Build a PHP Safety Net

Streamline and Safeguard: Automated Checks Before You Commit

AARON HOLBROOK, 2023

ZEEK

Why?

Why Have a Safety Net?

- Cleaner, more consistent, safer code
- Unified coding standard is auto-applied
- Automatically perform static analysis of code and help PREVENT an entire range of bugs
- Automatically run unit, integration or acceptance tests

ZEEK

AARON HOLBROOK

Principal Engineer at Zeek: Specializing in Solving Problems

Over 20 years of PHP experience

Public Speaker & Workshop Leader

Driven by Efficiency & Problem-Solving

A Lifelong Builder: Digital & Physical

Your Debugging Expert for the Day

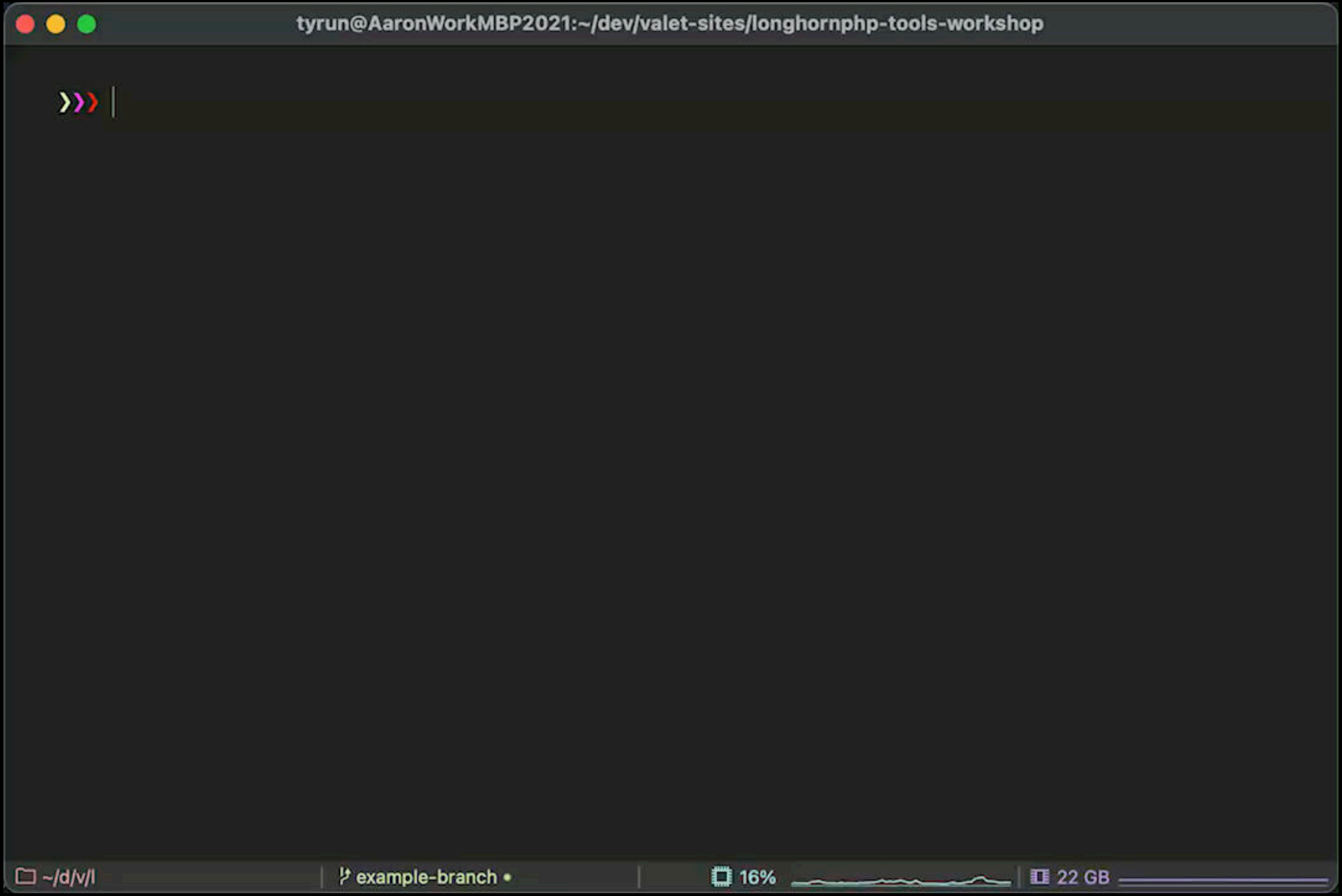
Prerequisites: Developer Workflows

- **Bash/Shell Terminal**: Ensure you have access to a Bash or Shell terminal. Windows users may consider using WSL or Git Bash.
- **PHP Locally Installed**: Make sure you have PHP installed on your local machine. We will be running various PHP-based commands. PHP 8.2 is recommended.
- **Composer**: This package manager for PHP is essential for some of the tools we'll be using. You can download it here (https://getcomposer.org).
- **GitHub Account**: If you don't have a GitHub account yet, please create one as we will be working with Git repositories (and automating GitHub Actions).
- **SSH Keys**: Generate an SSH private/public key pair if you haven't already. This is crucial for secure communication with GitHub. Here's a guide on how to do this (https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent).
- **GitHub Authentication**: Make sure you're locally authenticated with GitHub using your SSH keys. This will allow us to easily clone repositories and push changes.
- **GNU Make** (Command-Line Utility Installed): GNU Make is a build automation tool that we'll be using to manage and streamline various tasks in our PHP project. Here's how to install it based on your operating system:
  - Windows: You can install GNU Make through Cygwin or WSL (Windows Subsystem for Linux).
  - Linux: Generally available by default. If not, you can install it using the package manager for your specific distro, usually with a command like sudo apt-get install make for Debian-based distributions or sudo yum install make for Red Hat-based distributions.
  - Mac: It can be installed using Homebrew with the command brew install make.

ZEEK

What Does it Look Like in Action?

ZEEK



```
tyrun@AaronWorkMBP2021:~/dev/valet-sites/longhornphp-tools-workshop

❯❯❯ |

~/d/v/l    example-branch •    16%    22 GB
```

github.com/ZeekInteractive/longhornphp-tools-workshop

HANDS ON!

github.com/ZeekInteractive/longhornphp-tools-workshop

PHP Quality Tools

PHP Quality Tools

- PHP CS Fixer (for automatic code styling fixes)
- PHP Linter (for syntax checking)
- PHP Mess Detector (detect code smells and possible errors)
- PHPStan (static analyzer that looks at code typing and logic issues)
- Pest / PHPUnit
- Rector (automated refactoring)

ZEEK

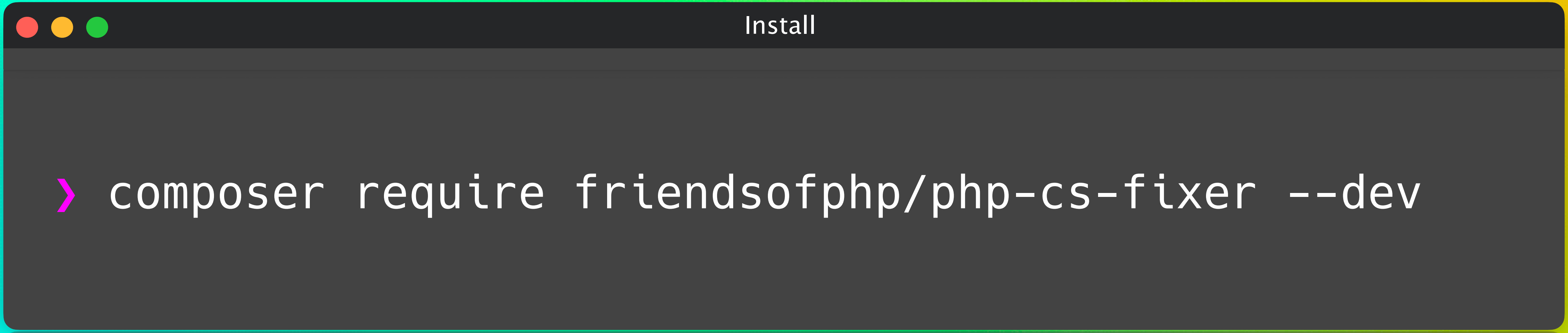
PHP CS Fixer

A tool to automatically fix PHP Coding Standards issues

The PHP Coding Standards Fixer (PHP CS Fixer) tool fixes your code to follow standards.

You can also define your (team's) style through configuration.

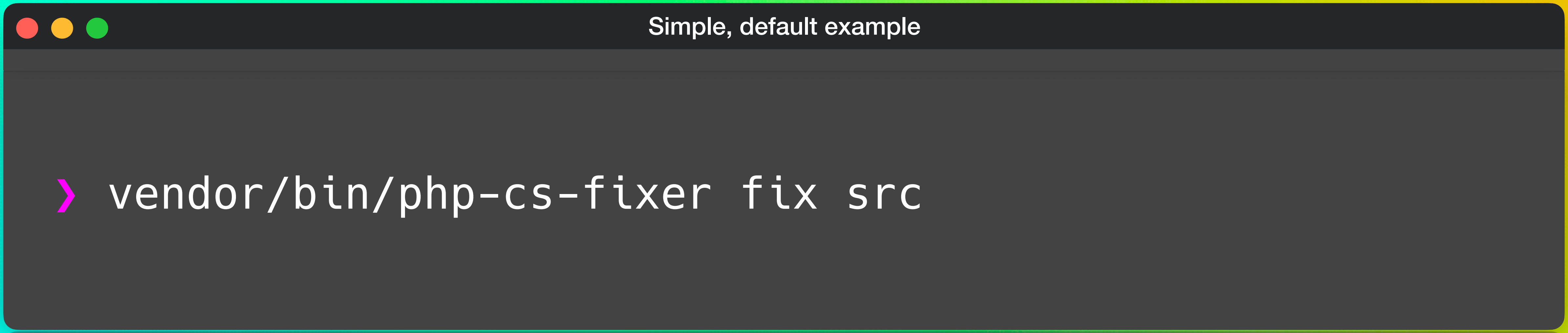
https://github.com/PHP-CS-Fixer/PHP-CS-Fixer



Install

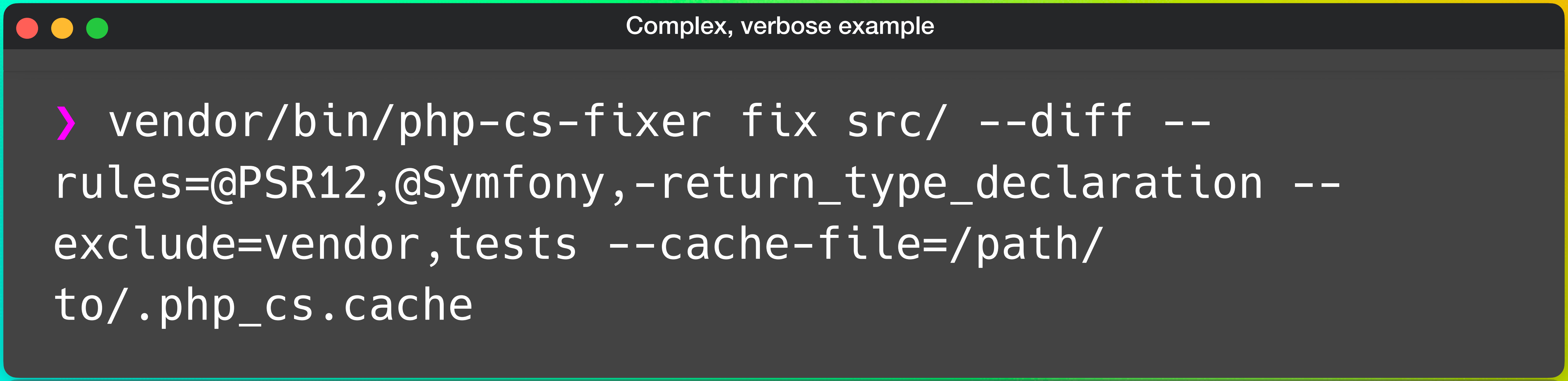
```
❯ composer require friendsofphp/php-cs-fixer --dev
```

