Ways to Optimize Magento Store Performance

by Ananthu G

In the highly competitive world of e-commerce, the performance of your Magento store plays a crucial role in attracting and retaining customers. A slow-loading website can result in a poor user experience, leading to higher bounce rates and lost sales. To ensure your Magento store operates at its peak efficiency, it‘s essential to implement optimization strategies.

In this article, the focus will be on the importance of speed for your Magento store, providing insights on testing the store‘s performance, and offering valuable tips. By addressing these aspects, Magento store owners can enhance the overall user experience and prevent potential customers from leaving the site due to slow loading times.

How to Test Magento 2 Performance?

Evaluating site speed is crucial for identifying and addressing performance issues in the Magento 2 platform. Conducting regular speed tests as part of a maintenance routine is essential for optimizing and enhancing the overall speed of your Magento store.

To assess the speed of your Magento 2 site, one can consider utilizing third-party services such as GTmetrix, WebPageTest, or Google PageSpeed Insights. These tools enable magento store owners to conveniently assess the store‘s performance by simply entering the URL and initiating the tests.

How can the Speed Optimized

Before optimizing the store with the following tips, please ensure it meets the latest Magento 2 system requirements.

 Choose Reliable Hosting

 Update the Magento Version

 Upgrade Package & Database

 Use Redis Cache - Avoid Built-in Cache

 Update Indexers

 Use Varnish Cache

 Check Third-Party Modules

 Optimize Product Images

 Minify and Merge JavaScript and CSS

 Instant results using Elasticsearch

 Content Delivery Network Integration

 Remove Database Logs

Choose Reliable Hosting

Opting for a hosting service that prioritizes speed, security, and reliability is essential to ensure the optimal performance and protection of an online store. While the temptation of low-priced hosting plans might be strong, it often results in significant drawbacks in terms of time and effort. Opting for mediocre hosting may lead to ongoing server issues that can adversely impact your Magento store's functionality, causing both performance and security concerns.

To mitigate these challenges, it is highly advisable to invest in a managed hosting provider. By choosing a managed hosting service, they can streamline the focus on business growth and leave the technical aspects of server management in the hands of experts. This approach ensures that your Magento store operates efficiently and securely, allowing them to dedicate the time and resources to expanding the online business without the distraction of server-related problems.

There are Magento system requirements that must be ensured by the hosting provider and see if the server meets them. For example, PHP 8. x, MariaDB 10.x, MySQL 8.x, Elasticsearch 7.x, Composer 2.x, etc.

Software dependencies	2.4.7-beta3	2.4.6-p4	2.4.6-p3	2.4.6-p2	2.4.6-p1	2.4.6	2.4.5-p6	2.4.5-p5	2.4.5-p4	2.4.5-p3
Composer	2.6	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2	2.2
Elasticsearch	--	--	--	--	--	--	--	--	--	--
OpenSearch	2.11	2.5	2.5	2.5	2.5	2.5	1.2	1.2	1.2	1.2
MariaDB	10.6	10.6	10.6	10.6	10.6	10.6	10.4	10.4	10.4	10.4
PHP	8.3, 8.2	8.2, 8.1	8.2, 8.1	8.2, 8.1	8.2, 8.1	8.2, 8.1	8.1	8.1	8.1	8.1
RabbitMQ	3.12	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9	3.11, 3.9
Redis	7.2	7.0	7.0	7.0	7.0	7.0	6.2	6.2	6.2	6.2

Magento System Requirements

Update the Magento Version

Upgrading the Magento store to the latest releases is crucial for unlocking enhanced features, addressing bugs, and significantly boosting the site's performance. This fundamental tip for optimizing the speed of the Magento 2 store underscores the importance of staying current with the latest software versions.

For those still using Magento 1, a highly advisable course of action is migrating to Magento 2. This transition not only ensures a better and more reliable e-commerce solution but also introduces heightened security measures, all contributing to a faster and more efficient online shopping experience for both merchants and customers. Embracing the latest Magento versions not only addresses existing issues but also aligns the store with the evolving technological landscape, providing a foundation for sustained growth and optimal performance.

