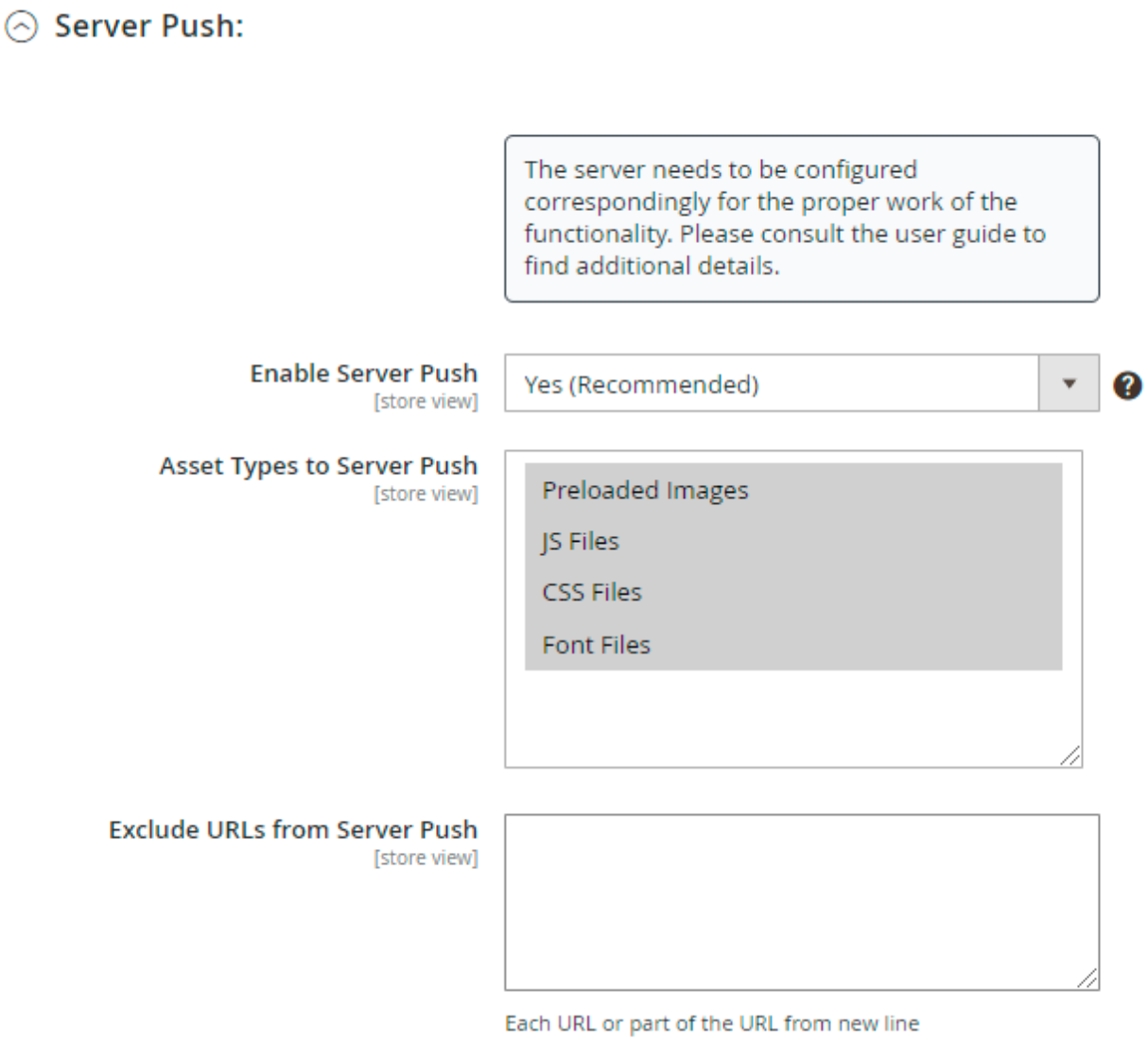
Flat Tables allow storing all the information about a category or a product. Such tables help to shorten the query in the database.



Keep in mind that flat catalog usage is not recommended by Magento documentation. See more details here.

Server Push

A powerful feature of HTTP/2 represents the ability of the server to send multiple responses for a single client request. That is, in addition to the response to the original request, the server can push additional resources to the client, without the client having to request each one explicitly.



Before enabling the option, the server needs to be configured correspondingly for the proper work of the functionality. Please, follow the steps below to configure the server you're using.

Nginx Server

Please, add the following changes to the configuration file which is responsible for Magento:

1. Specify **http2** flag for **listen** parameters

2. Add **http2_push_preload on**; to enable pushes (it should be under **server** scope)

3. Specify max pushes per request in **http2_max_concurrent_pushes 50**

```
server {
    listen 80;
    listen [::]:80;
    listen 443 ssl http2; <-- Add http2 flag
    listen [::]:443;

    http2_push_preload on; <-- Enable preload
    http2_max_concurrent_pushes 25; <-- Setup max pushes per request

    ...
```

Apache Server

1. First, you need to enable the HTTP/2 module.

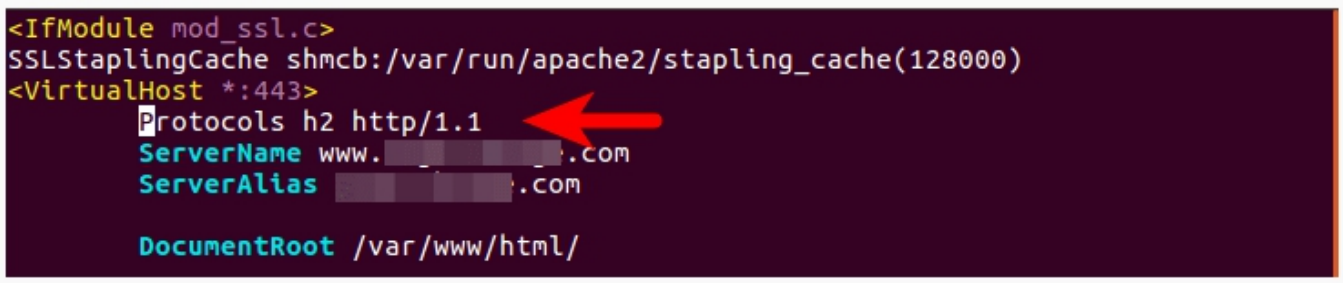
```
sudo a2enmod http2
```

2. Then edit your Apache virtual host file. If you enabled HTTPS with Let's Encrypt, then the SSL virtual host is created as a separate file ending with **le-ssl.conf**.

```
sudo nano /etc/apache2/sites-enabled/your-site-le-ssl.conf
```

3. Put the following directive after the opening **<VirtualHost *:443>** tag.

```
Protocols h2 http/1.1
```



4. Save and close the file. Then restart Apache for the changes to take effect.

```
sudo systemctl restart apache2
```

To make the feature work with **Varnish**, proceed to the Varnish configuration file and remove the following line from it:

```
unset resp.http.Link
```

Module Configuration

Now we can proceed to extension settings.

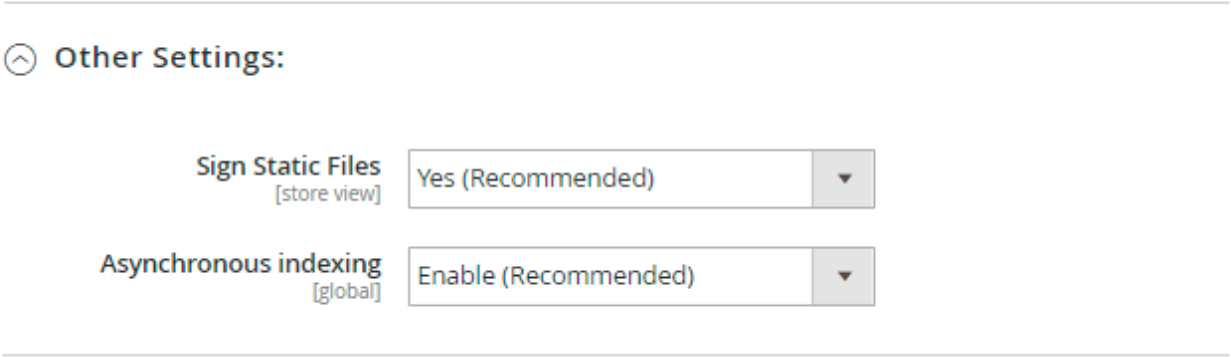
Enable Server Push - enable the option to activate server push.