https://github.com/PHP-CS-Fixer/PHP-CS-Fixer


Simple, default example

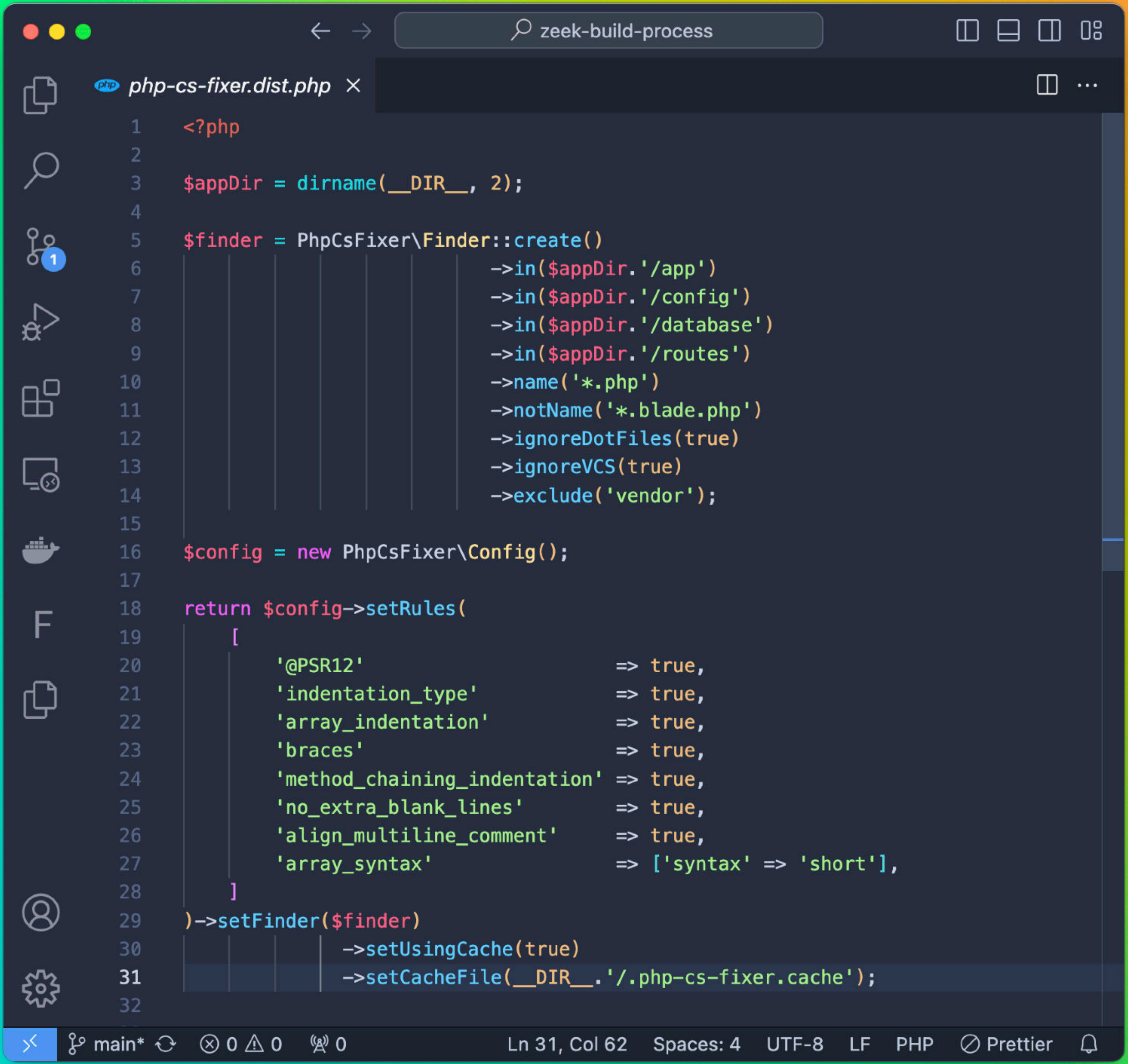
```
❯ vendor/bin/php-cs-fixer fix src
```

https://github.com/PHP-CS-Fixer/PHP-CS-Fixer


Complex, verbose example

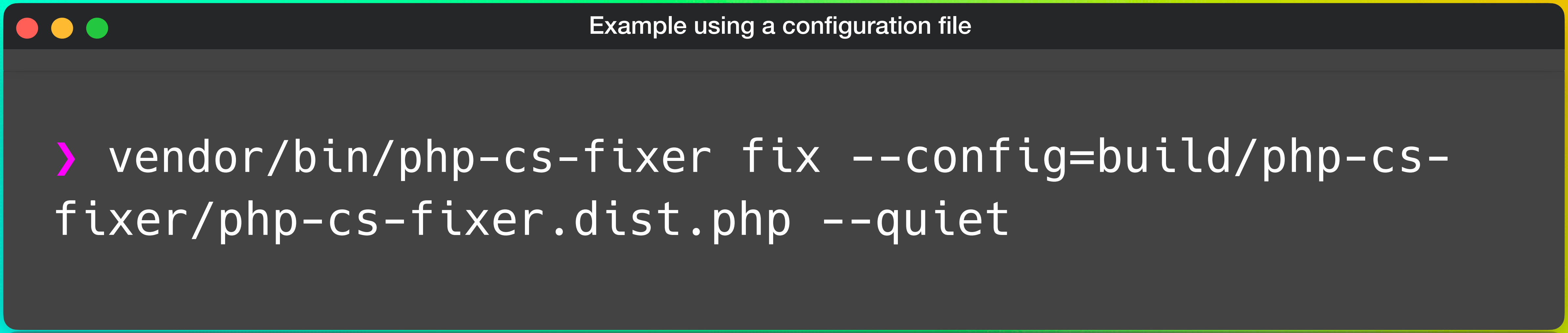
```
❯ vendor/bin/php-cs-fixer fix src/ --diff --rules=@PSR12,@Symfony,-return_type_declaration --exclude=vendor,tests --cache-file=/path/to/.php_cs.cache
```

https://github.com/PHP-CS-Fixer/PHP-CS-Fixer



```php
<?php

$appDir = dirname(__DIR__, 2);

$finder = PhpCsFixer\Finder::create()
                            ->in($appDir.'/app')
                            ->in($appDir.'/config')
                            ->in($appDir.'/database')
                            ->in($appDir.'/routes')
                            ->name('*.php')
                            ->notName('*.blade.php')
                            ->ignoreDotFiles(true)
                            ->ignoreVCS(true)
                            ->exclude('vendor');

$config = new PhpCsFixer\Config();

return $config->setRules(
    [
        '@PSR12'                      => true,
        'indentation_type'            => true,
        'array_indentation'           => true,
        'braces'                      => true,
        'method_chaining_indentation' => true,
        'no_extra_blank_lines'        => true,
        'align_multiline_comment'     => true,
        'array_syntax'                => ['syntax' => 'short'],
    ]
)->setFinder($finder)
            ->setUsingCache(true)
            ->setCacheFile(__DIR__.'/.php-cs-fixer.cache');
```

https://github.com/PHP-CS-Fixer/PHP-CS-Fixer


Example using a configuration file

```
❯ vendor/bin/php-cs-fixer fix --config=build/php-cs-fixer/php-cs-fixer.dist.php --quiet
```

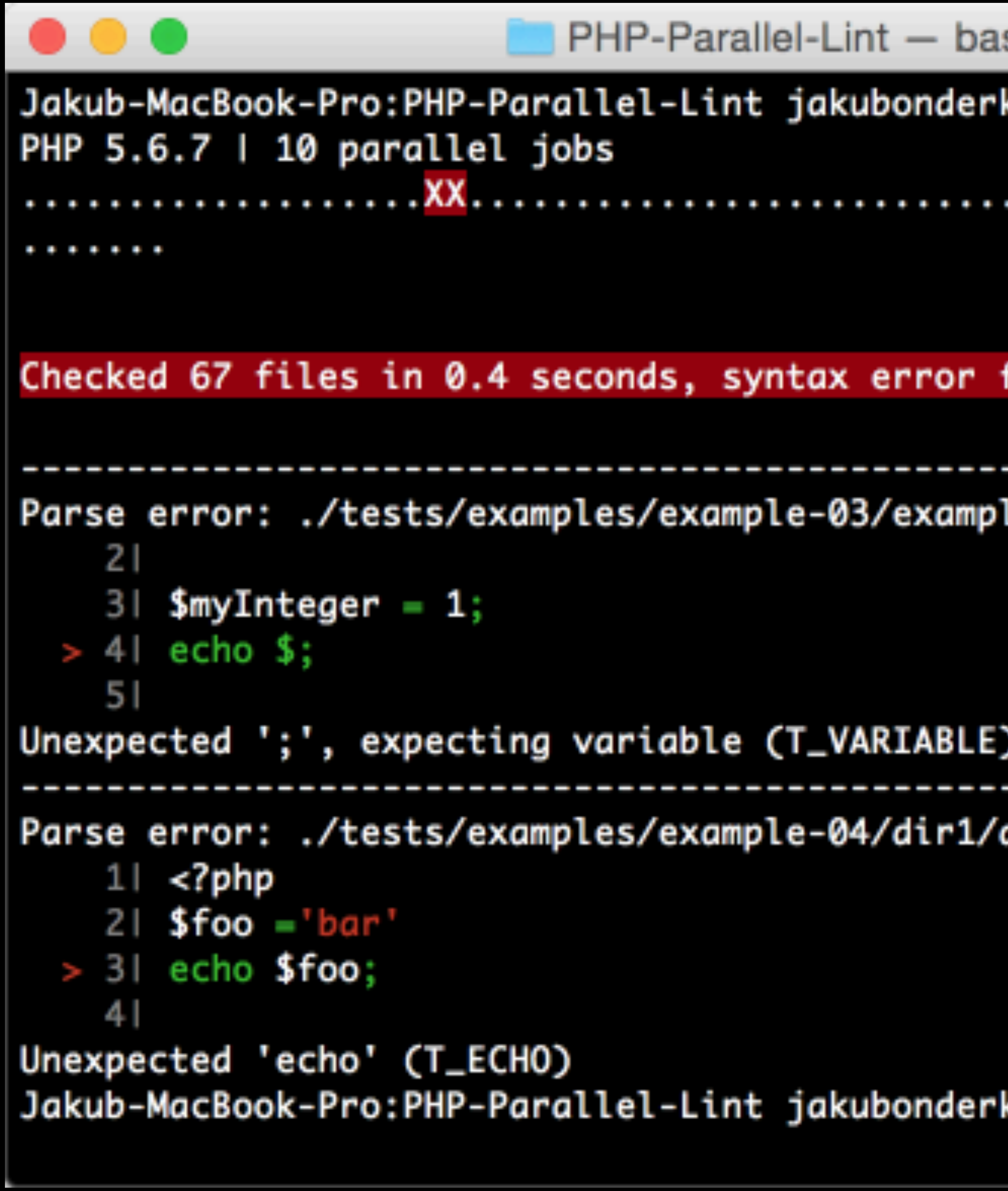
https://github.com/PHP-CS-Fixer/PHP-CS-Fixer

PHP Parallel Linter

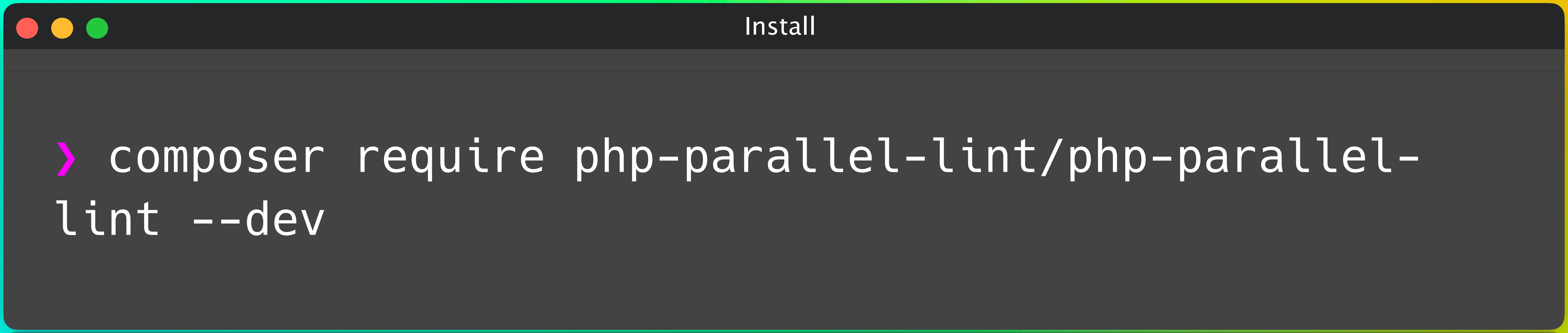
This application checks the syntax of PHP files in parallel

Linting's purpose is to identify syntax errors in PHP files.