Upgrade Package & Database

When upgrading Magento to the latest version, component dependency conflicts with different packages may be detected. Instead of patching them manually, update the packages and then run the update.

This usually happens to advanced users who have multiple versions of PHP installed. For security reasons, first, the prerequisites of the packages should be checked, which facilitates the upgrade process.

Use Redis Cache – Avoid Built-in Cache

In the present day, there is a significant focus on enhancing the speed and performance of Magento 2. Various technologies, including Varnish, Redis, Nginx, and comprehensive full-page caching solutions, are being integrated to address these concerns.

1. **Use Memcached or Redis**
   Both Redis and Zend_Cache_Backend_File are distributed memory caches designed to enhance the performance of large-scale web applications running on Magento 2 with dynamic databases. Their primary function is to alleviate the server's load by reducing the frequency of database requests and efficiently delivering cached data when external data requests permit.

   In Magento 2, Zend_Cache_Backend_File serves as the default back-end cache solution. However, users have the option to leverage Redis as an alternative, providing a more sophisticated caching mechanism. This transition from Zend_Cache_Backend_File to Redis is particularly beneficial for high-traffic Magento 2 stores.

When it comes to PHP sessions, both Redis and Zend_Cache_Backend_File prove to be suitable choices. Notably, Redis can be configured to handle an impressive number of session connections per second, exceeding 60,000. This capability makes it a robust solution for managing PHP sessions in Magento 2.

Choosing Redis as the backend cache introduces a high-speed caching system with comprehensive cache tag support. Additionally, it eliminates the necessity for an additional low-level file system cache. This results in outstanding and consistent performance, especially in the demanding environments of high-traffic Magento stores. The integration of Redis enhances stability and responsiveness, making it a preferred option for optimizing the performance of Magento 2 web applications.

2. **Configure Redis for Session Storage**

   Please include the following configuration in <Magento install dir>app/etc/env.php file:

```
'session' =>
  array (
   'save' => 'redis',
   'redis' =>
   array (
    'host' => '127.0.0.1',
    'port' => '6379',
    'password' => '',
    'timeout' => '2.5',
    'persistent_identifier' => '',
    'database' => '2',
    'compression_threshold' => '2048',
    'compression_library' => 'gzip',
    'log_level' => '4',
    'max_concurrency' => '6',
    'break_after_frontend' => '5',
    'break_after_adminhtml' => '30',
    'first_lifetime' => '600',
    'bot_first_lifetime' => '60',
    'bot_lifetime' => '7200',
    'disable_locking' => '0',
    'min_lifetime' => '60',
    'max_lifetime' => '2592000'
   )
  ),
```

Magento 2 also offers a command for establishing and configuring Redis. Utilize the `setup:config: set` command to configure the following parameters.

```
bin/magento setup:config:set --session-save=redis --session-save-redis-<parameter_name>=<parameter_value>...
```

Review all parameters to ensure that the Redis installation is functioning correctly with your Magento store.

```
bin/magento setup:config:set --cache-backend=redis --cache-backend-redis-<parameter>=<value>...
```

- **Configure Redis for Session Storage**
  Run the setup:config:set command and specify parameters that are specific to Redis default caching.
  With the following parameters:
- --cache-backend=redis enables the Redis default caching. If this feature has already been enabled, omit this parameter.
- --cache-backend-redis-<parameter>=<value> is a list of key-and-value pairs that configure the default caching:

  The following example enables Redis page caching, sets the host to 127.0.0.1, and assigns the database number to 1. All other parameters are set to the default value.

```
bin/magento setup:config:set --page-cache=redis --page-cache-redis-server=127.0.0.1 --page-cache-redis-db=1
```

  You can check all the details on the Magento DevDocs.

Update Indexers

Magento facilitates efficient management of bulk data processing, reducing system load times. To optimize this process, Magento uses indexers to update indexes on stored objects. By default, these indexers update on every object save operation, but this approach is not flawless, necessitating manual updates.

There are two modes for Magento indexers: „Update on Save“ and „Update on Schedule.“ The recommended mode is „Update on Schedule,“ allowing the configuration of a specific time for cron job execution. This ensures that indexers are updated at times of lower website traffic, minimizing any adverse impact on user experience.