Asset Types to Server Push - specify the files that will be used for the push.

Exclude URLs from Server Push - provide particular URLs to exclude from the push.

Other Settings

In this tab you can enable some additional settings to make your webstore perform better.



Sign Static Files - activate the option to make the browser automatically update cached values once they are changed.

Asynchronous indexing - enable to distinguish traffic patterns on the database level to avoid conflicts between read and write operations. Order data is placed in a temporary storage and moved to the Order Management grid with no accidents.

Image Optimizer

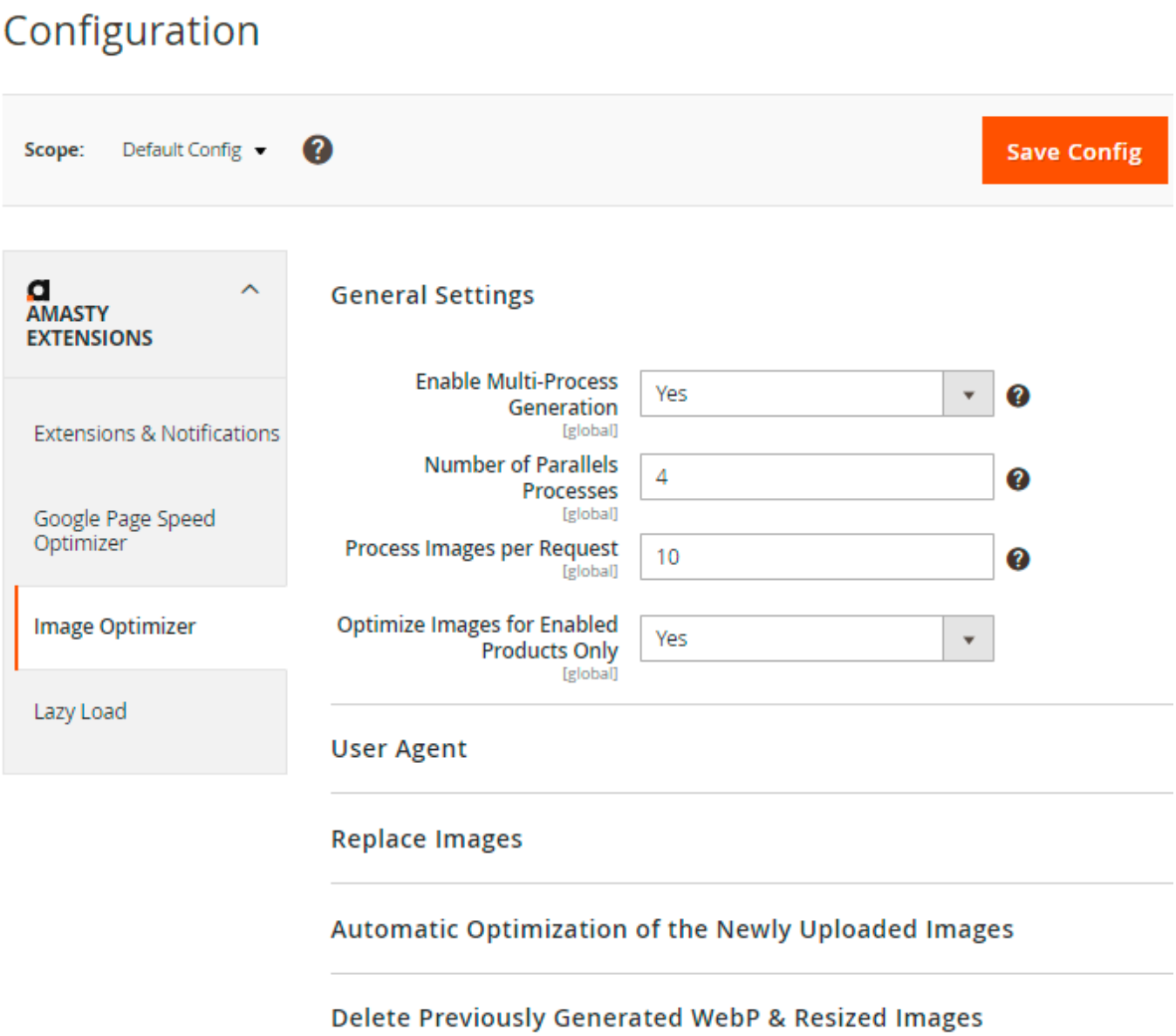
Switch to this tab to configure the automatic image optimization using various flexible settings. The extension allows you to optimize JPEG, PNG and GIF images.

The correct functioning of the extension requires some special software to be installed: JPEG optimization tool / PNG optimization tool / GIF optimization tool / WEBp tool. If you are not sure how to install the necessary software on the server correctly, please contact your administrator or Amasty

support team.

General Settings

Expand the **General Settings** tab to configure the main settings.



Enable Multi-Process Generation - select *Yes* to speed up images optimization.

Multi-Process Optimization significantly boosts the image optimization speed. But it generates extra load to the server as well. We recommend first to run the test image optimization process with the 'multi-process' *ON* and monitor your server performance.
The 'Multi-Process Generation' feature needs the php extension '**pcntl**' to be installed on the server. If you enable the feature and no performance boost happens, please ask your hoster/system administrator to check if the 'pcntl' extension installed.

Number of Parallels Processes - specify the number of processes to be performed at once. The

more parallel processes are set, the faster the image optimization process is, as well as the higher is the server load.

Process Images per Request - enter the number of images you want to be shown instantly, per one request.

You may also optimize images in bulk using a console command (here **4** is the number of threads):

```
bin/magento amasty:optimizer:optimize -j 4
```

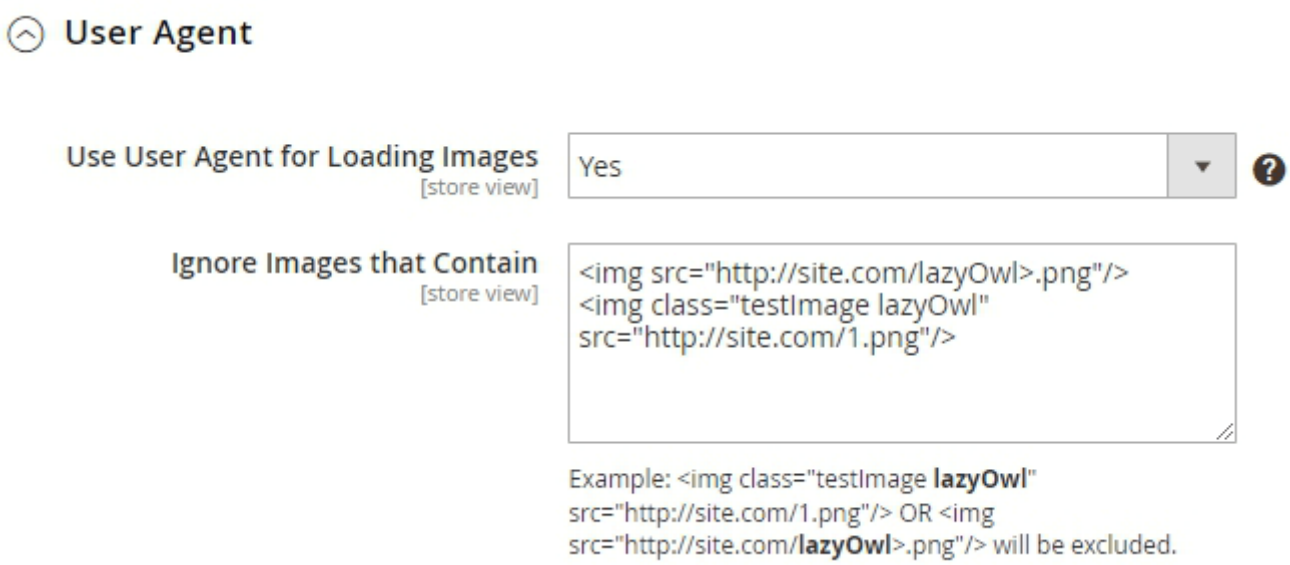
Moreover, you can specify the particular pattern (**i** is the id of the pattern):

```
bin/magento amasty:optimizer:optimize -j 4 -i 2
```

Optimize Images for Enabled Products Only - enable this option if you want to create smaller copies for the images of enabled products only. It helps to save extra space on your server and speed up optimization.

