# Code Quality

Code Quality Tools

Anonymous · 09/09/2026

# Code Quality Tools

Derafu Foundation comes with pre-configured code quality tools to ensure consistent code style and identify potential issues early in development. This document explains how to use these tools effectively.

## PHP CS Fixer

PHP CS Fixer is a tool that automatically fixes PHP coding standards issues in your code according to rules you define.

### Configuration

The configuration is defined in `php-cs-fixer.php` at the root of your project. Key features include:

- PSR-12 coding standards by default.
- Strict type declarations enabled.
- Automatic array syntax conversion to short syntax.
- Ordered imports.
- Modern PHP features optimization (e.g., arrow functions).
- PHPUnit strict assertions.

### Usage

Run PHP CS Fixer to check code style issues:

```
vendor/bin/php-cs-fixer fix --dry-run --diff
```

Fix code style issues automatically:

```
vendor/bin/php-cs-fixer fix
```

# PHPStan

PHPStan is a static analysis tool that finds bugs in your code without running it. It's focused on finding errors in code logic.

## Configuration

The configuration is defined in `phpstan.neon` at the root of your project. By default, it:

- Sets analysis level to 5 (out of 9)
- Analyzes code in the `src` and `tests` directories

## Levels Explained

- Level 1: Basic checks.
- Level 2: Possibly undefined variables.
- Level 3: Return types, phpdocs.
- Level 4: Type hints.
- Level 5: Basic dead code detection.
- Level 6: Detecting unreachable code.
- Level 7: Union types.
- Level 8: More precise analysis
- Level 9: Mixed type detection

## Usage

Run PHPStan analysis:

```
vendor/bin/phpstan analyse
```

# PHPUnit

PHPUnit is the standard testing framework for PHP applications.

## Configuration

The configuration is defined in `phpunit.xml` at the root of your project. Key features include:

- Test coverage reporting enabled.
- Strict mode enabled (fails on warnings, notices, etc.).
- Test results output to `var/tests-coverage.txt` and `var/tests-coverage.xml`.
- Test documentation output to `var/tests-testdox.txt`.

### Directory Structure

- `tests/src/`: Unit tests for your source code.
- `tests/fixtures/`: Test data and fixtures.

### Usage

Run the test suite:

```
vendor/bin/phpunit
```

Generate test coverage reports:

```
vendor/bin/phpunit --coverage-html var/coverage
```

## Integration with CI/CD

These tools are integrated into the CI workflow (`.github/workflows/ci.yml`), which automatically runs on each push to the repository. This ensures that code quality standards are maintained throughout development.

## Best Practices

1. **Run tools locally before committing**: This helps catch issues before they enter the codebase.
2. **Gradually increase PHPStan level**: Start with level 5 and work toward higher levels as your project matures.
3. **Aim for high test coverage**: Write tests for all critical code paths.
4. **Update rules as needed**: Customize tool configurations to match your project's specific requirements (remember to deactivate the `Derafu\\Foundation\\Installer::copyFiles` from the `post-install-cmd` and `post-update-cmd` scripts in your `composer.json` file).

---

Last updated on 09/09/2026