PHPUnit without a database

How Exponential's PHPUnit setup works for PHP developers - phpunit.xml, tests/bootstrap.php, the test suites and excluded groups, PHPUnit 10 to 13 across PHP 8.1 to 8.5, writing a DB-less test for kernel or extension...

Source file: `doc/eng-us/help/php-developers/phpunit-without-a-database.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/phpunit-without-a-database.md

This page is for PHP developers who write unit tests for kernel code or extensions. Most of Exponential's tests run without a database, an installed site or a web server, in seconds, on any machine and in CI on PHP 8.1 to 8.5. The application creators' page shows a first test (DB-less PHPUnit tests); this page explains the setup underneath.

Contents

1. The configuration
2. The bootstrap
3. Suites and groups
4. PHPUnit versions
5. Writing a test
6. Settings in a test
7. State between tests
8. Running only what you changed

The configuration

`phpunit.xml` at the root sets:

Setting	Value
bootstrap	`tests/bootstrap.php`
test suites	`security`, `kernel-classes`, `kernel-content`, `kernel-datatypes`, `kernel-search`, `kernel-private`, `lib`, `mongodb`, `ezjscore`, `expservices`, `ezoe`, `quality`, `design` and more
excluded groups	`database`, `mail-live`, `mongodb-live`, `network-live`
source	`kernel/` and `lib/`
environment	`EZP_TEST_ENVIRONMENT=1`, `memory_limit=1G`

Shell

```
php vendor/bin/phpunit                                    # every suite, without the excluded group
s
php vendor/bin/phpunit --testsuite kernel-classes         # one suite
php vendor/bin/phpunit tests/tests/lib/ezutils/           # one directory
php vendor/bin/phpunit --filter testRefusesNull <file>    # one test
php vendor/bin/phpunit --list-tests                       # what would run
```

The bootstrap

`tests/bootstrap.php` loads Composer's autoloader (which loads `autoload.php` through `composer.json`'s `files`), turns on the test class map `var/autoload/ezp_tests.php` (`ezpAutoloader::enableTests()`, generated with `php bin/php/ezpgenerateautoloads.php -s`), prepares PHPUnit's isolated-process children and gives the run its own audit directory under `var/tmp`. It does **not** boot the kernel: no siteaccess, no database connection, no session. Code that needs only classes and settings runs; code that queries the database does not.

Suites and groups

- **Suites** group the test files by directory under `tests/tests/` (`tests/tests/kernel/classes/`, `tests/tests/lib/` ...) and by extension.
- **Groups** mark tests that need something outside the process. Tag a test with `@group database` (or a `#[Group('database')]` attribute) when it needs a database; the normal run skips it, and the legacy runner runs it (Database and live tests).

PHPUnit versions

`composer.json` allows `^9.6 || ^10.5 || ^11.5 || ^12.0 || ^13.4`. CI resolves PHPUnit per leg: 10.5 on PHP 8.1, 11.5 on 8.2 and 8.3, 13 on 8.4 and 8.5. A test must work on all of them:

- Use attributes or annotations the oldest version in use understands, or both.
- Avoid assertions added in recent PHPUnit releases unless you check for them.
- Template methods (`setUp()`, `tearDown()`) declare `: void`, which PHPUnit requires; it is the one return type the coding standard allows, and only under `tests/`.

The migration notes are in PHPUnit 10 and PHPUnit 13.

Writing a test

PHP

```php
<?php
// tests/tests/kernel/classes/acmeReportServiceTest.php, or extension/acmereport/tests/acmeReport
ServiceTest.php
class acmeReportServiceTest extends PHPUnit\Framework\TestCase
{
    public function testTotalsAddAmountsPerStatus()
    {
        $items = array( new acmeReportItem( array( 'status' => 'open', 'amount' => 2 ) ),
                        new acmeReportItem( array( 'status' => 'open', 'amount' => 3 ) ) );
        $this->assertSame( array( 'open' => 5 ), acmeReportService::totals( $items ) );
    }

    public function testTotalsOfNothingIsEmpty()
    {
        $this->assertSame( array(), acmeReportService::totals( array() ) );
    }
}
```

A class extending `eZPersistentObject` can be constructed from an array without a database, as above; only `store()`, `fetchObject()` and friends need one. Keep the logic you want to test in methods that take values and return values, and the database calls in thin methods around them.

Settings in a test

`tests/toolkit/ezpinihelper.php` sets INI values in-process and restores them:

PHP

```php
public function setUp(): void
{
    ezpINIHelper::setINISetting( 'site.ini', 'DebugSettings', 'DebugOutput', 'disabled' );
}

public function tearDown(): void
{
    ezpINIHelper::restoreINISettings();
}
```

A test that changes an INI value and does not restore it changes the next test's settings. That is a classic cause of a test that passes alone and fails in the whole run.

State between tests

Many kernel classes keep static caches (settings, objects, the current user). PHPUnit runs every test of a run in one process, so the same discipline applies as under Velocity's persistent workers (Static state traps): reset what you changed, and do not rely on order. When two files fail only together, run them together to find the pair.

Running only what you changed

`tests/test-map.json` maps source files to the tests that cover them: hand-made entries (`map`), entries generated from test files that name source paths literally (`generated`), and files no test covers on purpose (`ignore`). The team's fast push gate runs the tests of a change from it. If you add a test for a file it does not name literally, add a `map` entry (Quality checks).

Related pages

- Testing and CI standard
- Tests for extensions
- Testing 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.