User Agent

Expand the **User Agent** tab to adjust settings for image loading according to the customer's browser user agent.

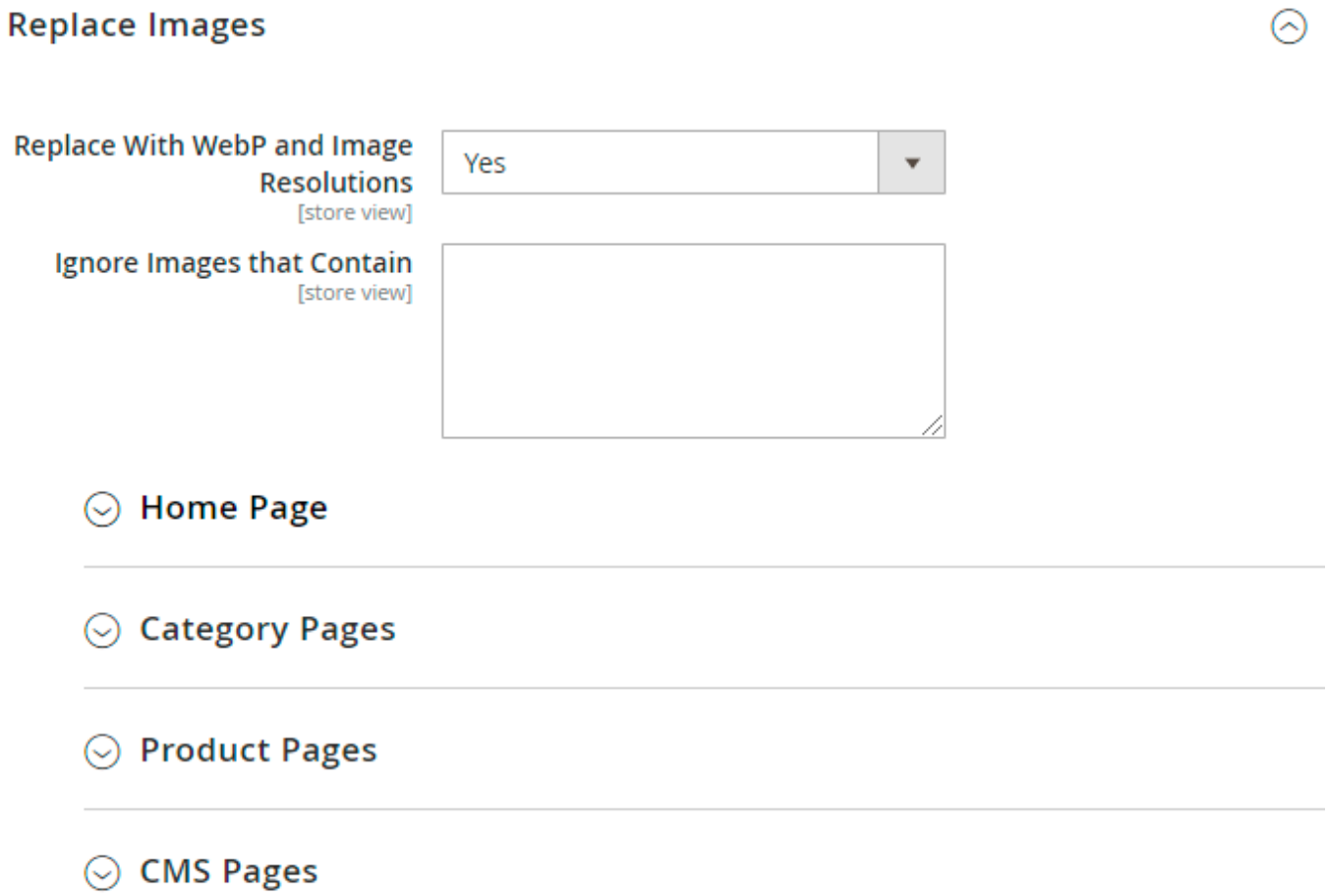


Set the **Use User Agent for Loading Images** to **Yes** to enable fetching appropriate image sizes based on the user agent data, obtained from the browser.

In the **Ignore Images that Contain** you can specify certain images to be excluded from the loading based on the user agent info. Such images will be delivered on the users' devices in the original size.

Replace Images

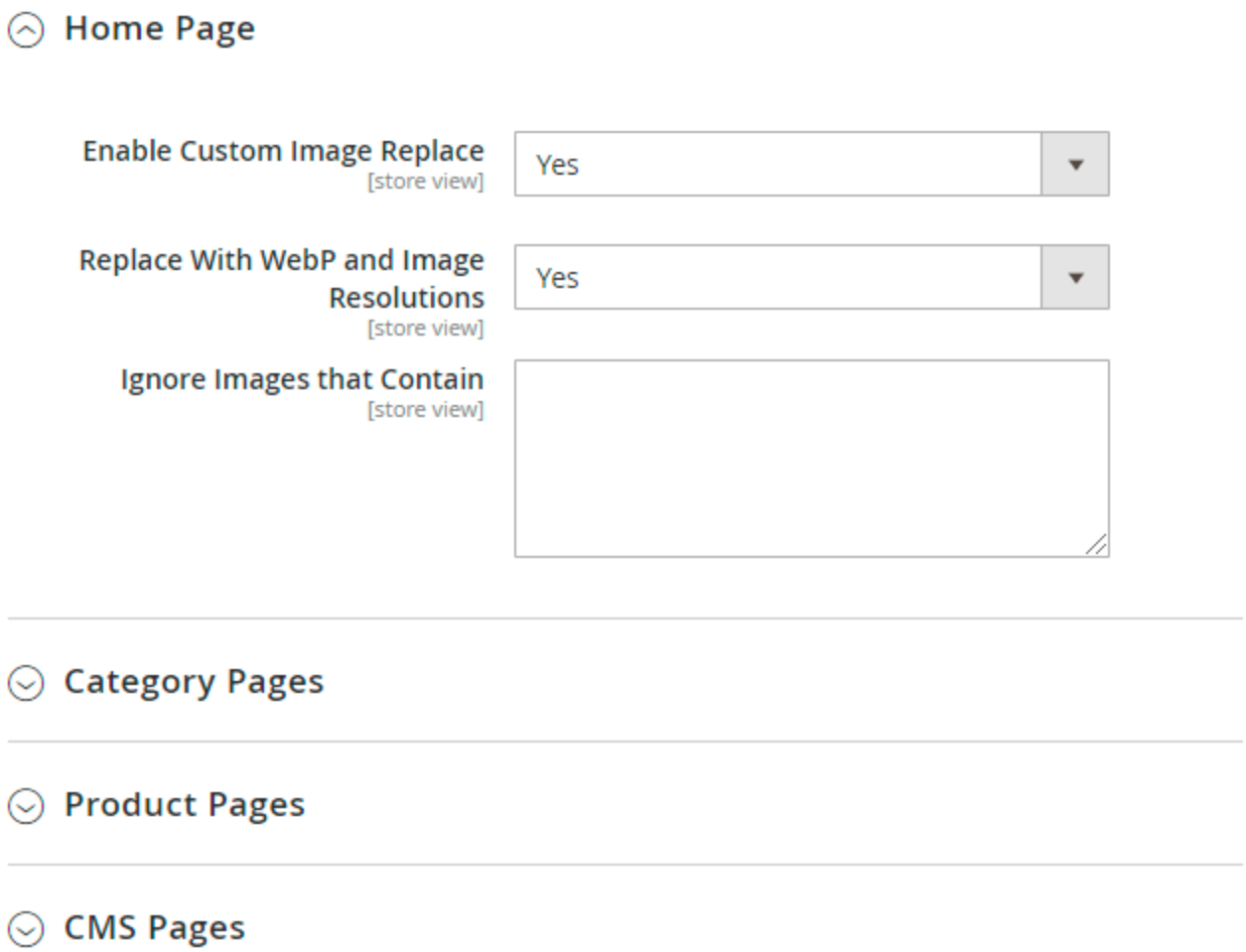
Choose the strategy of image optimization. Here you can implement the options selected in the Automatic Optimization of the Newly Uploaded Images section.



Replace With WebP and Image Resolutions - choose Yes to replace all existing images with the lighter ones, according to the optimization strategy.