For Magento 2, one needs to execute the following CLI command on the root directory using the SSH terminal:

```
php bin/magento indexer:reindex
```

To change the indexing mode to “Update on Schedule”, run the following command:

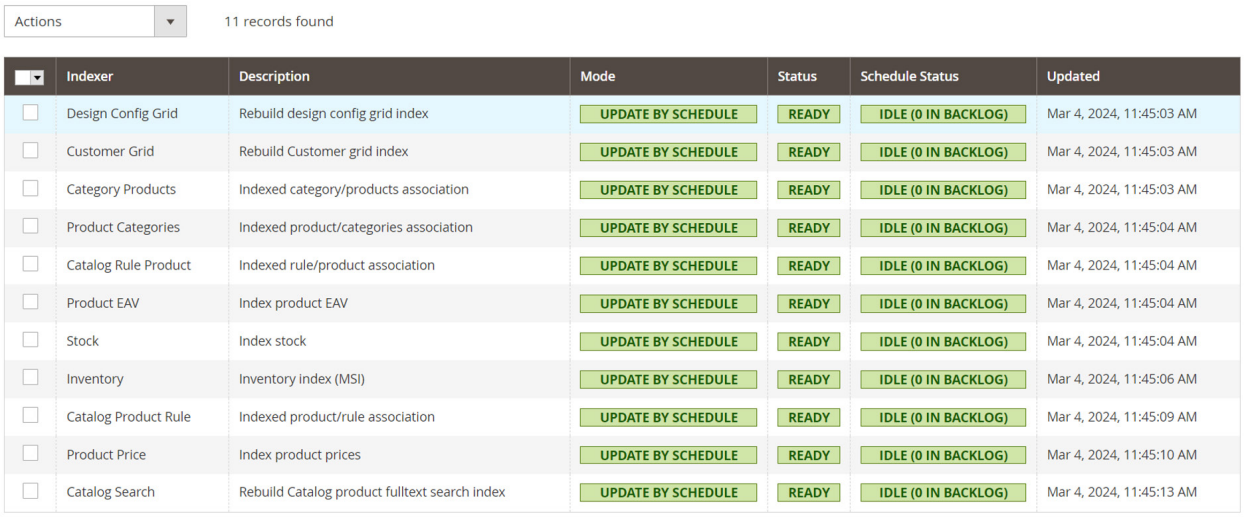
```
php bin/magento indexer:set-mode schedule
```

To change the indexing mode to “Update on Save”, run the following command:

```
php bin/magento indexer:set-mode realtime
```

In Magento Admin:

- Log in to Magento Admin and go to **System > Index Management**.
- Select all indexers.
- In the **Actions** dropdown, select **Update on Schedule.**
- Click **Submit** to set the schedule.