Syntax errors are basic mistakes in the code that prevent it from running, like missing semicolons or mismatched brackets.

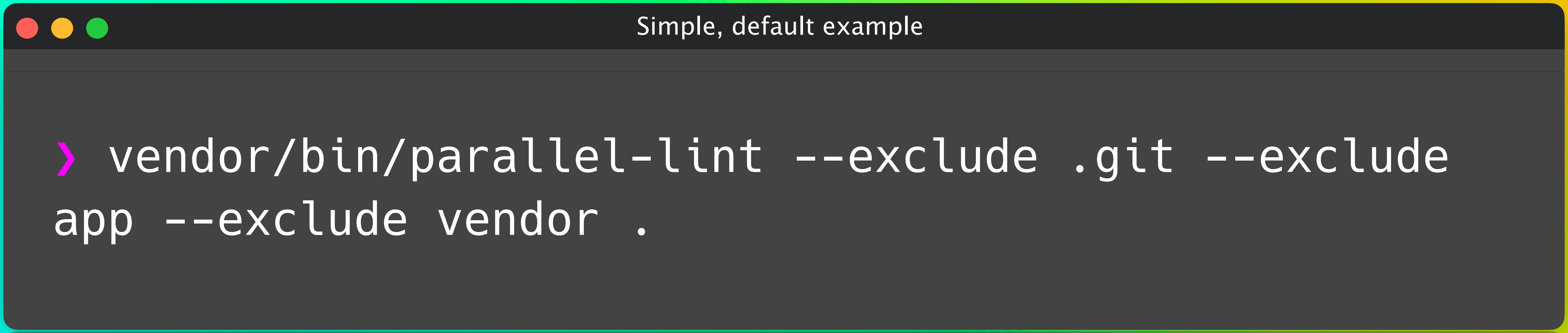


```
Jakub-MacBook-Pro:PHP-Parallel-Lint jakubonder
PHP 5.6.7 | 10 parallel jobs
....................XX.........................
.......


Checked 67 files in 0.4 seconds, syntax error 

-----------------------------------------------
Parse error: ./tests/examples/example-03/examp
    2|
    3| $myInteger = 1;
  > 4| echo $;
    5|
Unexpected ';', expecting variable (T_VARIABLE
-----------------------------------------------
Parse error: ./tests/examples/example-04/dir1/
    1| <?php
    2| $foo ='bar'
  > 3| echo $foo;
    4|
Unexpected 'echo' (T_ECHO)
Jakub-MacBook-Pro:PHP-Parallel-Lint jakubonder
```


Install

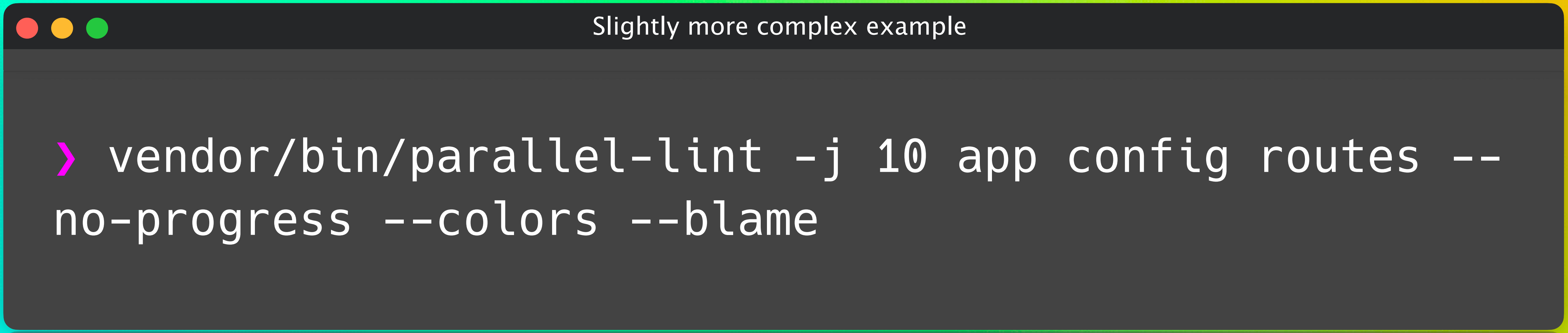
```
❯ composer require php-parallel-lint/php-parallel-
lint --dev
```

https://github.com/php-parallel-lint/PHP-Parallel-Lint


Simple, default example

```
❯ vendor/bin/parallel-lint --exclude .git --exclude
app --exclude vendor .
```

https://github.com/php-parallel-lint/PHP-Parallel-Lint



```
❯ vendor/bin/parallel-lint -j 10 app config routes --
no-progress --colors --blame
```

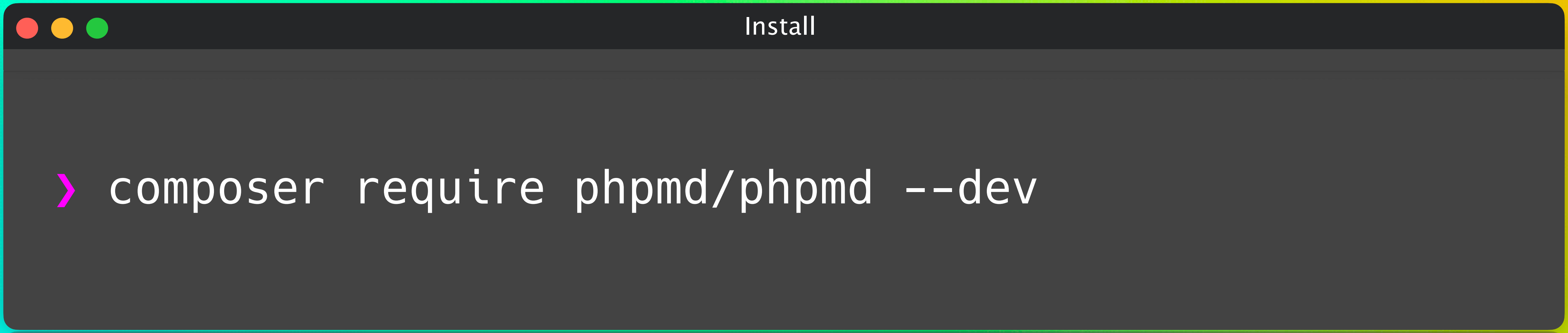
https://github.com/php-parallel-lint/PHP-Parallel-Lint

PHP Mess Detector

This application checks for code smells and best practices

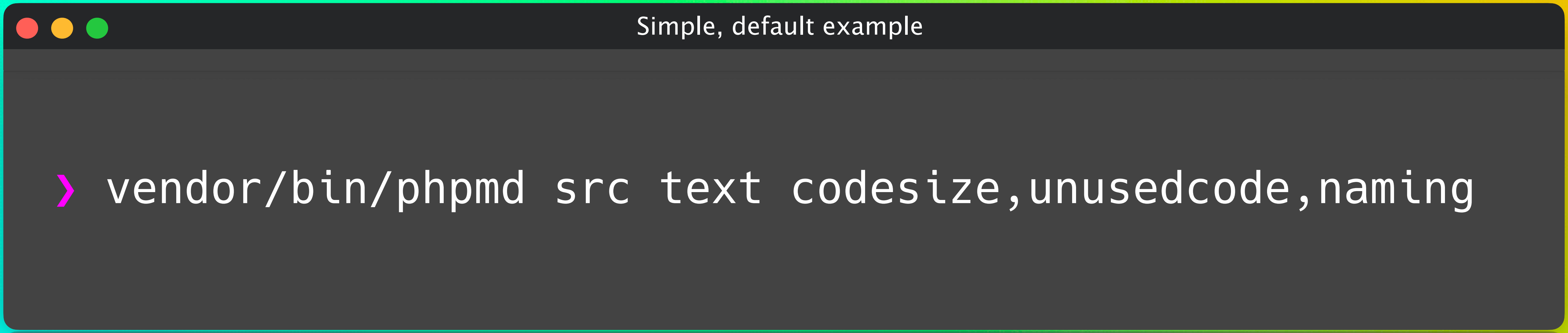
PHPMD looks for several potential problems:

- Possible bugs
- Suboptimal code
- Overcomplicated expressions
- Unused parameters, methods, properties



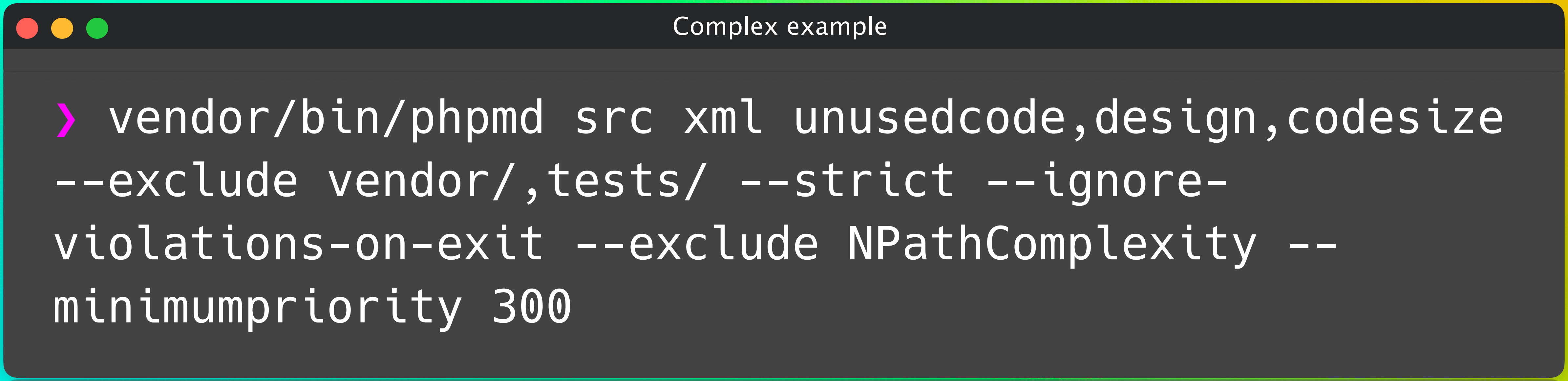
```
❯ composer require phpmd/phpmd --dev
```