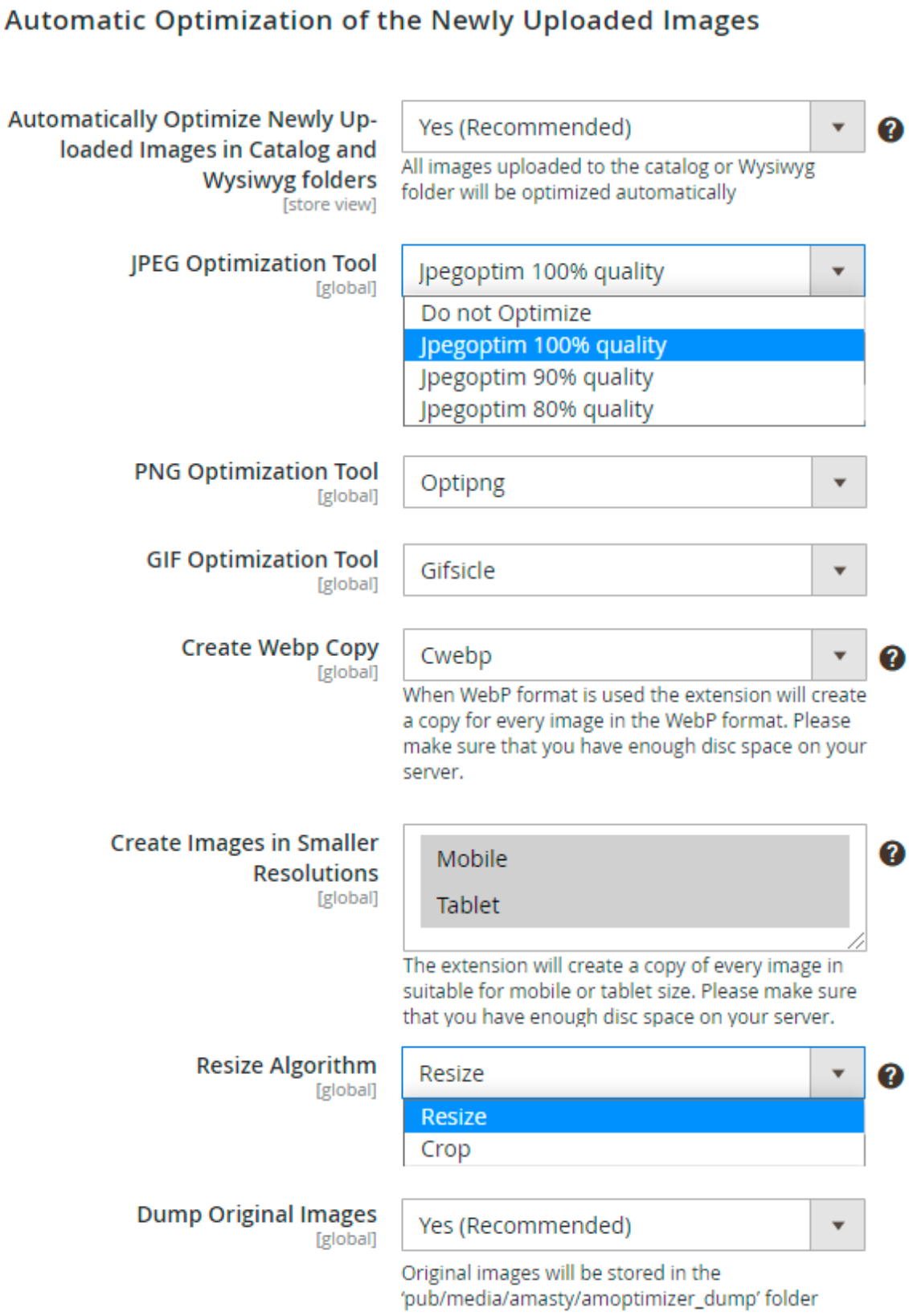
Ignore Images that Contain - specify particular images that should be skipped.

Additionally, you can adjust the optimization for particular store pages separately. The settings are the same as above.



Automatic Optimization of the Newly Uploaded Images

Expand this tab to configure how all new images will be treated. This way, you can automate the optimization process.



Automatically Optimize Images in catalog or Wysiwyg folder - set to *Yes* to optimize all images uploaded to the catalog or Wysiwyg folder automatically.

JPEG optimization tool - set the way to optimize JPEG images:

- **Do not Optimize** - select this option to forbid optimization;
- **jpegoptim 100% quality / jpegoptim 90% quality / jpegoptim 80% quality** - choose the quality level you want to get after compression. See the examples of optimized images right below the setting.

PNG optimization tool - select optipng to optimize PNG images.

GIF optimization tool - choose gifscale to compress GIF files.

Create Webp Copy - choose Cwebp to create a copy for every image in the WebP format.

Please make sure that you have enough disc space on your server. It is also necessary to install the Webp tool package on your server using the command

```
sudo apt-get install webp
```

For a specific setup with versioning, use this guide.

Create Images in Smaller Resolutions - choose the type of copies to be created:

- **Mobile** - the extension will create copies suitable for mobile devices. All copies will be placed in the automatically created folder - ***pub/media/amasty/amoptmobile***.
- **Tablet** - all the copies will be created specially for tablets and placed in ***pub/media/amasty/amopttablet*** folder.

Resize Algorithm - select the algorithm according to which the images will be compressed:

- **Crop** - cuts pictures by size according to the device used, but not proportional to the size of the picture itself. While the height of the image would be the same, in the result the cropped image will look like the part of the original picture.
- **Resize** - compresses the images in proportion to the picture itself according to the size of the field viewed. Simply it will be the same picture, but smaller in pixels.

Dump Original Images - set this option to *Yes* to store original images in the ***pub/media/amasty/amoptimizer_dump*** folder. It lets to collect all the original files in one place and easily update them in case the settings are changed.

WebP Compression Quality

By default, the compression quality of WebP images is set to 75%. But still you can change this setting. Check the list of all possible modifications here.

The configuration is set in the following file:

```
Amasty_Image_Optimizer_module_directory/Model/Command/Cwebp.php
```

To make adjustments, find this part:

```php
<?php
declare(strict_types=1);

namespace Amasty\ImageOptimizer\Model\Command;

use Amasty\ImageOptimizer\Api\Data\QueueInterface;

class Cwebp extends ShellCommand
{
    public function getName(): string
    {
        return (string)__('Cwebp');
    }

    public function getType(): string
    {
        return 'cwebp';
    }

    protected function getCommand(): string
    {
        return 'cwebp %s -o %s';
    }

    protected function getCheckCommand(): ?string
    {
        return 'cwebp -help';
    }
```

Make the needed changes according to this guide.

The result will be similar to this sample, where we've set the compression quality to 95%:

