OPcache and cache preloading

Two different kinds of "preloading" a PHP developer meets in Exponential - the opcode cache (OPcache) and the settings that matter under PHP-FPM and Velocity, and the site cache preloader bin/php/preload.php...

Source file: `doc/eng-us/help/php-developers/opcache-and-cache-preloading.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/opcache-and-cache-preloading.md

This page is for PHP developers who deploy code or care about the first request after a deploy. Two things are called "preloading" around Exponential, and they are different:

- **OPcache** keeps compiled PHP in memory, so files are not parsed again on every request.
- **The site cache preloader** (`bin/php/preload.php`, `exp:preload`) requests the pages of a site before visitors do, so the page view cache, the image aliases and the compiled templates exist.

PHP's own `opcache.preload` setting is not used by Exponential; nothing in the shipped settings configures it.

Contents

1. OPcache under PHP-FPM
2. OPcache under Velocity
3. What a code change needs
4. The site cache preloader
5. The cache warmer
6. After a deploy

OPcache under PHP-FPM

Use the PHP defaults or your host's settings; Exponential needs nothing special. Two points:

- With `opcache.validate_timestamps=0` (common in production), changed PHP files are not seen until PHP-FPM is reloaded. Reload the pool that actually serves the site after every deploy of PHP code.
- Generated PHP files (the INI cache, compiled templates, the class maps in `var/autoload/`) are written at runtime; OPcache picks them up like any other file.

OPcache under Velocity

Velocity runs PHP's command line binary, where OPcache is off unless asked for. `settings/velocity.ini` `[PHPSettings]` `IniOptions[]` therefore ships:

Option	Why
`opcache.enable_cli=1`	turn OPcache on for the server process; measured 18 to 26 percent faster

Option	Why
`opcache.memory_consumption=256`, `opcache.max_accelerated_files=20000`, `opcache.interned_strings_buffer=32`	room for the kernel, the extensions and the compiled templates
`opcache.revalidate_freq=0`	check timestamps on every include
`opcache.file_update_protection=0`	a persistent worker's request time is its start time, so every file written later (INI caches, compiled templates) counted as "too new" and was compiled on every include; with `0`, rendered pages cost about a quarter of the CPU they did
`apc.shm_size=256M`	room for the response cache and the SQL query cache in APCu

Read the full list in `settings/velocity.ini` and the background in Velocity: running Exponential in a persistent-worker web server.

What a code change needs

Change	Under PHP-FPM	Under Velocity
a PHP file edited	reload PHP-FPM if timestamps are not validated	`exp:velocity deploy`: workers keep the classes they loaded
a class added or renamed	autoloads, then reload	`exp:velocity deploy` (`--kernel` for a kernel class)
a kernel file with the engine archive on	(not applicable)	the archive is rebuilt on restart when needed (The engine archive)
a template edited	nothing, or clear template caches	`exp:velocity cache clear` for the response cache

The site cache preloader

Shell

```
php bin/php/preload.php --apply                                    # the default siteaccess
php bin/php/console exp:preload --apply --siteaccess=<siteaccess>
```

A run starts at the section pages of a site, follows its links up to a page limit and a link depth, and requests them as an anonymous visitor. It warms the page view cache, the image aliases and the compiled templates, which are files shared by PHP-FPM and Velocity. One run at a time holds a lock; every run is recorded with its broken links. **Setup > Preload Sites** starts the same run in the background and shows it (Preloading caches, Site cache preloader).

The cache warmer

`exp:warm` (`bin/php/warm.php`) walks the published content as an anonymous visitor, so nobody pays for the first render after an expiry. It complements Velocity's response cache and the role-aware HTTP cache (HTTP/2 and cache warming).

After a deploy

1. Regenerate autoloads and clear the caches the change needs (or `exp:velocity deploy`, which does it in order).
2. Reload PHP-FPM, or restart Velocity's workers.
3. Clear the caches that hold rendered output **after** the reload, so a page rendered by old code in between is not kept.
4. Warm: `exp:preload` or `exp:warm`.

Related pages

- Velocity and the opcode cache
- Deploying guide
- Caches guide
- 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.