Static analysis with PHPStan

How Exponential uses PHPStan - level 1 over kernel/ and lib/ with a baseline of existing findings, what is excluded and why, running it locally and in CI, reading a finding, fixing it instead of baselining it, and...

Source file: `doc/eng-us/help/php-developers/static-analysis-with-phpstan.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/static-analysis-with-phpstan.md

This page is for PHP developers who commit to Exponential or want the same safety net for an extension. PHPStan reads the code without running it and reports calls to methods that do not exist, wrong argument counts, undefined variables and similar mistakes. In Exponential it runs at level 1 with a baseline, so only findings that your change adds fail.

Contents

1. The configuration
2. Running it
3. The baseline
4. Reading and fixing a finding
5. What is excluded and why
6. Analysing your extension

The configuration

`phpstan.neon.dist`:

Parameter	Value	Meaning
`includes`	`phpstan-baseline.neon`	the known findings, ignored
`level`	`1`	undefined variables, unknown classes and methods, wrong argument counts and the checks of level 0
`phpVersion`	`80100`	the code is analysed as PHP 8.1
`paths`	`kernel`, `lib`	what is analysed
`excludePaths`	`vendor/*`, `var/*`, `*/tests/*`, `kernel/private/classes/commands/*`, `.../dfs_filter_iterator.php`	see below
`bootstrapFiles`	`autoload.php`	the class maps, so PHPStan finds the kernel's classes
`tmpDir`	`.devtools/phpstan`	the result cache
`treatPhpDocTypesAsCertain`	`false`	docblock types guide the analysis, but a defensive `is_array()` on a value documented as an array is not an error

Since the code has no type declarations, PHPStan's type knowledge comes from docblocks; good `@param` and `@return` lines make the analysis better (Why the code has no type declarations).

Running it

Shell

```
make devtools          # once: PHPStan into .devtools/ (composer.dev.json), never into vendor/
make phpstan           # .devtools/vendor/bin/phpstan analyse -c phpstan.neon.dist --memory-limit
=4G
```

The pre-push hook runs the same command, so a push with a new finding is refused unless you use `--no-verify` (Git hooks and make targets). In CI the **PHPStan** job of the Quality workflow runs on PHP 8.3 and blocks.

The baseline

`phpstan-baseline.neon` lists every finding the code had when the check was introduced (3033 at the time), each with its message, identifier, count and path. A finding in the baseline is ignored; a new one fails. Two rules:

- **Fix new findings; do not add them to the baseline.** `make phpstan-baseline` accepts all current findings and is meant for the rare change that raises the level or reorganizes the baseline, in a change of its own.
- **Shrink the baseline when you are in the file anyway.** When you fix a baselined finding, PHPStan stops matching that entry; `reportUnmatchedIgnoredErrors: false` keeps that from failing, and a later baseline regeneration removes the entry.

Reading and fixing a finding

A finding names the file, the line, the message and an identifier:

```
------ ------------------------------------------------------------
 Line   extension/acmereport/classes/acmereportservice.php
------ ------------------------------------------------------------
 42     Call to an undefined method acmeReportItem::fetchByOwner().
        method.notFound
------ ------------------------------------------------------------
```

Common message	Usual cause	Fix
`Variable $x might not be defined`	assigned only in one branch	initialize it before the branch
`Call to an undefined method`	a typo, or a method of another class	use the right method; check `kernel/classes/`
`... invoked with 3 parameters, 2 required`	the signature changed	adapt the call
`Access to an undefined property`	a dynamic property	declare it in the class
`Instantiated class X not found`	a missing class map entry or a typo	regenerate autoloads; fix the name

Many baselined findings of the form `Variable $Params might not be defined` come from view files, which receive `$Params` from `eZProcess::runFile()`. The class-based views (`Exponential\Runnable\ModuleView`) take them from `$scope` instead and do not have the problem (Module views as classes).

What is excluded and why

Path	Reason (from `phpstan.neon.dist`)
`vendor/*`, `var/*`	third-party and generated code
`*/tests/*`	tests are checked by running them
`kernel/private/classes/commands/*`	command classes that also declare global helper functions; loading two in one analysis process redeclares them
`kernel/private/classes/clusterfilehandlers/dfsbackends/dfs_filter_iterator.php`	its iterator methods lack PHP 8.1's tentative return types or `#[\ReturnTypeWillChange]`, which PHPStan cannot baseline

Analysing your extension

PHPStan is not limited to the kernel. With the development tools installed, point it at your extension with a small configuration of your own that bootstraps the kernel's autoloader:

```
# extension/acmereport/phpstan.neon
parameters:
    level: 1
    paths:
        - .
    excludePaths:
        - tests/*
    bootstrapFiles:
        - ../../autoload.php
```

Shell

```
.devtools/vendor/bin/phpstan analyse -c extension/acmereport/phpstan.neon --memory-limit=2G
```

Regenerate the autoloads first (`php bin/php/ezpgenerateautoloads.php -e`) so your own classes are known. Start at level 1 like the kernel and raise the level when the findings are gone.

Related pages

- Quality checks
- The coding standard (phpcs.xml.dist)
- CI and reproducing a red check
- 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.