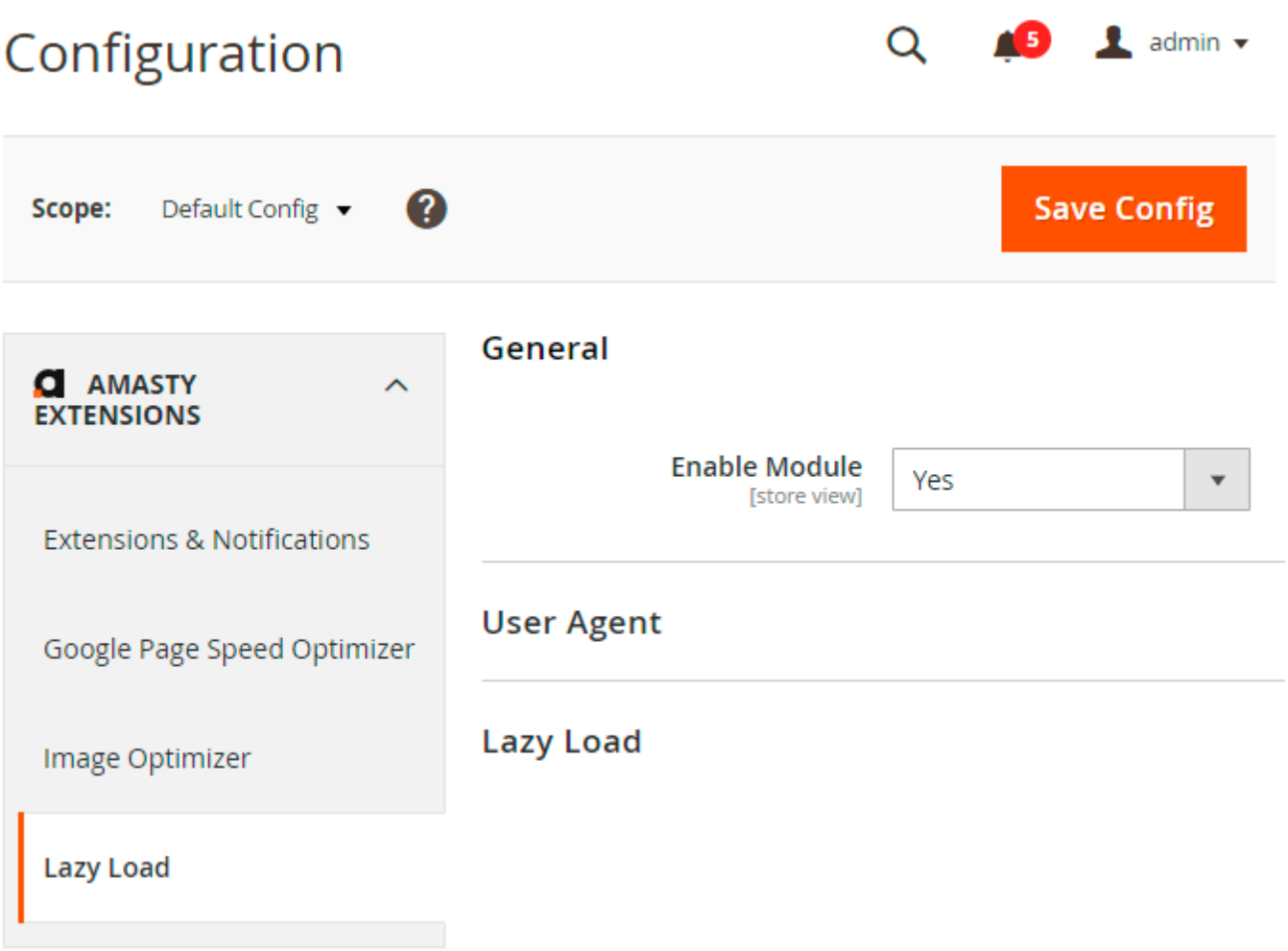
The described changes are applicable to other libraries as well.

Delete Previously Generated WebP & Resized Images

Clear Images Folder - the *Clear WebP Images Folder / Clear Mobile Images Folder / Clear Tablet Images Folder* buttons delete the optimized files from the ***pub / media / amasty / folder_name*** folders, depending on which button is pressed. It is used if the previously selected optimization settings were inconvenient to use and there is a need to re-generate the queue and re-optimize the images.

Lazy Load

This extension is used to load images only when they become visible, which makes pages load faster. Proceed to the **Lazy load** tab to enable the fucntionality.



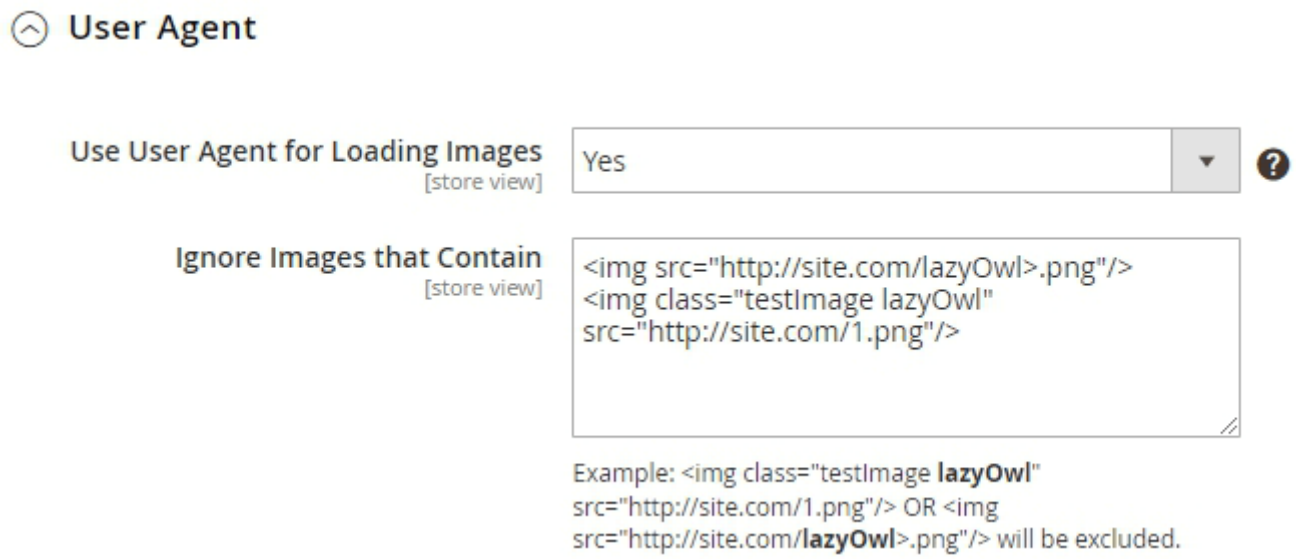
To deliver the media correctly, the **srcset** attribute will be deleted from Lazy Load images.

General

Enable Module - set to Yes to activate lazy load functionality.

User Agent

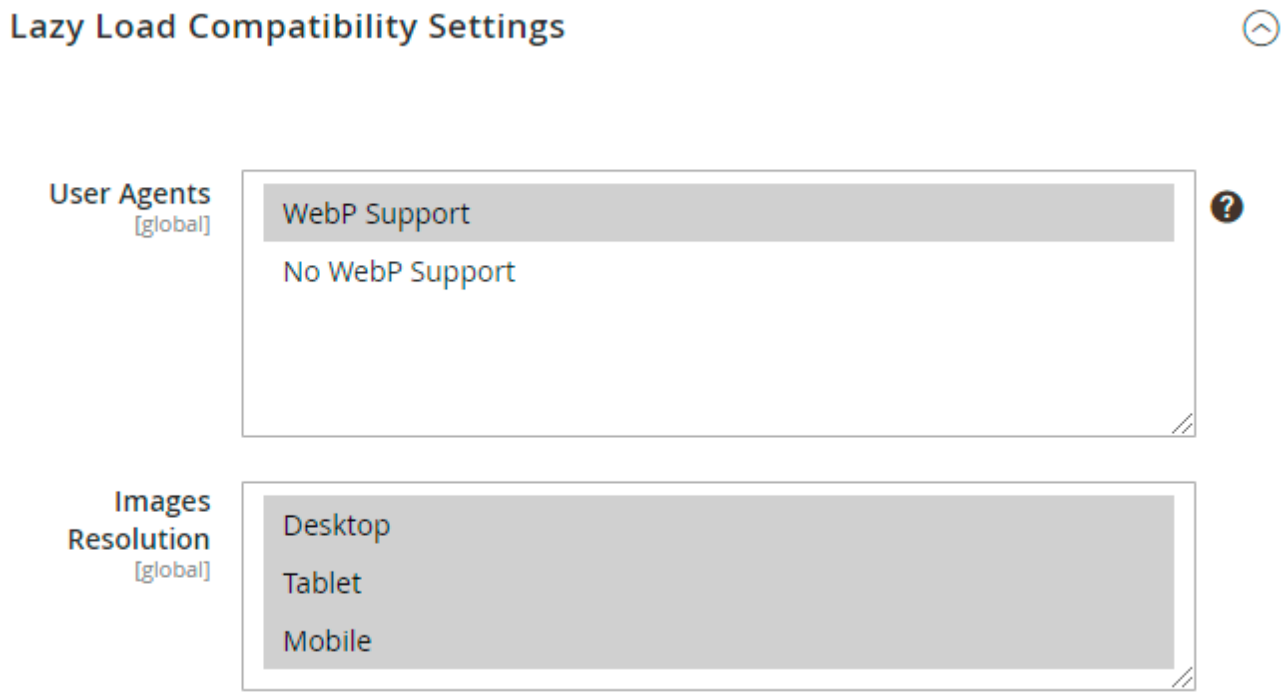
If you have **Image Optimizer** installed, you will be able to configure the user agent for lazy load functionality. Expand the **User Agent** tab to adjust settings for image loading according to the customer's browser user agent.



