

The Shopware performance checklist

16 settings that make a Shopware 6 store faster, each with the exact config and what it can break.

Checked against Shopware 6.7.14.0 and the Shopware developer docs, September 2026.

Audit sheet

#	CHANGE	WHERE	DONE
CACHE			
01	Turn on the HTTP cache	Config file	☐
02	Move the app and system cache to Redis	Config file	☐
03	Keep pending cache invalidations in Redis	Config file	☐
04	Compress the cart and the cache with zstd	Server	☐
QUEUE AND CRON			
05	Switch off the admin worker and run real workers	Server	☐
06	Run scheduled tasks from cron	Server	☐
07	Send mails through the queue	Config file	☐
08	Stop writing mail variables on every send	Config file	☐
09	Generate the sitemap from cron	Server	☐
CATALOGUE AND STOREFRONT			
10	Load reduced product data in listings	Administration	☐
11	Switch off the product stream indexer	Config file	☐
12	Turn on the Speculation Rules API	Administration	☐
13	Stop search from falling back to MySQL	Config file	☐
PHP AND DEPLOYS			
14	Tune OPcache for production	Server	☐
15	Dump the environment to .env.local.php	Deployment	☐
16	Make the Composer classmap authoritative	Deployment	☐

1. Audit

Tick what your store already has in the sheet below. The empty boxes are your to-do list.

2. Stage

Change one item at a time on staging, never straight on production.

3. Measure

Compare response times and server load before and after each change.

Cache

Serve more requests without PHP, and keep what is cached small.

01 Turn on the HTTP cache ☐

CONFIG FILE

Cached pages skip PHP entirely. On a store that has it off, this is usually the largest single gain.

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

WATCH OUT The built-in cache still lives in the app cache. Under real traffic, put Varnish or Fastly in front and let it hold the pages.

02 Move the app and system cache to Redis ☐

CONFIG FILE FROM 6.5.8.3

The file cache slows down under load and cannot be shared between app servers. Redis handles both.

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

WATCH OUT Tags that no longer point anywhere pile up and Redis memory keeps growing. Schedule a cleanup, for example `frosh:redis-tag:cleanup` from FroshTools.

03 Keep pending cache invalidations in Redis



CONFIG FILE

By default they queue up in a MySQL table. On a busy store with many writes at once, that table can deadlock.

```
# config/packages/shopware.yaml
shopware:
  redis:
    connections:
      ephemeral:
        dsn: 'redis://localhost:6379/0'
  cache:
    invalidation:
      delay_options:
        storage: redis
        connection: 'ephemeral'
```

WATCH OUT The `ephemeral` connection has to be defined under `shopware.redis.connections`, as above. Without it the setting has nothing to connect to.

04 Compress the cart and the cache with zstd



SERVER FROM 6.6.4.0

Smaller cart and cache entries use less Redis memory and move less data per request. Cache compression is already on by default; this changes the method.

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

WATCH OUT Install the zstd PHP extension first. It is not bundled with PHP. Before 6.7.12.0 the cache key is `cache_compression_method`. Clear the cache after switching.

Queue and cron

Background work belongs to workers and cron, not to whoever has the Administration open.

05 Switch off the admin worker and run real workers ☐

SERVER

The admin worker only works through the queue while someone has the Administration open. Close the tab and indexing, mails and imports stop moving.

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

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

WATCH OUT Run a worker on the `failed` transport as well. Keep the time and memory limits, so systemd or supervisor restarts the process cleanly.

06 Run scheduled tasks from cron ☐

SERVER

Cleanups and recurring indexing start on time instead of waiting for a worker to get round to them.

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

WATCH OUT Match the cron interval to your shortest task interval. Every 5 minutes is a sensible start.

07 Send mails through the queue ☐

CONFIG FILE

Checkout stops waiting on the mail server. The order confirmation leaves a few seconds later from a worker.

```
# config/packages/prod/framework.yaml
framework:
  mailer:
    message_bus: 'messenger.default_bus'
```

WATCH OUT Only once item 05 is live. Without workers, mails sit in the queue and nobody receives them.