https://phpmd.org/


Simple, default example

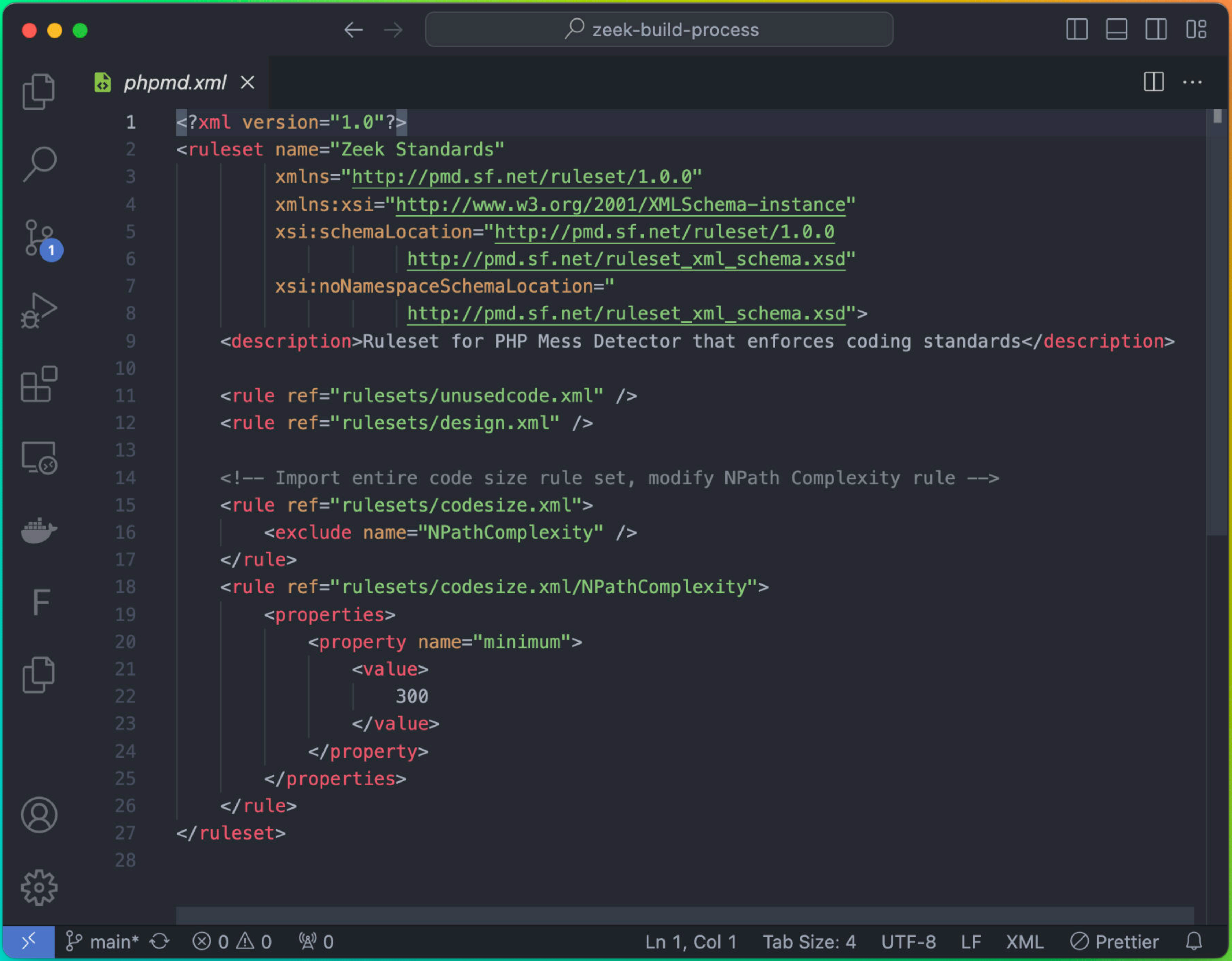
```
❯ vendor/bin/phpmd src text codesize,unusedcode,naming
```

https://phpmd.org/



```
❯ vendor/bin/phpmd src xml unusedcode,design,codesize
--exclude vendor/,tests/ --strict --ignore-
violations-on-exit --exclude NPathComplexity --
minimumpriority 300
```

https://phpmd.org/

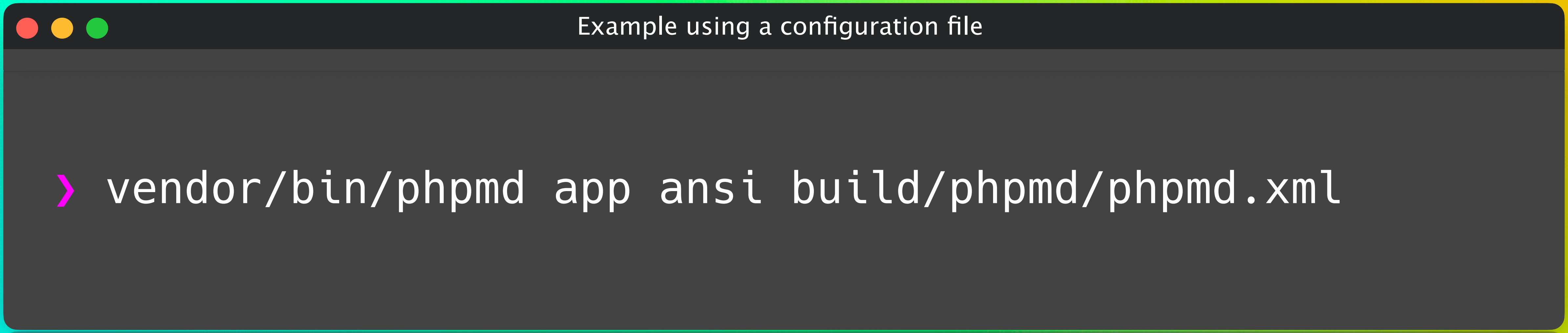


zeek-build-process

phpmd.xml

```xml
<?xml version="1.0"?>
<ruleset name="Zeek Standards"
         xmlns="http://pmd.sf.net/ruleset/1.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://pmd.sf.net/ruleset/1.0.0
                     http://pmd.sf.net/ruleset_xml_schema.xsd"
         xsi:noNamespaceSchemaLocation="
                     http://pmd.sf.net/ruleset_xml_schema.xsd">
    <description>Ruleset for PHP Mess Detector that enforces coding standards</description>

    <rule ref="rulesets/unusedcode.xml" />
    <rule ref="rulesets/design.xml" />

    <!-- Import entire code size rule set, modify NPath Complexity rule -->
    <rule ref="rulesets/codesize.xml">
        <exclude name="NPathComplexity" />
    </rule>
    <rule ref="rulesets/codesize.xml/NPathComplexity">
        <properties>
            <property name="minimum">
                <value>
                    300
                </value>
            </property>
        </properties>
    </rule>
</ruleset>
```

https://phpmd.org/



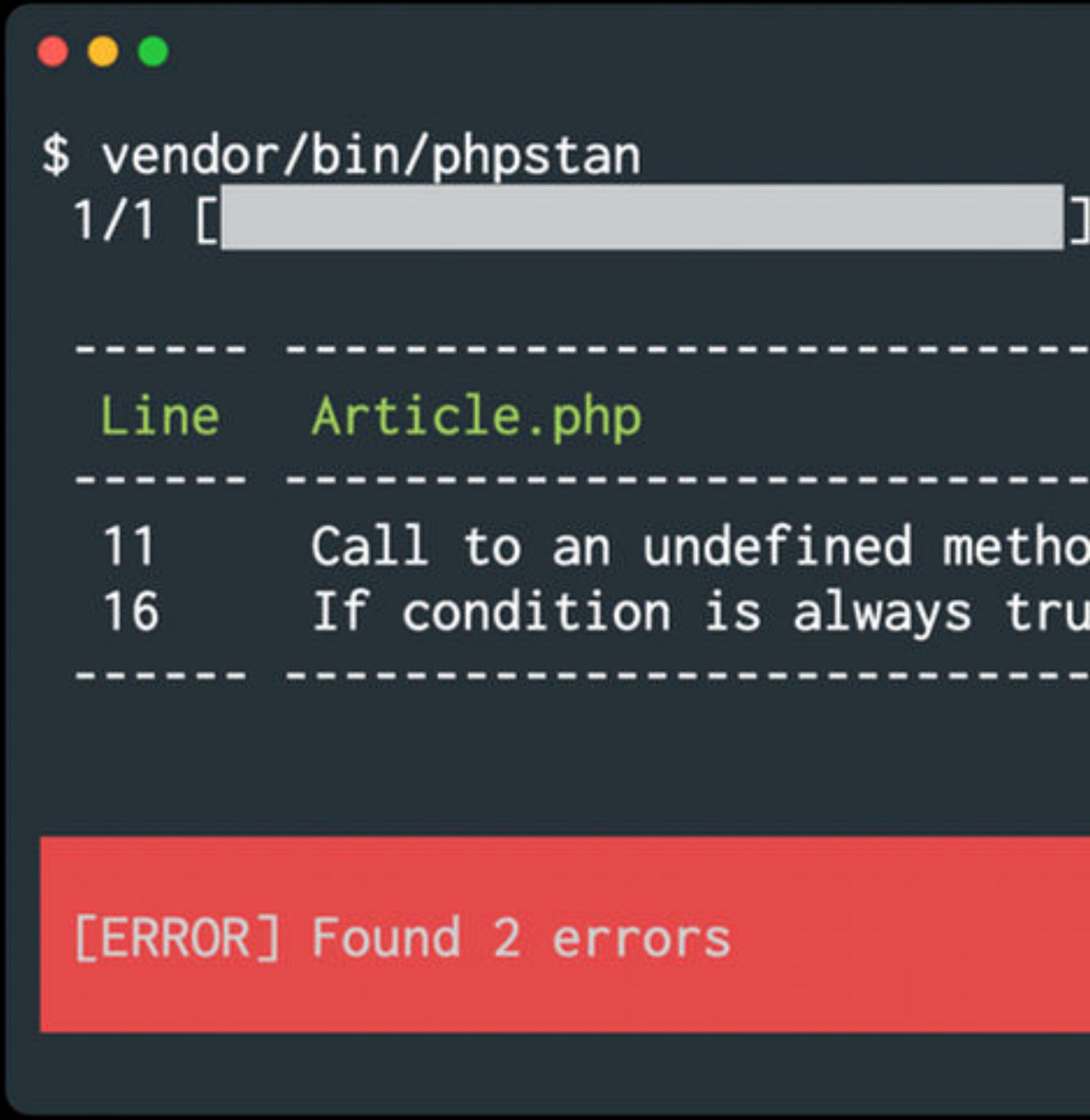
```
❯ vendor/bin/phpmd app ansi build/phpmd/phpmd.xml
```

https://phpmd.org/

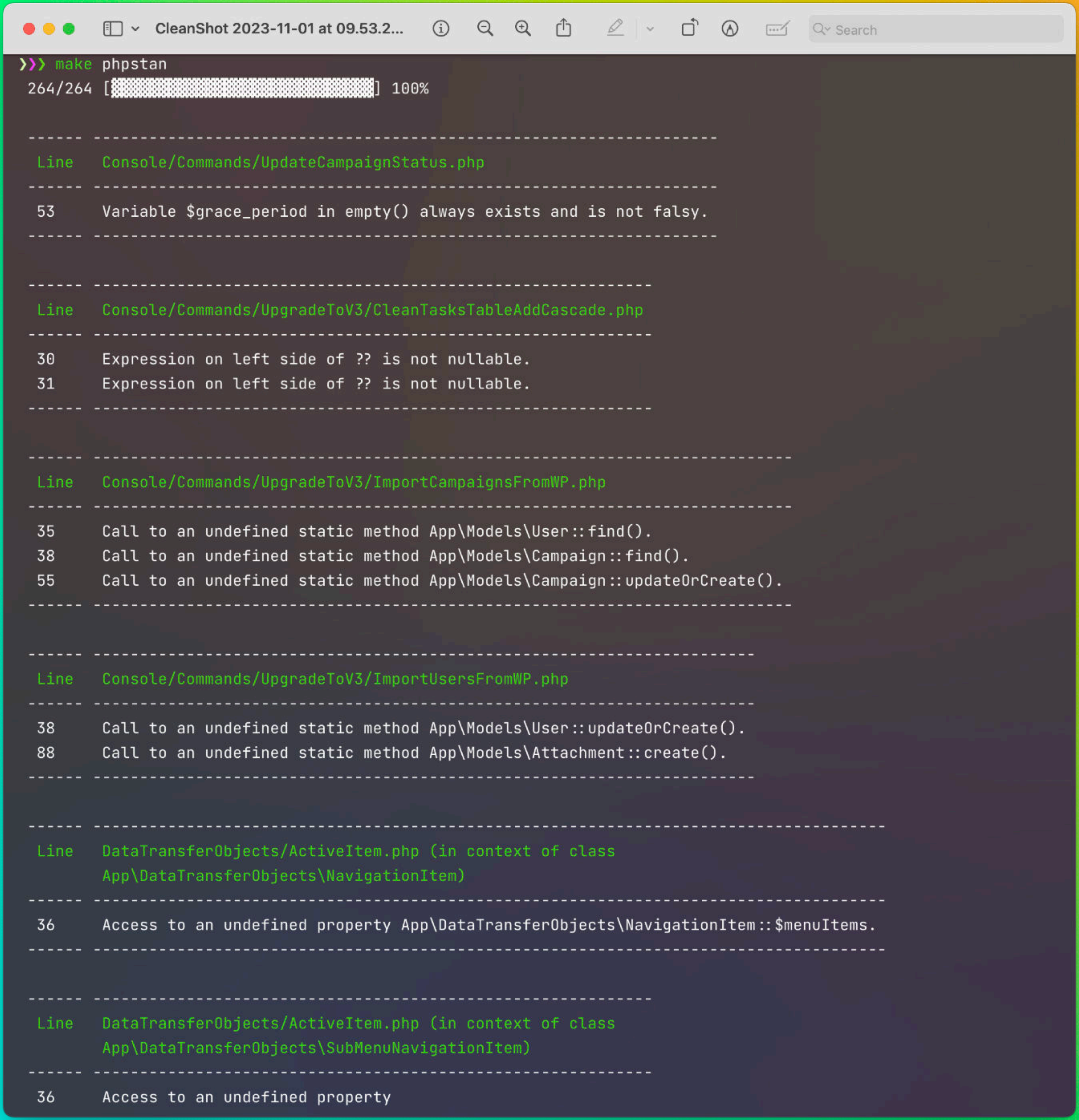
PHPStan

PHPStan finds bugs in your code without writing tests

PHPStan scans your whole codebase and looks for both obvious & tricky bugs. Even in those rarely executed if statements that certainly aren't covered by tests.