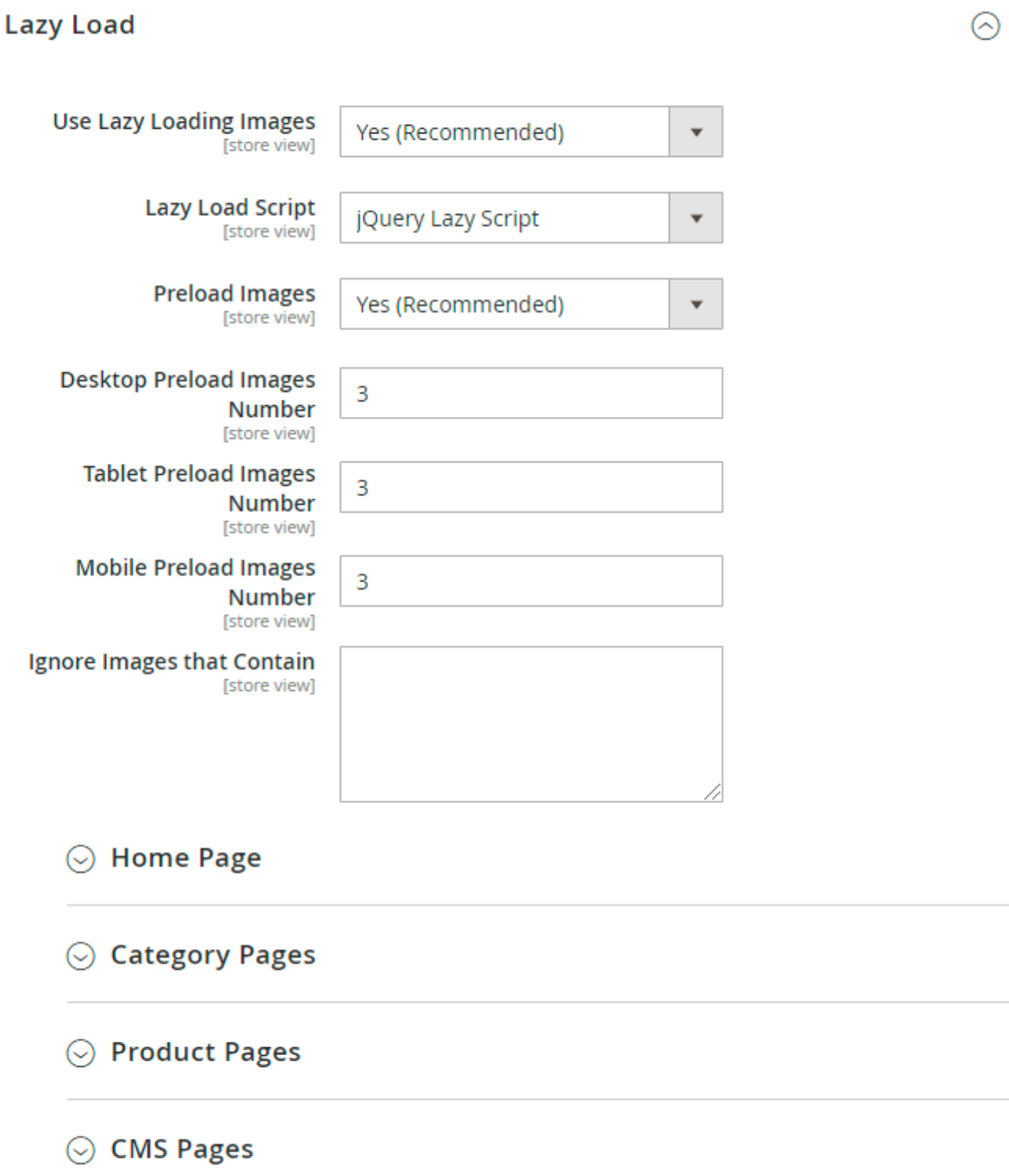
Set the **Use User Agent for Loading Images** to **Yes** to enable fetching appropriate image sizes based on the user agent data, obtained from the browser.

In the **Ignore Images that Contain** you can specify certain images to be excluded from the loading based on the user agent info. Such images will be delivered on the users' devices in the original size.

Lazy Load is fully compatible with the Full Page Cache Warmer extension. If you have both modules installed, you can warm pages according to the user agents.



Lazy Load



Use Lazy Loading Images - set to *Yes* to make off-screen images load only when a customer scrolls to them. It prevents pages full of images from lagging.

Lazy Load Script - you can choose between 2 lazy load scripts: *jQuery Lazy Script* or *Native JS Lazy Script*. We recommend trying both of them to choose one that fits your system best.

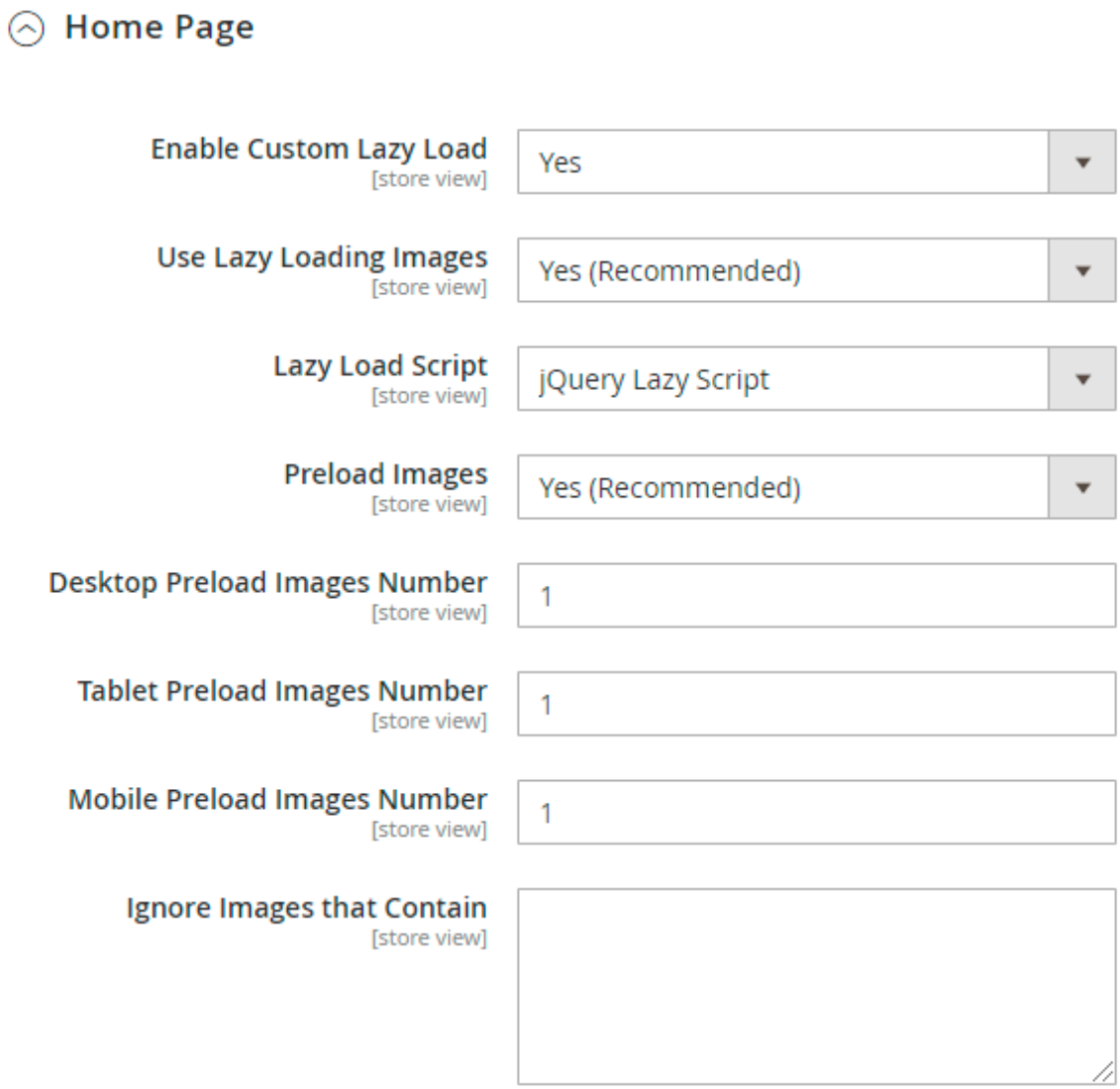
You can decide whether to request the preload of Magento 2 key images or load them from scratch when they happen on screen. Decide how many images to load by setting **Preload Images Number**.