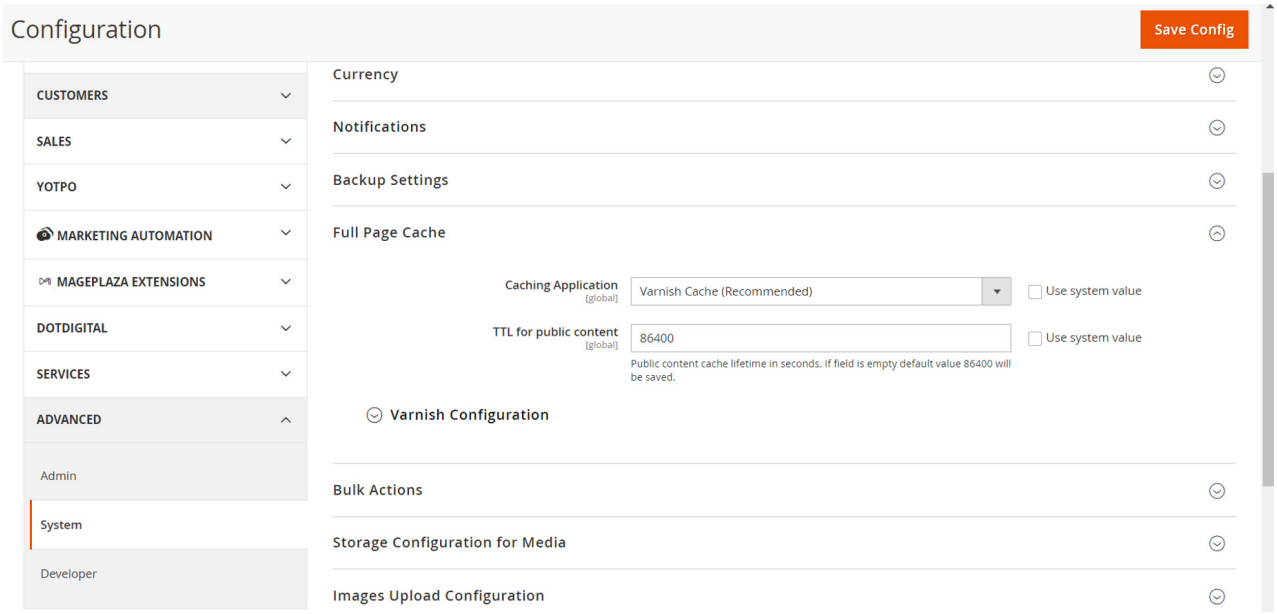


Magento Admin - Update By Schedule

Use Varnish Cache

When implementing Varnish with Magento 2, the advantage is its ability to cache static content and reduce the load on the web server. Magento 2.2 and later versions have built-in support for Varnish. To enable Varnish Cache

- Go to **Stores > Configuration > Advanced > System > Full Page Cache.**
- In the **Caching Application list**., click **Varnish Caching.**
- Enter the **TTL value in the public content field.**
- Expand **Varnish Configuration** and set the information.



Magento Admin - Update By Schedule

Check Third-Party Extensions

It is highly recommended to conduct an audit of third-party extensions installed in your Magento store. This can be done by systematically disabling each module, clearing the cache, and evaluating the impact on store performance.

Thoroughly test all major CMS pages, including the homepage, categories, products, and checkout pages. If disabling a Magento module improves the store speed, it‘s a sign of a potential problem.

Promptly reach out to the extension providers or developers, explain the problem encountered, and request a refund. Keep them informed about the specific issues faced during the testing process.

Optimize Product Images

Image size affects website speed. Therefore, one must ensure that images are properly optimized and have a good quality-to-size ratio.

In older technology, the server downloads content images while the page is loading. To solve this problem, Magento uses lazy loading. This means that the images will load after the page has fully loaded or while the user is scrolling through the catalog.

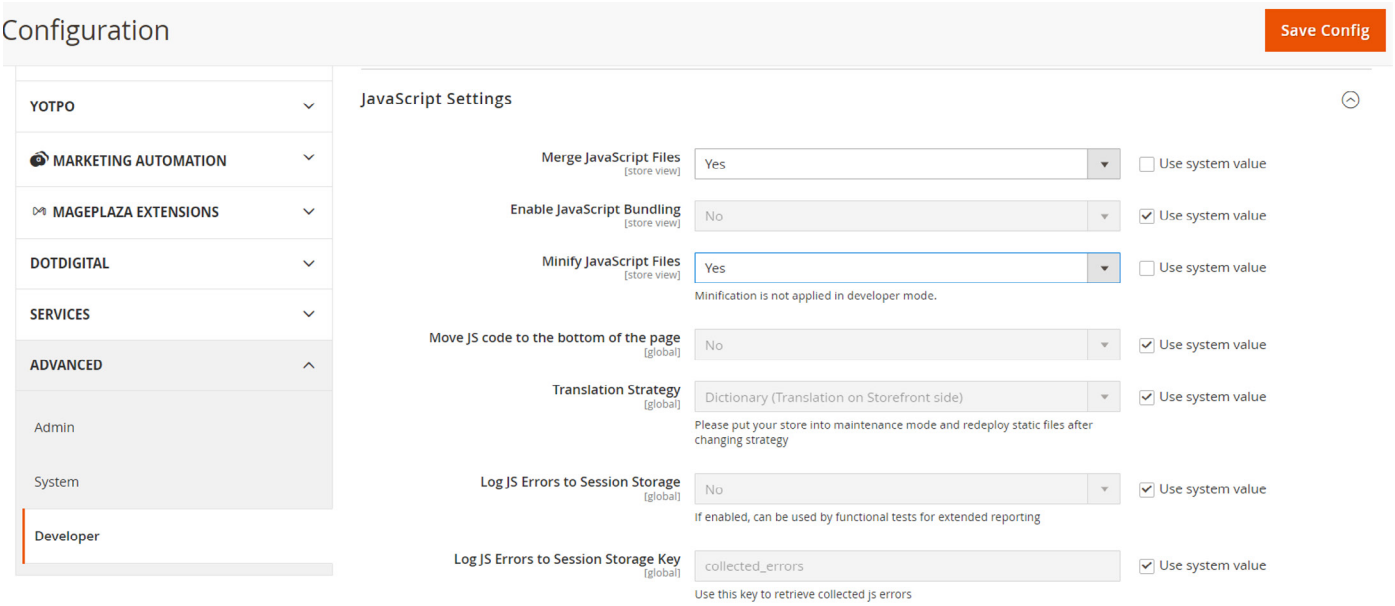
Store owners can also use AWS or CDN to deliver the content faster. Make sure that the images are well-optimized and have a good quality-to-size ratio.

