FREE CHECKLIST · SHOPWARE 6

The Shopware performance checklist

15 quick wins to make your store faster.

Checked against Shopware 6.7.14 and its developer docs, September 2026 · huzaifamustafa.com

Shopware's defaults are built to run on any server. On a production store with real traffic, that leaves speed on the table.

Every item below comes from Shopware's own hosting docs and is checked against the 6.7.14 source code. Each one lists the change and the catch. Try them on staging first, one at a time, and measure before and after.

Caching

01 Keep the HTTP cache on

Pages come out of the cache instead of being rendered by PHP on every request. This is the biggest single win for most stores. Shopware turns it on by default, so check that nobody switched it off.

```
# .env.local
SHOPWARE_HTTP_CACHE_ENABLED=1
```

CATCH The built-in cache stores pages in the app cache. Once you run several app servers, put a reverse proxy such as Varnish in front.

02 Move the app and system cache to Redis

The file system cache gets slow under load and doesn't share between servers. Redis does both.

```
# config/packages/cache.yaml
framework:
  cache:
    app: cache.adapter.redis_tag_aware
    system: cache.adapter.redis_tag_aware
    default_redis_provider: redis://localhost
```

CATCH `redis_tag_aware` needs Shopware 6.5.8.3 or newer and the PHP Redis extension. Clean up orphaned tags on a schedule (for example `frosh:redis-tag:cleanup` from FroshTools) or Redis memory keeps growing.

03 Move delayed cache invalidation to Redis

By default, pending invalidations are stored in MySQL. On busy stores that puts extra load on the database.

```
shopware:
  redis:
    connections:
      ephemeral:
        dsn: 'redis://localhost:6379/1'
  cache:
    invalidation:
      delay_options:
        storage: redis
        connection: 'ephemeral'
```

CATCH `ephemeral` is a connection name, so it has to exist under `shopware.redis.connections`. Worth it on high-traffic stores. On a small store you won't notice a difference.

04 Compress the cart and switch to zstd

Shopware compresses cache entries with gzip by default but leaves the cart uncompressed. Compressing the cart and moving both to zstd means less memory and less data moving between PHP and storage. zstd is available from 6.6.4.0.

```
shopware:
  cart:
    compress: true
    compression_method: zstd
  cache:
    compression_method: zstd
```

CATCH zstd needs the PHP zstd extension, which PHP doesn't ship by default. Install it first (for example `pecl install zstd`, then `extension=zstd` in php.ini). Clear the cache after you change the cache compression method.

Background work

05 Turn off the admin worker and run real workers

The admin worker only processes the queue while someone has the Administration open. Background jobs stall, and the open browser tab keeps sending requests to your server.

```
# config/packages/shopware.yaml
shopware:
  admin_worker:
    enable_admin_worker: false

# run with systemd or supervisor
bin/console messenger:consume async low_priority --time-limit=60 --memory-limit=512M
```

CATCH Also run a worker for the `failed` transport. Otherwise failed messages are never processed.

06 Run scheduled tasks from cron

Cleanup, indexing and other recurring jobs run on time instead of whenever the queue gets around to them.

```
*/5 * * * * /usr/bin/php /var/www/html/bin/console scheduled-task:run --no-wait
```

CATCH Match the cron interval to your shortest task interval. Every 5 minutes fits the defaults, since cache invalidation runs every 5 minutes.

07 Send emails through the queue

Checkout doesn't wait for the mail server. The order confirmation goes out a moment later from a worker.

```
framework:
  mailer:
    message_bus: 'messenger.default_bus'
```

CATCH Only do this once the workers from item 05 are running, or emails sit in the queue.

08 Stop updating mail template variables on every send

Each sent email also writes example data back to the database. Turning that off removes a write from every order.

```
shopware:
  mail:
    update_mail_variables_on_send: false
```

CATCH The mail template editor in the Administration loses its example data for auto-completion.

09 Generate the sitemap from cron

Large catalogues make sitemap generation heavy. Run it at a quiet hour instead of inside the scheduled task queue. Available from 6.7.1.0.

```
shopware:
  sitemap:
    scheduled_task:
      enabled: false

# cron, for example nightly
php bin/console sitemap:generate
```

CATCH If you disable the task and forget the cron job, your sitemap stops updating.

Database and storefront

10 Load reduced product data in listings

Listings load products without the full description and search keywords. That cuts database load and memory on catalogues with long descriptions. Available from 6.7.12.0 and on by default in new installations. Existing shops have to switch it on.

```
Administration: Settings > Products >
"Load reduced product data in listings"
```

CATCH Only enable it if your theme and extensions don't show the full description or keywords in listings. Check every listing template after you switch it on.

11 Disable the product stream indexer

Saves indexing work on every product change, which adds up fast with large imports. Available from 6.6.10.0.

```
shopware:
  product_stream:
    indexing: false
```

CATCH Category pages built on a stream don't update until the HTTP cache expires, and the "Item in dynamic product group" rule always evaluates to false. Skip this if you use that rule.

12 Turn on the Speculation Rules API

The browser pre-renders the page a shopper is about to open, so the next click feels instant. Available from 6.6.10.0.

```
Administration: Settings > System > Storefront >
"Speculation rules API (Experimental)"
```

CATCH Pre-rendering adds load to your server and can skew your analytics.

PHP and deployment

13 Tune OPcache and turn on preloading

PHP stops checking files on disk for changes and keeps Shopware's classes in memory. Preloading adds another 2 to 5 percent.

```
opcache.validate_timestamps=0
opcache.max_accelerated_files=20000
opcache.interned_strings_buffer=20
opcache.enable_file_override=1
opcache.preload=/path/to/var/cache/opcache-preload.php
opcache.preload_user=nginx
zend.assertions=-1
realpath_cache_ttl=3600
```

CATCH With timestamps off, PHP won't see new code. Reload PHP-FPM (or use cachetool) on every deployment. Preloading also needs a PHP-FPM restart after every cache clear, and the Extension Manager stops working.

14 Dump your environment to .env.local.php

Shopware skips parsing the `.env` files on every request. If your `APP_URL` is correct, also stop the Administration from calling your shop to check it on every load.

```
bin/console system:setup --dump-env
# or
bin/console dotenv:dump prod

# .env.local, only if APP_URL is correct
APP_URL_CHECK_DISABLED=1
```

CATCH Run the dump again whenever an environment variable changes. The dumped file wins over your `.env` files.

15 Use an authoritative Composer classmap

The autoloader looks classes up in one map instead of searching the file system.

```
// composer.json
"config": {
  "optimize-autoloader": true,
  "classmap-authoritative": true
}
```

CATCH It only pays off when every plugin is installed through Composer. One plugin in `custom/plugins` outside Composer makes Shopware switch the authoritative mode back off, and the gain is gone. Run `composer dump-autoload` on every deployment so new classes are in the map.

Stuck on one of these?

Some of these take 5 minutes. Others, like workers and Redis, depend on your hosting and can break things when set up wrong. Send me a short message about your store through the contact form and I'll reply by email.

huzaifamustafa.com/contact/

Sources: Shopware 6.7.14.0 source code and developer documentation, developer.shopware.com/docs/guides/hosting/performance/