08 Stop writing mail variables on every send ☐

CONFIG FILE

Every mail sent also stores example data for the template editor. Turning it off removes a database write from every order.

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

WATCH OUT The mail template editor in the Administration loses its example data for autocomplete.

09 Generate the sitemap from cron ☐

SERVER FROM 6.7.1.0

On a large catalogue the sitemap is a heavy job. Run it at night instead of inside the scheduled task queue.

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

# crontab, nightly
0 3 * * * php /var/www/html/bin/console sitemap:generate
```

WATCH OUT Disable the task and forget the cron entry, and the sitemap quietly stops updating.

Catalogue and storefront

Less work per product save and per page a shopper opens.

10 Load reduced product data in listings ☐

ADMINISTRATION FROM 6.7.12.0

Listings load only what the default product box shows, with a plain-text teaser of up to 512 characters instead of the full description.

```
Settings > Products
"Load reduced product data in listings"
config key: core.listing.partialDataLoading
```

WATCH OUT Themes and extensions that read other product fields in listings will show gaps. New installs have it on; existing stores switch it on per sales channel.

11 Switch off the product stream indexer ☐

CONFIG FILE FROM 6.6.10.0

Saving a product no longer re-indexes dynamic product groups. With large imports that adds up fast.

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

WATCH OUT Category pages built on a dynamic product group only update when the HTTP cache expires, and rule conditions that check for an item in a dynamic product group always return false.

12 Turn on the Speculation Rules API ☐

ADMINISTRATION FROM 6.6.10.0

Once a shopper hovers over or presses a link, the browser starts rendering the next page. The click lands on a page that is already built.

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

WATCH OUT Pages rendered but never opened still cost server time, and they can inflate page views in your analytics.

13 Stop search from falling back to MySQL ☐

CONFIG FILE

When OpenSearch or Elasticsearch fails, Shopware quietly runs the search against MySQL. On a large catalogue that fallback can overload the database.

```
# .env.local
SHOPWARE_ES_THROW_EXCEPTION=1
```

WATCH OUT Search failures now show up as errors. Put monitoring on the search cluster before you switch this on.

PHP and deploys

Settings that only pay off when your deploy script knows about them.

14 Tune OPcache for production ☐

SERVER

PHP stops checking every file for changes on every request and keeps more of Shopware compiled in memory.

```
; php.ini
opcache.validate_timestamps=0
opcache.max_accelerated_files=20000
opcache.interned_strings_buffer=20
zend.assertions=-1
realpath_cache_ttl=3600
```

WATCH OUT With timestamps off, PHP never picks up new code on its own. Reload PHP-FPM on every deploy. Preloading adds another 2 to 5%, but needs a full FPM restart after every cache clear and breaks the Extension Manager.

15 Dump the environment to .env.local.php ☐

DEPLOYMENT

Shopware reads one compiled PHP array instead of parsing the `.env` files on every request.

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

WATCH OUT The dumped file wins over every `.env` file. Dump again whenever a variable changes, ideally as a deploy step.

16 Make the Composer classmap authoritative ☐

DEPLOYMENT

The autoloader looks classes up in one map and never searches the file system.

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

WATCH OUT Only if every plugin is installed through Composer. Plugins in `custom/plugins` that Composer does not manage, or that came through the Extension Manager, stop loading.

NEED A HAND?

We can run this *audit* for you.

About half of these take 5 minutes. The rest, like workers, Redis and OPcache, depend on your hosting and break things quietly when set up wrong.

Send us your store URL and a line about your hosting. We reply within 24 hours with what we would change first.

elixentdigital.com/contact

[Book a 30-minute call](#)

Sources: Shopware developer documentation (developer.shopware.com/docs/guides/hosting/performance/ and [/hosting/infrastructure/](https://developer.shopware.com/docs/guides/hosting/infrastructure/)) and the Shopware 6.7.14.0 source code.