Minify and Merge JavaScript and CSS Files

Another major reason for a slow website is JS and CSS files that are not merged and minified. Code minification compresses the code by removing all unnecessary characters, spaces, comments, and line breaks. It reduces the amount of code that needs to be transferred and makes your Magento website much faster than before.

Similarly, merging combines JS and CSS files into one. This reduces latency and allows files to be loaded quickly in batches. To Enable JS minification from Magento Admin.

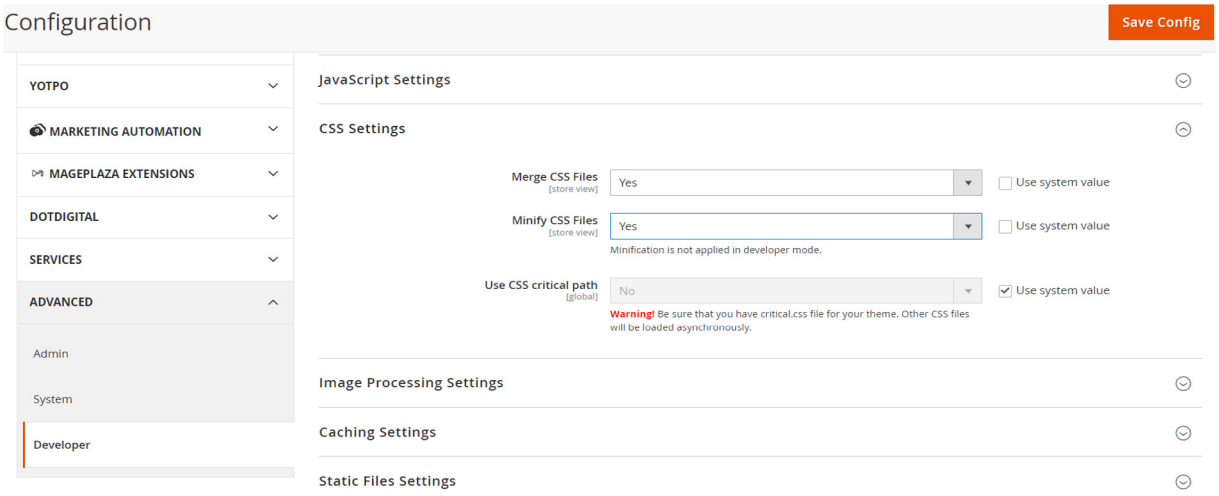
- Put the Magento store into production mode.
- Go to **Stores > Configuration > Advanced > Developer > JavaScript Settings.**
- Set the **Merge JavaScript files** and **Minimize JavaScript files** options to **Yes**.
- Click **Save Config** and **Flush Cache** on the **System > Cache Management** page.



Magento Admin - Minify and merge JavaScript

To Enable CSS minification from Magento Admin.