Desktop Preload Images Number - set the number of images to be pre-loaded on the page from the start. You can also assign a specific number of images for tablet devices with the **Tablet Preload Images Number** and for phones with the **Mobile Preload Images Number** fields.

Ignore Images That Contain - specify the images that should not be optimized.

For better compatibility with other extensions, the **img** tag which includes empty **src** will be ignored.

All other tabs of this block have the same structure, but you can specify the settings for particular site pages. Thus, you can select the most suitable type of optimization for each store page.



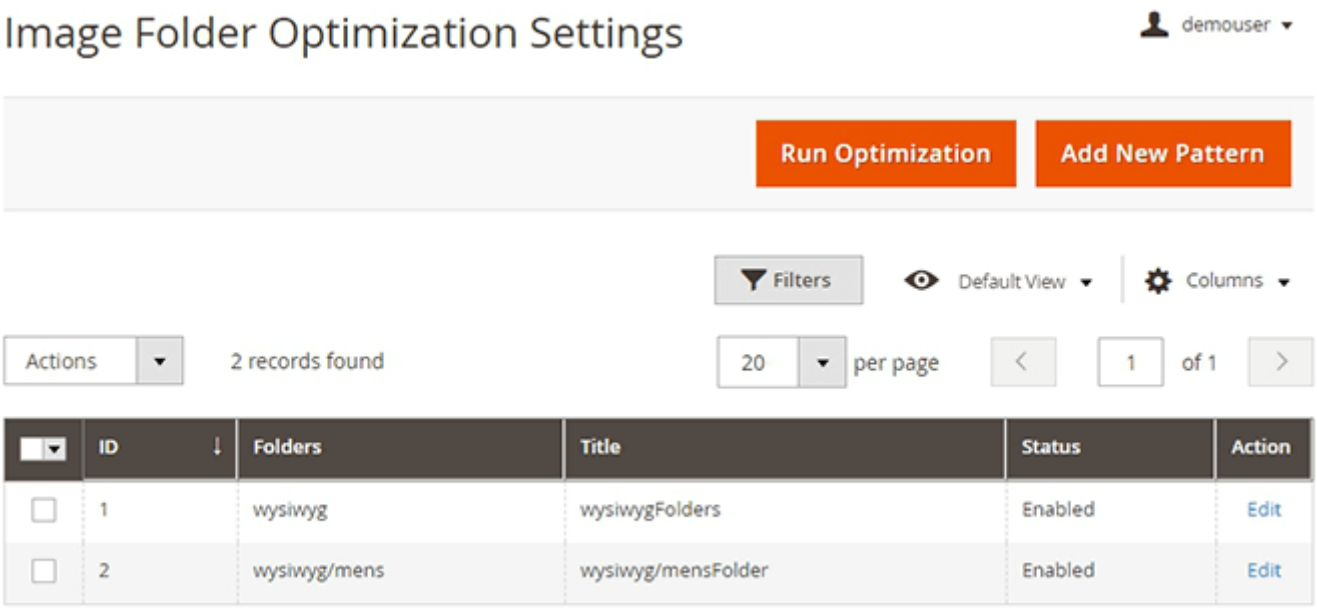
Save the configuration.

When you’re ready with the optimization, don’t forget to check your store with the Google PageSpeed Insights tool once again and compare the scores before and after optimization.

Image Folder Optimization Settings

These settings will allow to precisely configure the optimization pattern for folders.

Navigate to **Content → Image Optimizer → Image Folder Optimization Settings**. To create new set of rules, please click on the **Add New Pattern** button.



All configured patterns will be displayed in one handy grid. You can manage them any time you need just in a few clicks.

Run Optimization - click this button to optimize selected folders images.

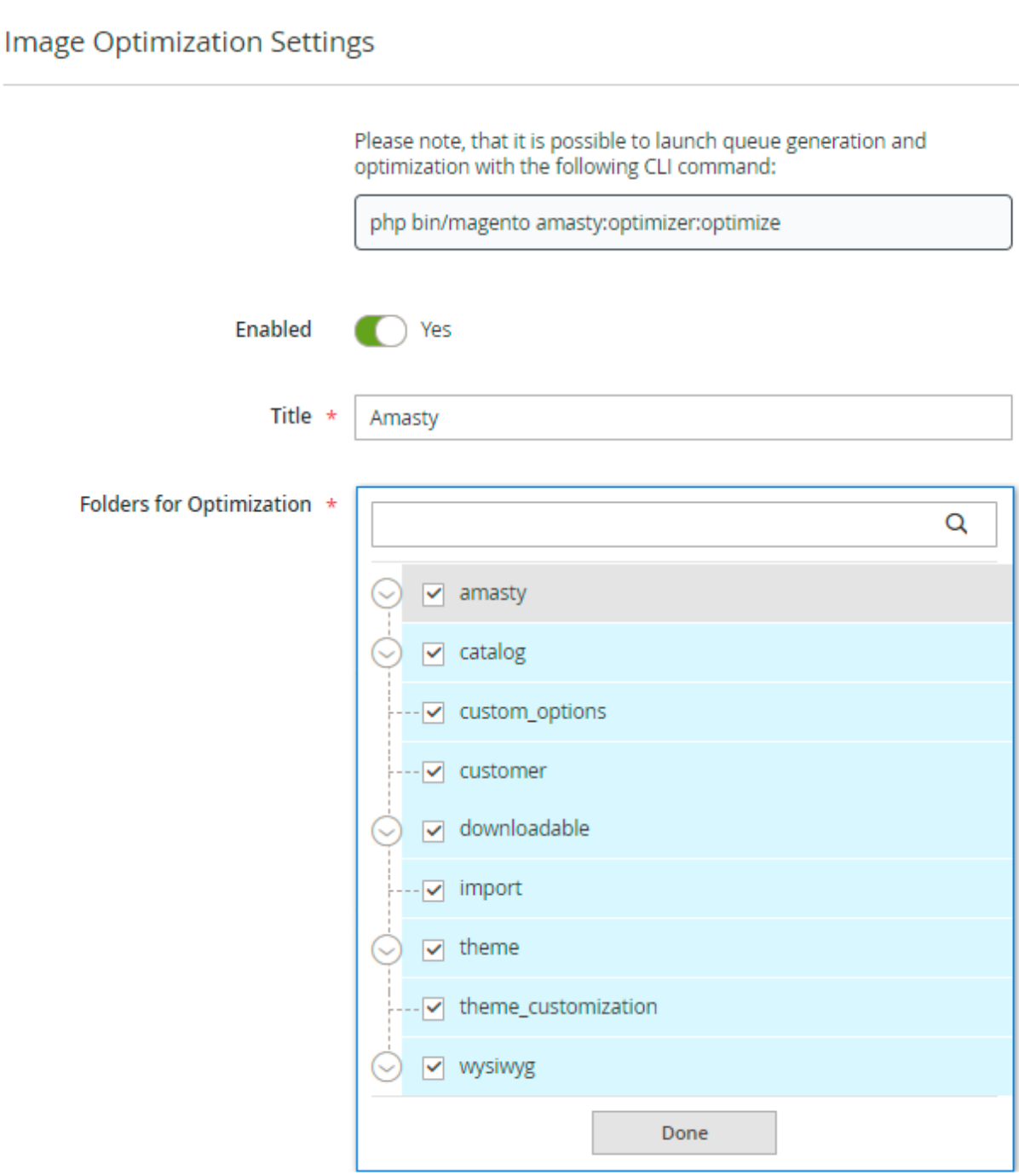
To run optimization manually, execute the following command:

```
php bin/magento amasty:optimizer:optimize
```

Please mind that running optimization will make an additional load to the server.