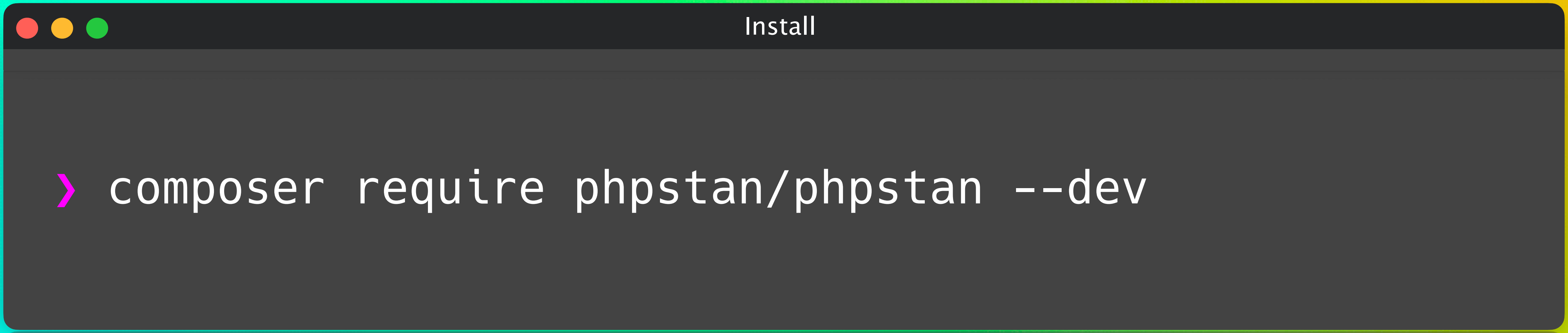


```
$ vendor/bin/phpstan
 1/1 [                              ]

 ------ ---------------------------
  Line   Article.php
 ------ ---------------------------
  11     Call to an undefined metho
  16     If condition is always tru
 ------ ---------------------------


 [ERROR] Found 2 errors
```



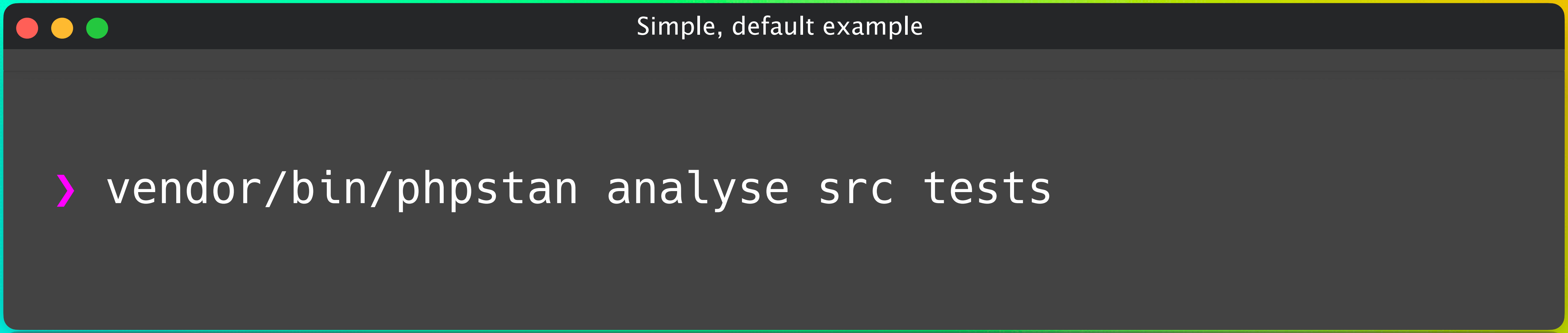
```
❯❯❯ make phpstan
 264/264 [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓] 100%

 ------ -------------------------------------------------------------------
  Line   Console/Commands/UpdateCampaignStatus.php
 ------ -------------------------------------------------------------------
  53     Variable $grace_period in empty() always exists and is not falsy.
 ------ -------------------------------------------------------------------


 ------ ------------------------------------------------------------
  Line   Console/Commands/UpgradeToV3/CleanTasksTableAddCascade.php
 ------ ------------------------------------------------------------
  30     Expression on left side of ?? is not nullable.
  31     Expression on left side of ?? is not nullable.
 ------ ------------------------------------------------------------


 ------ ---------------------------------------------------------------------------
  Line   Console/Commands/UpgradeToV3/ImportCampaignsFromWP.php
 ------ ---------------------------------------------------------------------------
  35     Call to an undefined static method App\Models\User::find().
  38     Call to an undefined static method App\Models\Campaign::find().
  55     Call to an undefined static method App\Models\Campaign::updateOrCreate().
 ------ ---------------------------------------------------------------------------


 ------ -----------------------------------------------------------------------
  Line   Console/Commands/UpgradeToV3/ImportUsersFromWP.php
 ------ -----------------------------------------------------------------------
  38     Call to an undefined static method App\Models\User::updateOrCreate().
  88     Call to an undefined static method App\Models\Attachment::create().
 ------ -----------------------------------------------------------------------


 ------ ------------------------------------------------------------------------------------
  Line   DataTransferObjects/ActiveItem.php (in context of class
         App\DataTransferObjects\NavigationItem)
 ------ ------------------------------------------------------------------------------------
  36     Access to an undefined property App\DataTransferObjects\NavigationItem::$menuItems.
 ------ ------------------------------------------------------------------------------------


 ------ ------------------------------------------------------------
  Line   DataTransferObjects/ActiveItem.php (in context of class
         App\DataTransferObjects\SubMenuNavigationItem)
 ------ ------------------------------------------------------------
  36     Access to an undefined property
```

https://phpstan.org/



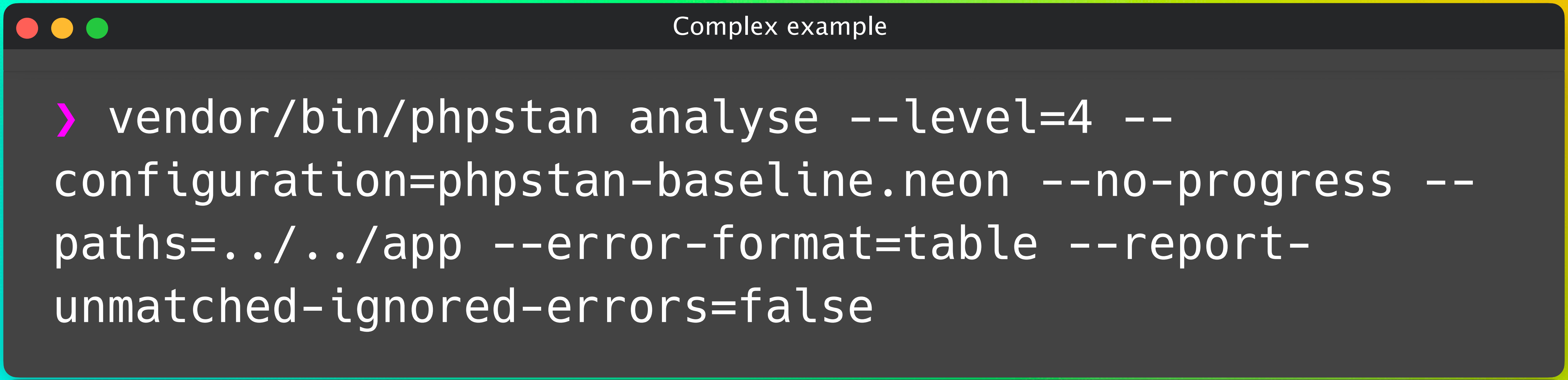
```
❯ composer require phpstan/phpstan --dev
```

https://phpstan.org/


Simple, default example

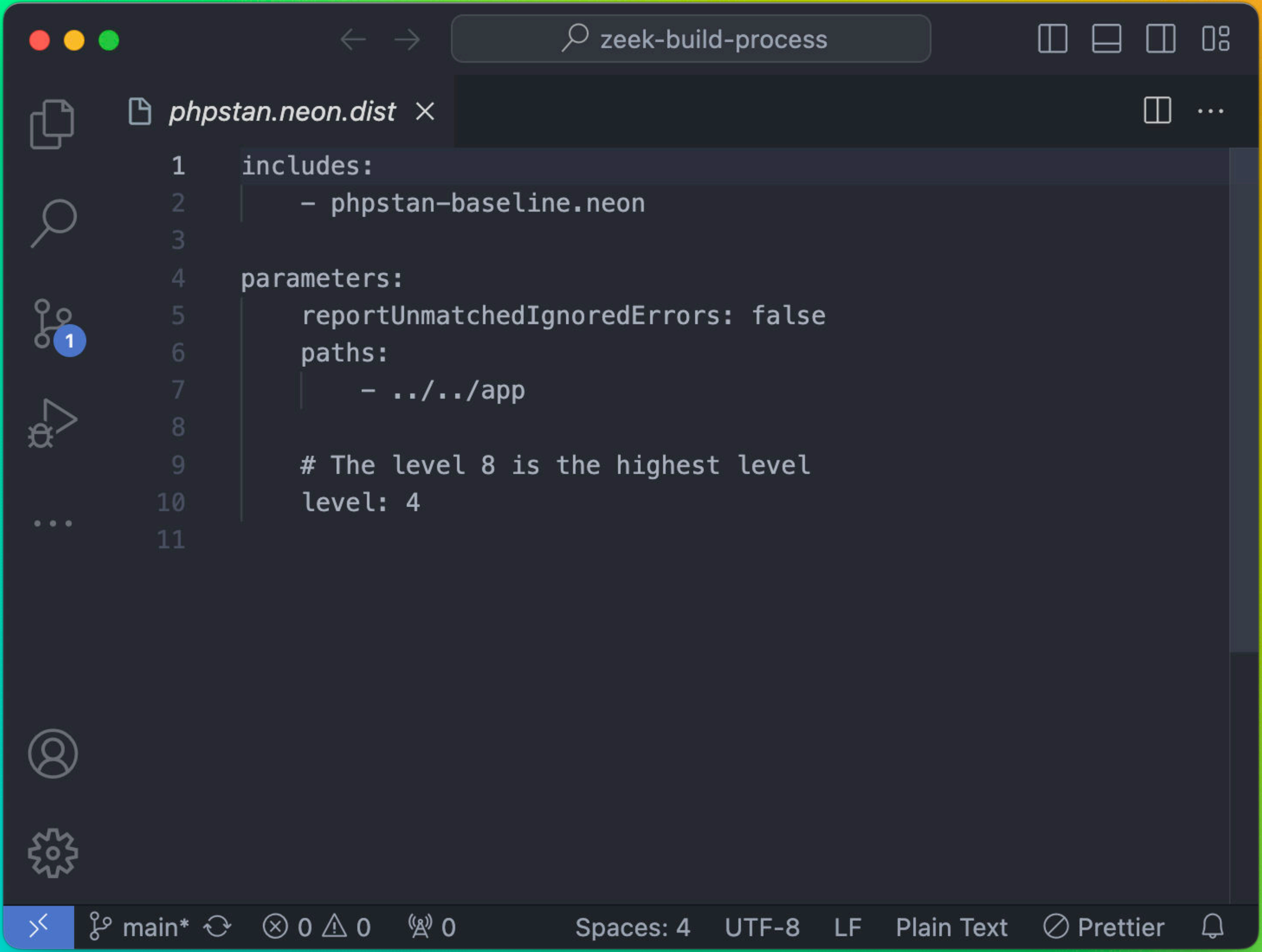
```
❯ vendor/bin/phpstan analyse src tests
```

