Servers and deployment: Apache and PHP-FPM, Velocity, FrankenPHP

How Exponential is served and deployed, for developers who know PHP-FPM behind nginx, Laravel Octane, RoadRunner, FrankenPHP worker mode, Gunicorn or Puma - the three ways to serve it, the persistent-worker rules, the...

Source file: `doc/eng-us/help/web-framework-developers/servers-and-deployment.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/web-framework-developers/servers-and-deployment.md

This page is for developers who deploy framework applications to PHP-FPM, to an application server (Laravel Octane, RoadRunner, FrankenPHP worker mode) or to Gunicorn, Puma or a servlet container. Exponential runs under three servers, all driven by `./console exp:velocity` and `settings/velocity.ini`, and ships one command that performs every step a code change needs (Servers: Velocity, PHP-FPM and FrankenPHP).

Contents

1. Three ways to serve
2. Rules for persistent workers
3. Deploying a change
4. What moves how
5. A release, step by step
6. Compared with other stacks

Three ways to serve

Server	Engine value	Process model
Apache or nginx with PHP-FPM	(not an engine)	one request per PHP process run, nothing kept
Exponential Velocity	`qbix` (recommended)	a parent loads the application once, forks persistent workers that serve many requests and are reset after each; a response cache answers repeat public requests
FrankenPHP	`frankenphp`	Caddy with PHP built in, one binary

As of 6.0.15 Velocity is part of Exponential: `composer.json` requires `se7enxweb/exponential-velocity`, so every installation has it. `velocity.ini` `[ServerSettings] Engine` ships as `php` (PHP's built-in server, for development); any real site sets `qbix` (Velocity engines, Serving the site, Deploying guide).

Rules for persistent workers

- No per-request state in static properties or globals without a reset.
- No `exit` for flow control except `eZExecution::cleanExit()`, and never catch its exception.
- Read input through `eZHTTPTool`, which the kernel prepares per request, not `$_POST` directly.
- Event listener registrations must be idempotent.
- A changed class or setting reaches the workers only after a restart; a new kernel class needs `--kernel`.

These are the same rules Octane or RoadRunner impose; write code that keeps them and it behaves the same under PHP-FPM (Velocity: the persistent-worker server).

Deploying a change

Shell

```
./console exp:velocity deploy --dry-run --allow-root-user   # what it would do
./console exp:velocity deploy --allow-root-user             # the usual case
./console exp:velocity deploy --kernel --allow-root-user    # a kernel class was added or renamed
```

In order: extension autoloads (`--kernel`: kernel too), INI and template caches, the engine archive when kernel files changed (every file parsed first, so a broken file stops the deploy before any server is touched), a PHP-FPM reload when needed, the Velocity restart, then content, HTTP and template-block caches and Velocity's response cache last, so a page rendered by old code is never cached again. It prints PASS, FAIL or SKIP per step and stops at the first failure (Deploying with Velocity). `--packer` also clears packed stylesheets and scripts.

What moves how

From A deployment pipeline:

Thing	Moves by
Kernel and extensions	Git and Composer (`composer.lock` committed in the project)
Your extension	its Git tag, required by version
Settings	Git, without secrets; secrets set per machine
Content classes	packages or upgrade scripts
Tables	your install and upgrade commands (Migrations)
Content	stays in production; copy production to staging, never the other way

A release, step by step

1. Tests pass in CI.
2. Deploy to staging, run the upgrade steps, test.
3. Back up production.
4. Maintenance mode if the release changes data (`php bin/php/maintenance.php on --allow-admin`).

5. Update the code (Git or `composer update <package>`), run the upgrade steps.
6. `./console exp:velocity deploy` (`--kernel` when kernel classes changed).
7. Maintenance off; verify with `exp:velocity status`, `exp:cache status`, the pages and the error log.

Rollback is the previous tag plus a backup restore when upgrade steps changed data.

Compared with other stacks

Stack	Typical deploy	Exponential
Symfony, Laravel on PHP-FPM	`composer install`, `cache:clear`, migrations, FPM reload	`composer`, your upgrade commands, `exp:velocity deploy`
Laravel Octane, RoadRunner	`octane:reload`	`exp:velocity deploy` (Velocity restart included)
Django, Rails	migrate, collect assets, restart Gunicorn or Puma	the same shape

Related pages

- Velocity is part of Exponential 6.0.15
- FrankenPHP as the Exponential web engine
- Scaling applications
- 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.