- Put the Magento store into production mode.
- Navigate to **Stores > Configuration > Advanced > Developer > CSS Settings**.
- Set the **Merge CSS Files** and **Minify CSS Files** options to **Yes**.
- Click **Save Config** and **Flush Cache** on the **System > Cache Management** page.



Magento Admin - Minify and merge CSS

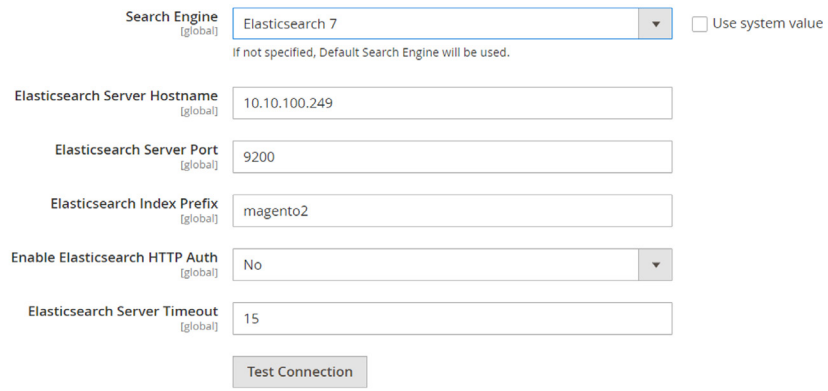
Detailed testing is recommended to ensure that the implementation of JS and CSS file merging and minification does not introduce any unintended issues or conflicts. This involves thoroughly examining the website's performance before and after the implementation, and testing various functionalities and scenarios to verify that everything operates smoothly and efficiently.

Instant results using Elasticsearch

Elasticsearch provides extremely fast full-text search. The indexing service runs on the server instead of searching an entire database for matches, providing fast web content search capabilities.

For store owners with many products, Elasticsearch offers a way to speed up search results and catalog pages. In Magento 2.4.x and later versions, Elasticsearch is preactivated with default settings. However, for earlier versions, it can be configured by following these steps:

- Go to **Stores** > **Configuration** > **Catalog** > **Catalog** > **Catalog Search.**
- Click on **Elasticsearch** under the **Search Engine** option.
- Click **Save**.



Magento Admin - Elasticsearch Configurations

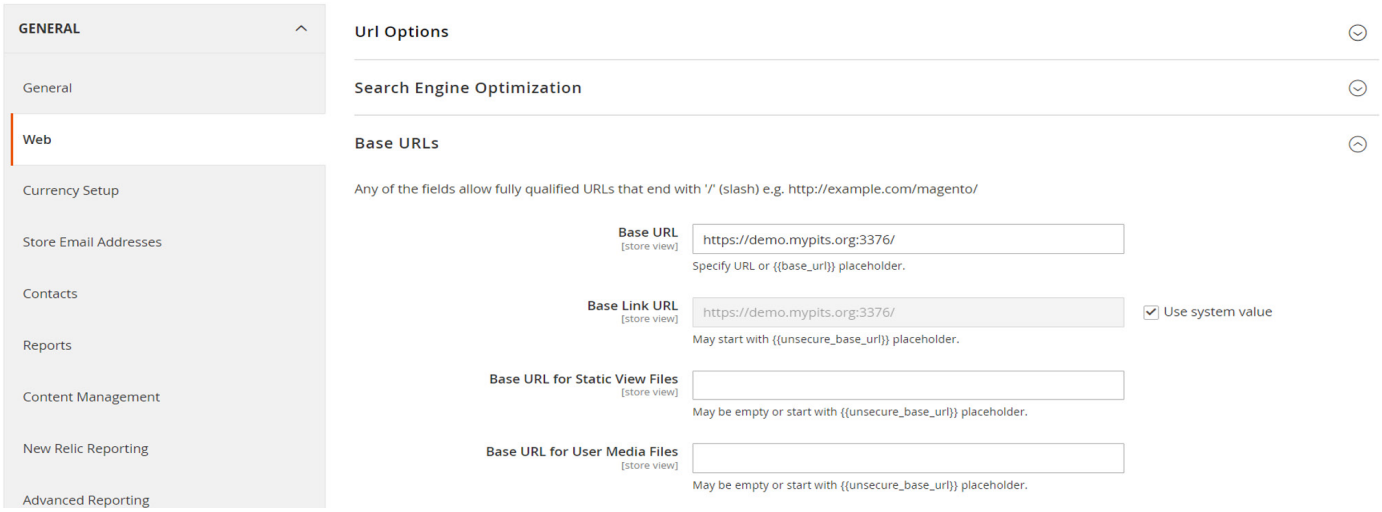
Content Delivery Network Integration

A content delivery network (CDN) stores static content from your Magento store, including CSS, JavaScript, images, videos, and fonts. This leads to a notable decrease in response time for users accessing the shop.