https://phpstan.org/



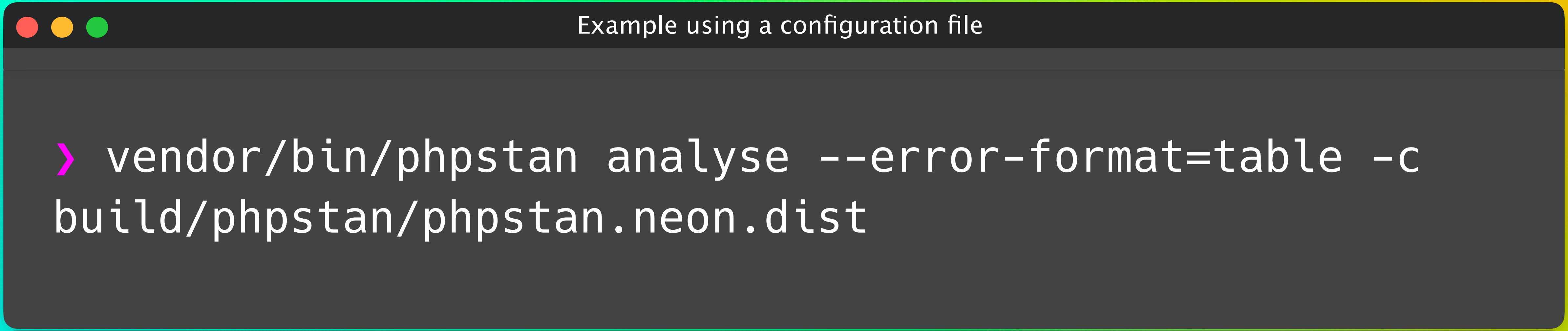
```
❯ vendor/bin/phpstan analyse --level=4 --configuration=phpstan-baseline.neon --no-progress --paths=../../app --error-format=table --report-unmatched-ignored-errors=false
```

https://phpstan.org/



```
includes:
    - phpstan-baseline.neon

parameters:
    reportUnmatchedIgnoredErrors: false
    paths:
        - ../../app

    # The level 8 is the highest level
    level: 4

```

https://phpstan.org/



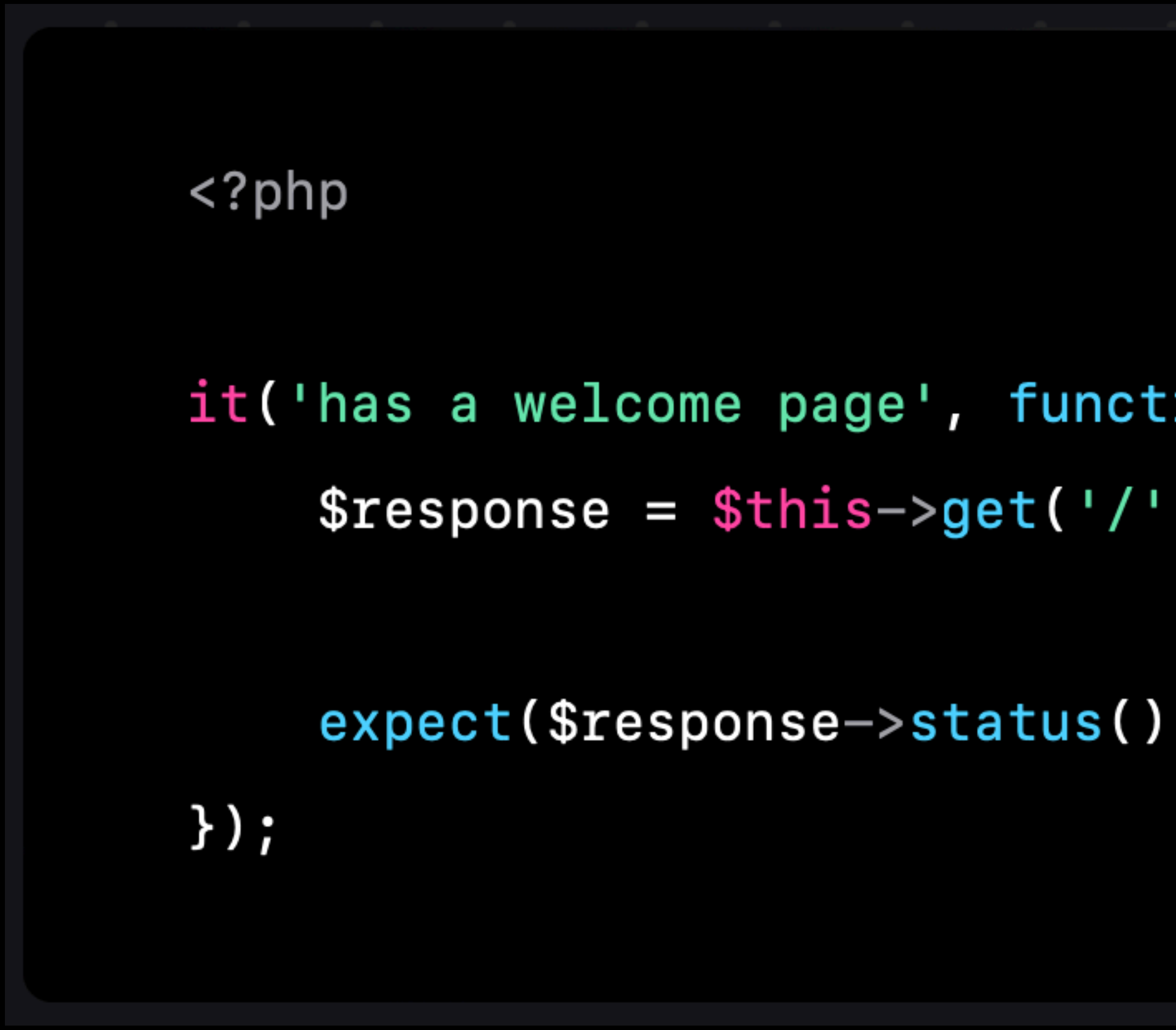
```
❯ vendor/bin/phpstan analyse --error-format=table -c
build/phpstan/phpstan.neon.dist
```

https://phpstan.org/

Pest / PHPUnit

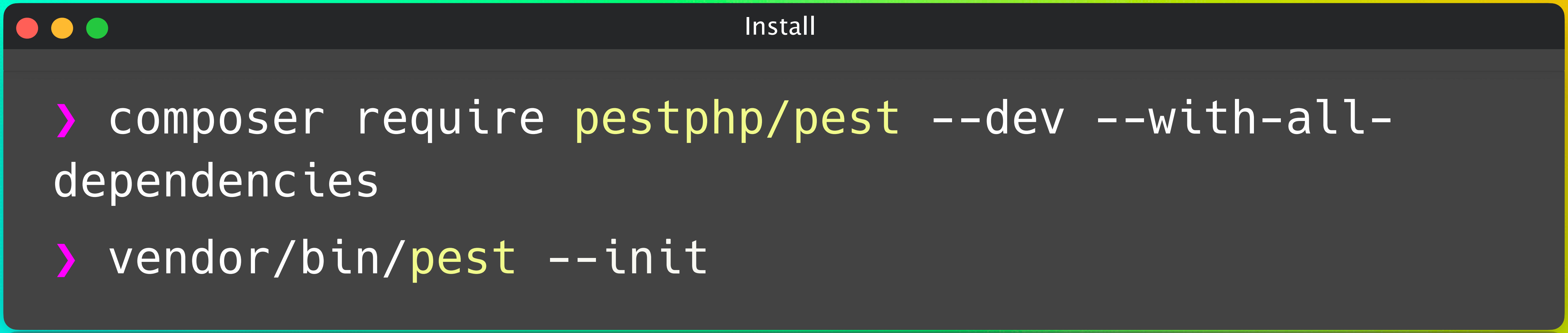
The elegant PHP testing framework

Pest is a testing framework with a focus on simplicity, meticulously designed to bring back the joy of testing in PHP.



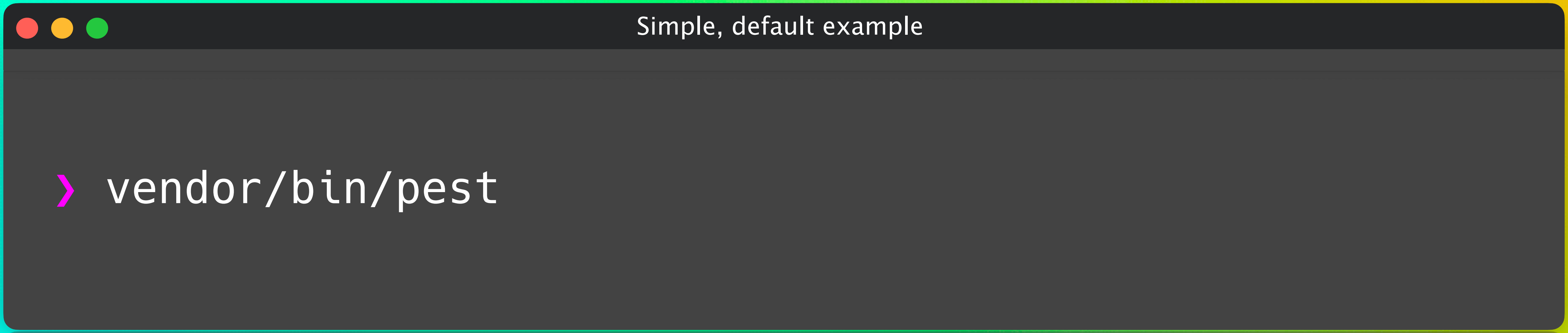
```
<?php

it('has a welcome page', funct
    $response = $this->get('/'

    expect($response->status()
});
```

https://pestphp.com/



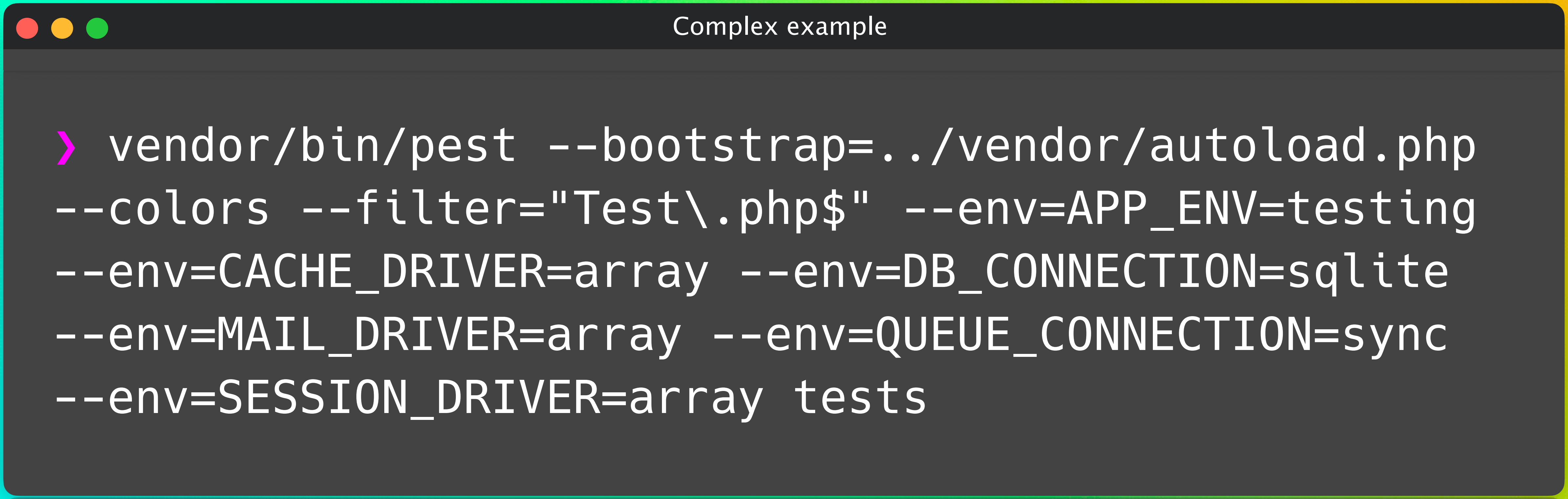
```
❯ composer require pestphp/pest --dev --with-all-dependencies
❯ vendor/bin/pest --init
```