New Pattern for Image Optimization settings

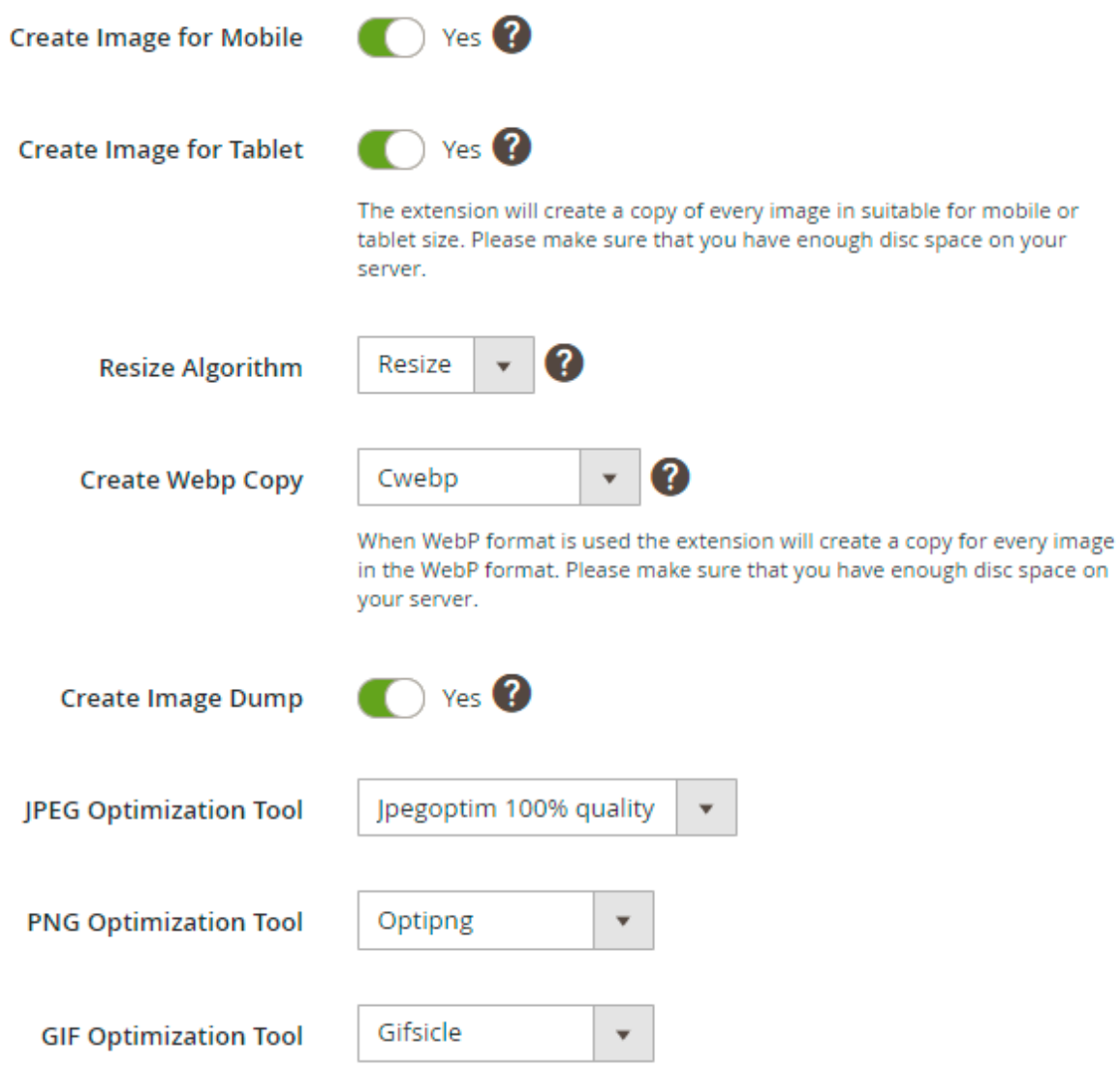
As you click the **Add New Pattern** button, a new **Image Optimization Settings** page will expand.



Set the **Enabled** switch to **Yes** to allow this set of configurations.

Title - enter the title of the configuration set to easily distinguish numerous configs in the grid.

Folders for Optimisation - click on the folder name to include it to the list of folders that will be optimized.



Create Image for Mobile/Tablet - set the corresponding switch to **Yes** to automatically create appropriately sized images for various displays. All newly created copies will be saved on your website server, so check in advance if you have enough disc space.

Resize Algorithm - select the algorithm according to which the images will be compressed:

- **Crop** - cuts pictures by size according to the device used, but not proportional to the size of the picture itself. While the height of the image would be the same, in the result the cropped image will look like the part of the original picture.
- **Resize** - compresses the images in proportion to the picture itself according to the size of the field viewed. Simply it will be the same picture, but smaller in pixels.

Create Webp Copy - switch to **Cwebp** if you want to create a copy for every image in folders in the WebP format. All newly created copies will be saved on your website server, so check in advance if you have enough disc space.

Create Image Dump - set to **Yes** to to store original images in the ***pub/media/amasty/amoptimizer_dump*** folder. It allows collect all the original files in one place and easily update them in case the settings are changed.

JPEG optimization tool - set the way to optimize JPEG images:

- **Do not Optimize** - select this option to forbid optimization;
- **jpegoptim 100% quality / jpegoptim 90% quality / jpegoptim 80% quality** - choose the quality level you want to get after compression. See the examples of optimized images right below the setting.

PNG optimization tool - select optipng to optimize PNG images.

GIF optimization tool - choose gifscale to compress GIF files.

Click on the **Save** button to preserve the settings.

AJAX Compatibility

You are able to to add the compatibility with 3-rd party extensions which use AJAX. It can be done with the usage of the following sample:

```
<?php

class Test
{
    /**
     * @var \Magento\Framework\DataObjectFactory
     */
    private $dataObjectFactory;

    /**
     * @var \Magento\Framework\Event\ManagerInterface
     */
    private $eventManager;

    public function __construct(
        \Magento\Framework\DataObjectFactory $dataObjectFactory,
        \Magento\Framework\Event\ManagerInterface $eventManager
    ) {
        $this->dataObjectFactory = $dataObjectFactory;
        $this->eventManager = $eventManager;
    }

    public function execute()
    {
        $someHtml = '<div><img src="test.jpg">...</div>';
        $data = $this->dataObjectFactory->create(
            [
                'page' => $someHtml,
```

```
                //
\Amasty\PageSpeedOptimizer\Model\Output\LazyLoadProcessor::PAGE_CONFIG keys
                'pageType' => 'catalog_category_view',
                // optional. Example of configs
                //
\Amasty\PageSpeedOptimizer\Model\Output\LazyLoadProcessor::prepareLazyConfig
                'lazyConfig' => [

                ]
            ]
        );
        //if Amasty PageSpeedOptimizer is not installed output page will be
the same
        $this->eventManager->dispatch('amoptimizer_process_ajax_page',
['data' => $data]);
        $someHtml = $data->getData('page');

        return $someHtml;
    }
}
```

Varnish Compatibility

To make Varnish compatible with the WebP functionality, you need to customize the Varnish configuration. Also, now the extension supports WebP for Safari 14.

To add WebP images support for users who have never visited the site using Varnish, you need to change the Varnish **`defaul.vcl`** configuration file.

Follow the steps below:

1. Find the **`vcl_hash`** subroutine

2. Add the code:

```
# Amasty Google Page Speed Optimizer device detection and cache separation
mechanism
    if (req.http.cookie !~ "X-Magento-Vary=") {
        call device_detect;

        if (req.http.X-Amasty-Accept-Webp && req.http.X-Amasty-Device) {
            hash_data(req.http.X-Amasty-Accept-Webp);
            hash_data(req.http.X-Amasty-Device);
        }
    }
```

3. Add a new subroutine using this code:

```
sub device_detect {
    unset req.http.X-Amasty-Accept-Webp;
    unset req.http.X-Amasty-Device;

    if (req.http.Accept ~ "image\/webp" ||
        req.http.User-Agent ~
"(?i)(\biPhone.*Mobile|\biPod|\biPad|\bMacintosh).*Version\/14" ||
        req.http.User-Agent ~ "(?i)Edge|Firefox|Chrome|Opera"
    ) {
        set req.http.X-Amasty-Accept-Webp = "true";
    }

    if (req.http.User-Agent ~ "(?i)Mobile") {
        set req.http.X-Amasty-Device = "mobile";
    } elsif (req.http.User-Agent ~ "(?i)Tablet") {
        set req.http.X-Amasty-Device = "tablet";
    } else {
        set req.http.X-Amasty-Device = "desktop";
    }
}
```

4. Reload Varnish.

Find out how to install the Google Page Speed Optimizer via Composer