To provide the viewers with a better shopping experience, It is recommended to configure a CDN for the Magento store.

To enable CDN functionality, proceed as follows:

- Navigate to **Stores** > **Configuration** > **General** > **Web.**
- Select the **Base URLs** section and do the following:
- In the **Base URL for Static View Files** field, enter the URL of the CDN location where the static view files are stored.
- In the **Base URL for User Media Files** field, enter the URL of the location of JavaScript files on the CDN.
- Click **Save Config**.



Magento Admin - CDN Configurations

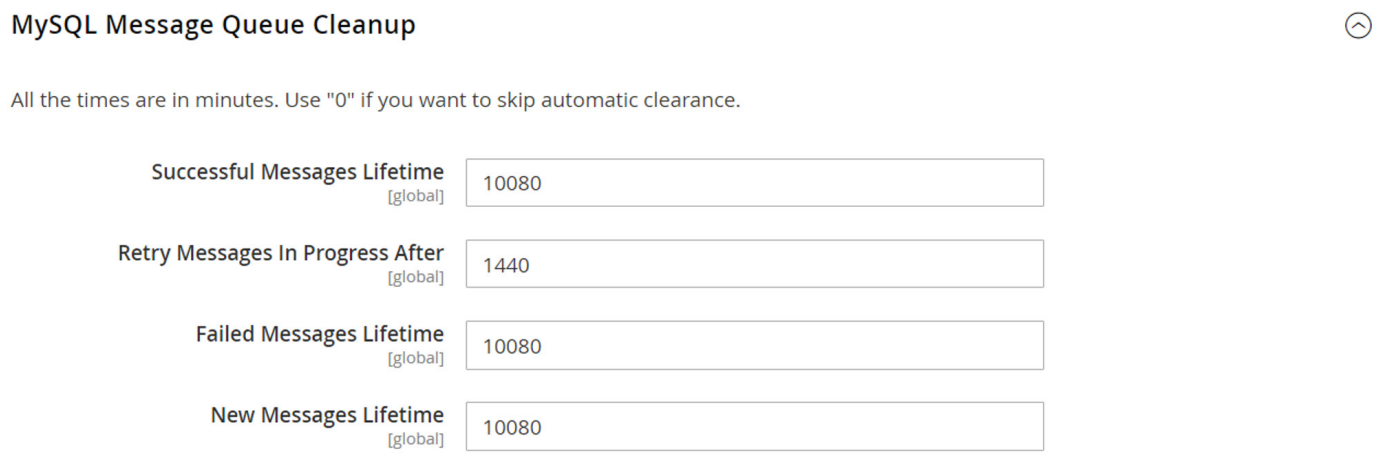
Remove DataBase Logs

Removing database logs significantly improves database performance by reducing process execution delays.

Note: It is recommended to take a recent backup before making any new changes to the database.

Additionally, Magento generates new log files automatically and frequently, so deleting old log files does not cause any problems. To set the log time:

- Go to **Store > Configuration > Advanced > System.**
- Open the **MySQL Message Queue Cleanup** section.
- Under **New Message Expiration Time**, set a specific amount of time to automatically clean up old log files.
- Click **Save Config.**



Magento Admin - New Message Expiration Time Configurations

References

https://experienceleague.adobe.com/en/docs
https://developer.adobe.com/commerce/docs/
https://www.cloudways.com/blog/magento-optimization/#magento-speed-optimization-performance
https://envisionecommerce.com/blog/a-guide-to-magento-store-performance-optimization/

Contact us:
kontakt@pitsolutions.ch
www.pitsolutions.com