https://pestphp.com/


Simple, default example

```
❯ vendor/bin/pest
```

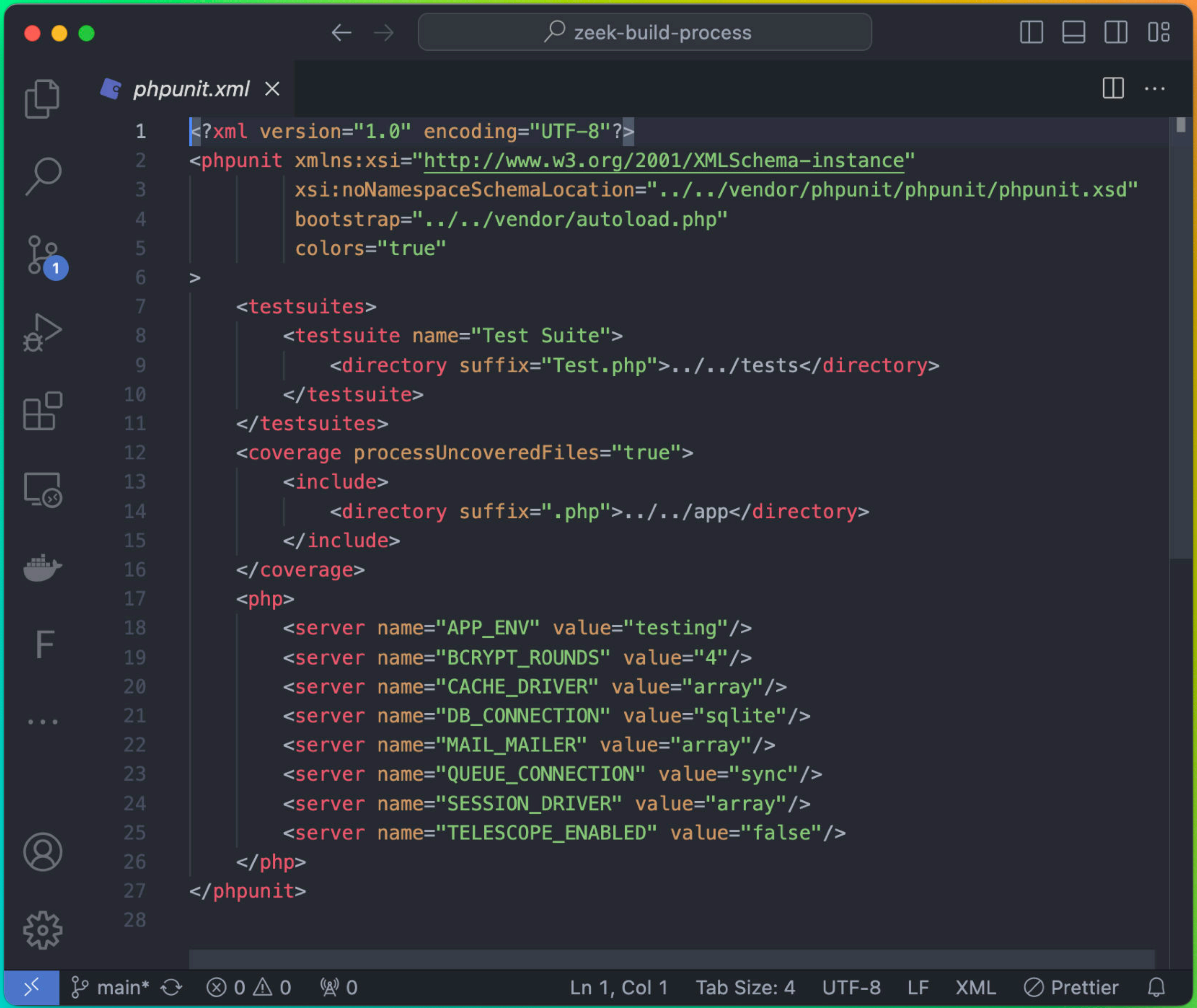
https://pestphp.com/



Complex example

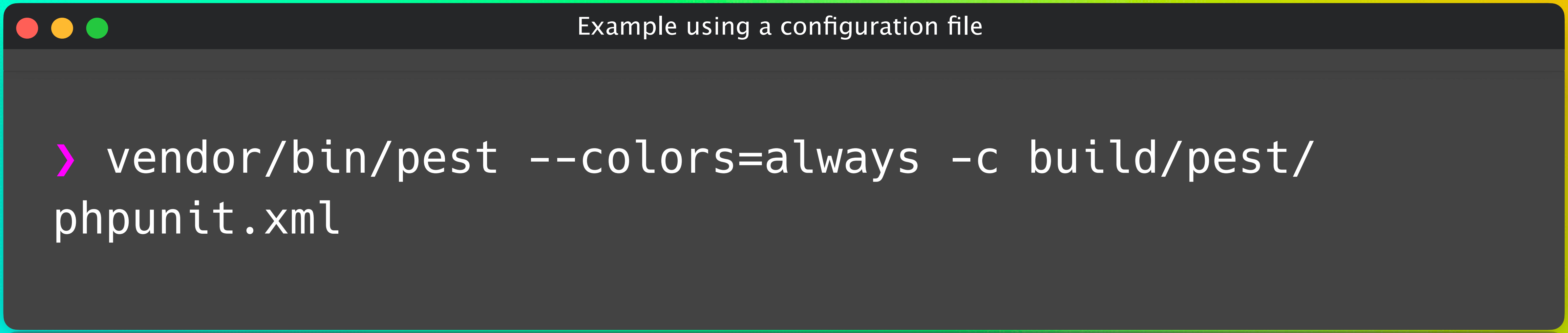
```
❯ vendor/bin/pest --bootstrap=../vendor/autoload.php
--colors --filter="Test\.php$" --env=APP_ENV=testing
--env=CACHE_DRIVER=array --env=DB_CONNECTION=sqlite
--env=MAIL_DRIVER=array --env=QUEUE_CONNECTION=sync
--env=SESSION_DRIVER=array tests
```

https://pestphp.com/



```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="../../vendor/phpunit/phpunit/phpunit.xsd"
         bootstrap="../../vendor/autoload.php"
         colors="true"
>
    <testsuites>
        <testsuite name="Test Suite">
            <directory suffix="Test.php">../../tests</directory>
        </testsuite>
    </testsuites>
    <coverage processUncoveredFiles="true">
        <include>
            <directory suffix=".php">../../app</directory>
        </include>
    </coverage>
    <php>
        <server name="APP_ENV" value="testing"/>
        <server name="BCRYPT_ROUNDS" value="4"/>
        <server name="CACHE_DRIVER" value="array"/>
        <server name="DB_CONNECTION" value="sqlite"/>
        <server name="MAIL_MAILER" value="array"/>
        <server name="QUEUE_CONNECTION" value="sync"/>
        <server name="SESSION_DRIVER" value="array"/>
        <server name="TELESCOPE_ENABLED" value="false"/>
    </php>
</phpunit>
```

https://pestphp.com/


Example using a configuration file

```
❯ vendor/bin/pest --colors=always -c build/pest/
phpunit.xml
```

https://pestphp.com/

Consistency Across Projects

Introducing Make

GNU Make

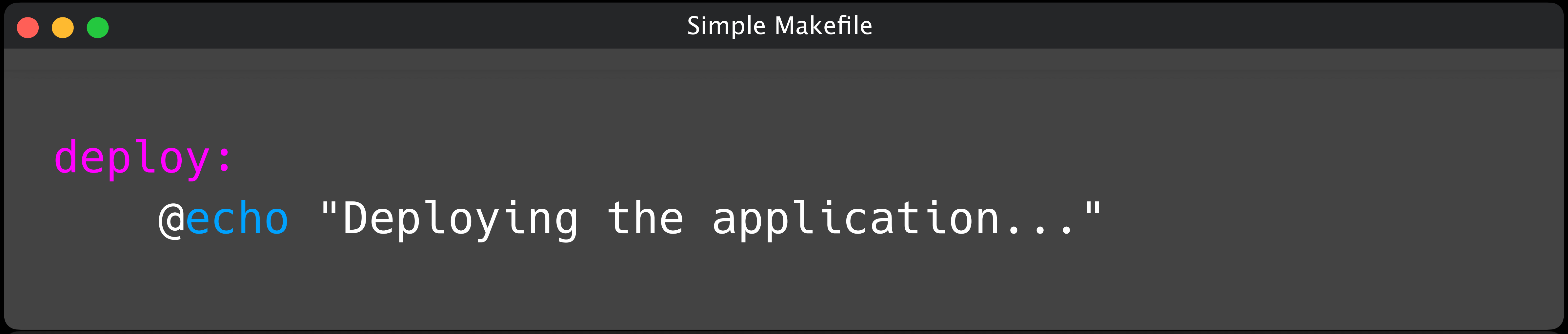
What is GNU Make?

- Automated Build Tool
- Reads `Makefile` for build rules
- Ideal for automating repetitive tasks

GNU Make

Inside a Makefile

- Rules with targets, prerequisites, and commands
- Variables and macros for flexibility
- Comments for clarity # This is a comment

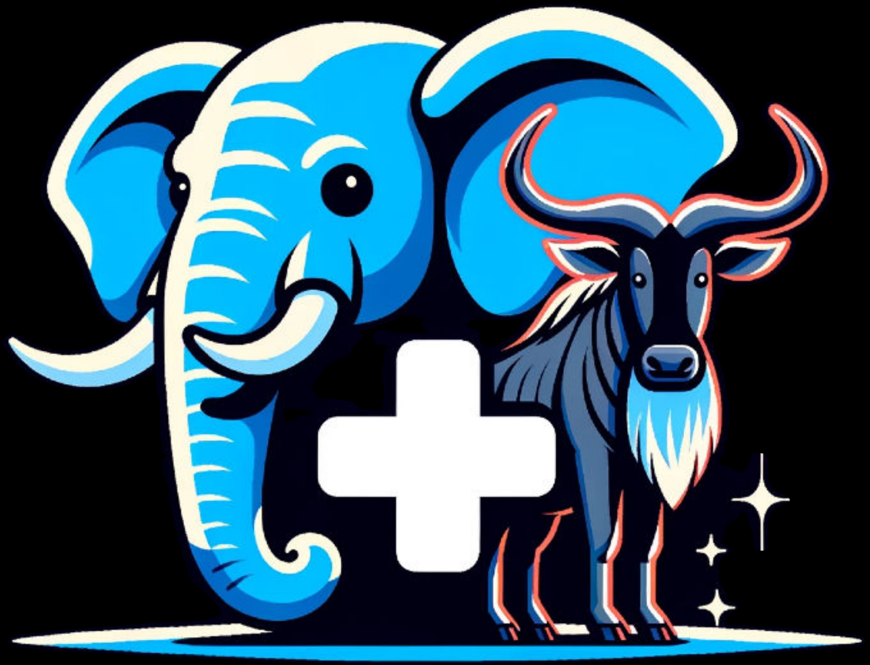


Simple Makefile

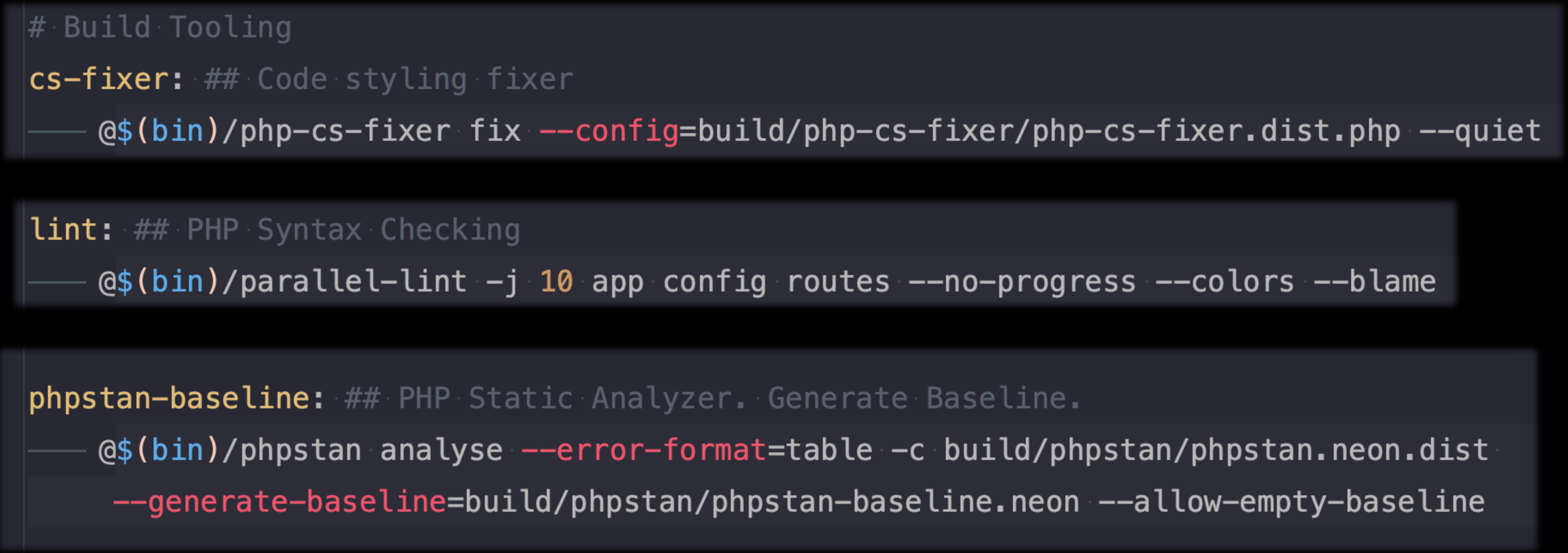
```
deploy:
	@echo "Deploying the application..."
```

GNU Make

Why Use Make for PHP?



