OPcache

OPcache for hosted Exponential sites: the settings under PHP-FPM, Velocity and FrankenPHP, why Velocity's differ, the one setting never to use, what JIT measured, and how to check and profile the cache.

Source file: doc/eng-us/help/hosting/performance/opcache.md Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/hosting/performance/opcache.md

This page is for system administrators at hosting companies who configure PHP for Exponential sites. It answers: which OPcache settings to use under each server, why Velocity's settings differ from PHP-FPM's, which setting breaks Exponential, what the project measured, and how to check the cache on a live site. The source is Chapter 10, section 10.12 and Velocity and the opcode cache.

Contents

1. Why it matters
2. Under PHP-FPM
3. Under Velocity
4. Under FrankenPHP
5. JIT
6. Checking and profiling
7. Checklist

Why it matters

The comments in `settings/velocity.ini` record a measurement on a content page with four workers: no opcode cache 526.6 ms (1.9 requests a second), OPcache 390.8 ms (2.6 requests a second), recorded there as +25.8 %. "So: the cache alone is the win." Timestamp checks cost nothing measurable, and the tracing JIT was slower than no JIT.

Under PHP-FPM

Enable OPcache in the pool's `php.ini` (chapter 10):

INI

```
opcache.enable=1
opcache.memory_consumption=256
opcache.max_accelerated_files=20000
opcache.interned_strings_buffer=32
; keep timestamp checks on: Exponential rewrites PHP cache files (INI caches,
; compiled templates, expiry.php) under the same name
opcache.validate_timestamps=1
opcache.revalidate_freq=2
```

> **Warning**
> Never set `opcache.validate_timestamps=0` on an Exponential site: a cache file rewritten after a clear would be served in its old form until PHP restarts.

With timestamp checks on, `exp:velocity deploy` (with `[DeploySettings] PhpFpmReload=auto`) can wait `revalidate_freq` seconds for changed files instead of reloading the FPM master, which spares the other sites of that master (PHP-FPM tuning).

Under Velocity

Velocity passes its own PHP options to the server process (`settings/velocity.ini [PHPSettings] IniOptions[]`), because the command-line PHP may have OPcache off system-wide:

Option	Shipped	Why
`opcache.enable_cli`	`1`	the server is a command-line process
`opcache.memory_consumption`	`256`	MB
`opcache.max_accelerated_files`	`20000`	
`opcache.interned_strings_buffer`	`32`	
`opcache.revalidate_freq`	`0`	check every include
`opcache.file_update_protection`	`0`	see below
`apc.shm_size`	`256M`	APCu for the response cache and the query cache

OPcache compares a file's time against the **request's start time**, and a Velocity worker is one long request that started when the worker did:

- with `revalidate_freq` at its default of 2, a worker never checked a file again, so a rewritten cache file was served as it was when the worker first compiled it;
- with `file_update_protection` at its default, every file written after the worker started (INI caches, override caches, compiled templates, all regenerated after a deploy or a clear) counted as "too new" for the worker's whole life and was compiled again on every include. Measured on the demonstration server: the cache served 2.7 % of includes under Velocity against 97 % under PHP-FPM, and a rendered page cost 644 ms of CPU against 166 ms.

The server itself reads an include whole and drops the cached copy of a file changed in the last two seconds, so nothing half written is served. No setting needs to change; a restart applies the shipped values. Chapter 10 sums it up: "Any PHP setting that compares against 'the request time' behaves as frozen in a Velocity worker; suspect that first when something works under PHP-FPM and not under Velocity."

Under FrankenPHP

The embedded PHP reads no machine `php.ini`. It gets `opcache.enable=1` from `[FrankenPHPSettings] IniOptions[]` and the tuning above from `[PHPSettings] IniOptions[]`. `revalidate_freq=0` is harmless there, because each request has its own request time (FrankenPHP).

JIT

Measured in `settings/velocity.ini`: tracing JIT 416.7 ms against 390.8 ms for OPcache alone on the same page, so "Tracing JIT is slower than no JIT on this workload and is deliberately absent." The FrankenPHP page reports the same. If a customer asks for JIT, measure their site with and without it before you change a plan default (Benchmarking a hosted site).

Checking and profiling

Shell

```
grep -n "opcache\." settings/velocity.ini                    # Velocity's shipped options
php bin/php/cache.php opcache status --allow-root-user       # the cache of the process that ans
wers
```

Setup > Caches has **Reset OPcache** and **Empty APCu** for the server process that answered the page; the other server keeps its own (Caches).

To see how requests use the cache, switch on the opcode cache profile. It is off unless a marker file exists; while it exists every request appends one JSON line to `var/tmp/opcache_profile.log`:

Shell

```
touch var/tmp/opcache_profile.on
curl -s -o /dev/null https://example.com/
tail -n 3 var/tmp/opcache_profile.log
```

Read `included_not_cached` on the second and later requests of a process: close to 0 is healthy, many means files are compiled again on every include. `opcache_ini` shows the settings the process runs with. Switch it off again by moving the `.on` file out of `var/tmp/`; the log grows with every request while it is on.

Checklist

- ☐ OPcache enabled in every PHP-FPM pool; `validate_timestamps=1`.
- ☐ Velocity's shipped `[PHPSettings] IniOptions[]` left in place.
- ☐ JIT changed only after measuring.
- ☐ The profile switched off after use.

Related pages

- PHP-FPM tuning
- Velocity workers and pools
- Caching layers
- Velocity and the opcode cache
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.