- Simplify multiple command execution
- Combine PHP tools like phpstan, cs-fixer, and more
- Set up advanced flags per subcommand



```
# Build Tooling
cs-fixer: ## Code styling fixer
	@$(bin)/php-cs-fixer fix --config=build/php-cs-fixer/php-cs-fixer.dist.php --quiet
```

```
lint: ## PHP Syntax Checking
	@$(bin)/parallel-lint -j 10 app config routes --no-progress --colors --blame
```

```
phpstan-baseline: ## PHP Static Analyzer. Generate Baseline.
	@$(bin)/phpstan analyse --error-format=table -c build/phpstan/phpstan.neon.dist
	 --generate-baseline=build/phpstan/phpstan-baseline.neon --allow-empty-baseline
```

GNU Make

Building a Safety Net with Make

- Unified command for linting, testing, and analyzing
- Easy addition of new tools and flags
- Ensure consistent build and testing environment

Introducing Git Hooks

Git Hooks
(client side)

- pre-commit: Runs before a commit is created, useful for performing local checks.
- prepare-commit-msg: Runs before the commit message editor is opened but after default message is created. Useful for editing the default commit message.
- commit-msg: Runs after the commit message is entered but before the commit is made, generally to validate or modify the commit message.
- post-commit: Runs after the commit is made; often used for notifications or other post-commit actions.
- pre-rebase: Runs before a rebase is executed, often used to disallow rebasing of published commits.
- post-rewrite: Runs after a commit is amended or rebased; typically used for notification or to refresh status.
- pre-push: Runs before a `git push`, useful for doing server-side validation without making a round-trip.
- ... the list goes on ...

pre-commit

Runs before a commit is created, useful for performing local checks.

- **Common Uses**
  - Code Linting
  - Unit Testing
  - Code Formatting

pre-commit

Runs before a commit is created, useful for performing local checks.

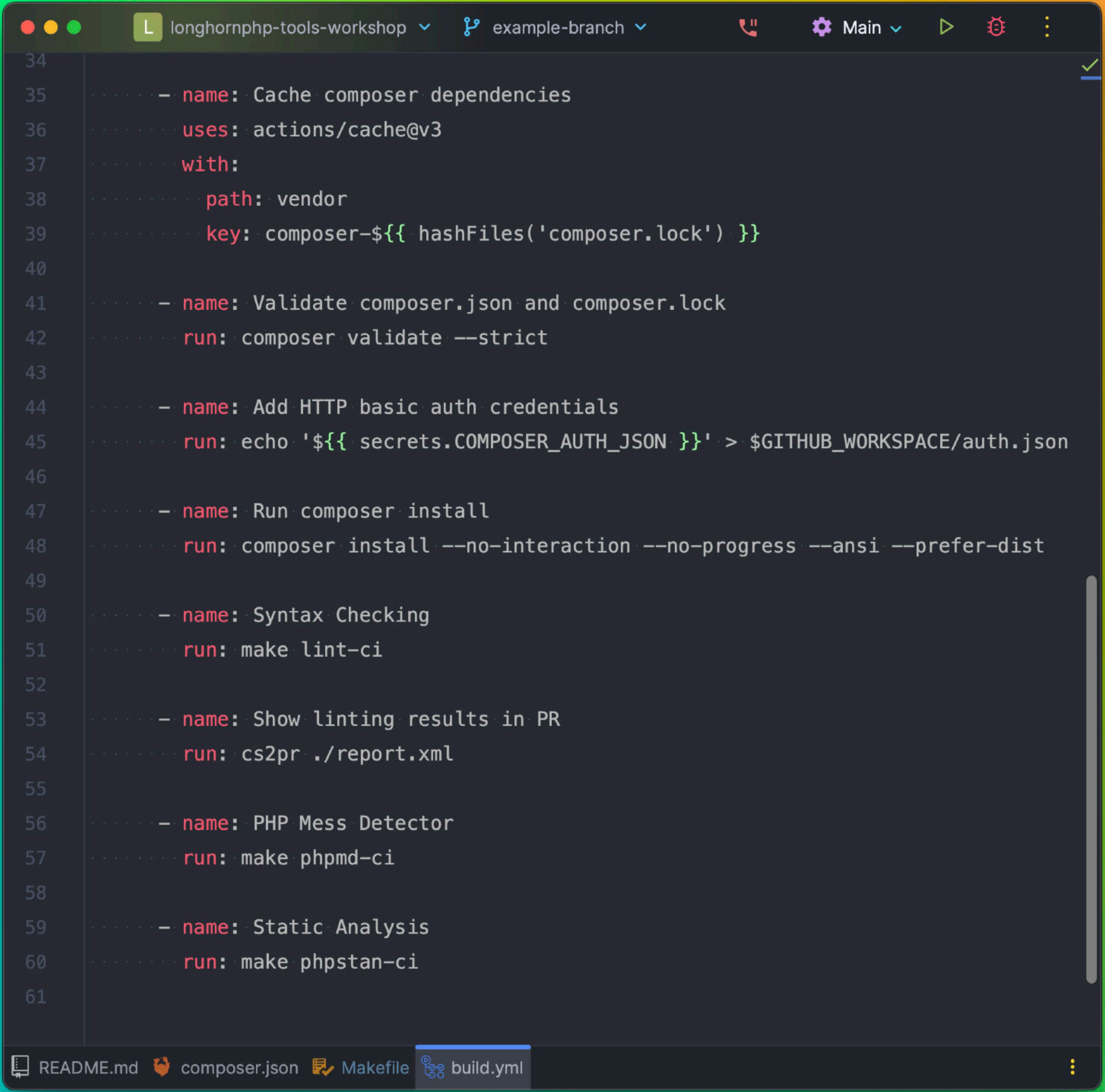
- **Benefits**
  - Ensures code quality
  - Prevents bad commits
  - Streamlines workflow

pre-commit

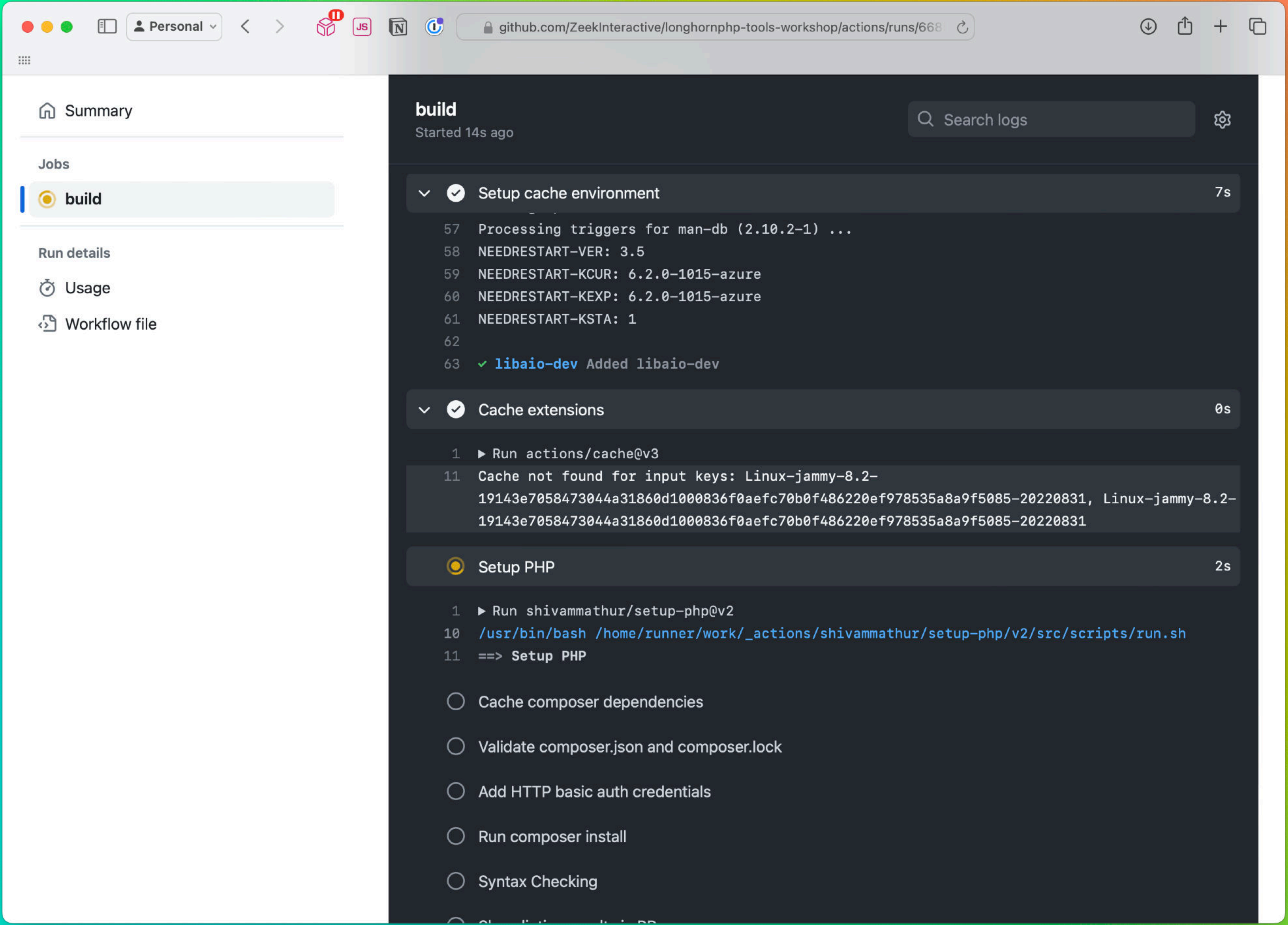
Runs before a commit is created, useful for performing local checks.

- **Setup**
  - Navigate to `.git/hooks`
  - Create & make `pre-commit` file executable
  - Add your script

Introducing GitHub Actions



```yaml
      - name: Cache composer dependencies
        uses: actions/cache@v3
        with:
          path: vendor
          key: composer-${{ hashFiles('composer.lock') }}

      - name: Validate composer.json and composer.lock
        run: composer validate --strict

      - name: Add HTTP basic auth credentials
        run: echo '${{ secrets.COMPOSER_AUTH_JSON }}' > $GITHUB_WORKSPACE/auth.json

      - name: Run composer install
        run: composer install --no-interaction --no-progress --ansi --prefer-dist

      - name: Syntax Checking
        run: make lint-ci

      - name: Show linting results in PR
        run: cs2pr ./report.xml

      - name: PHP Mess Detector
        run: make phpmd-ci

      - name: Static Analysis
        run: make phpstan-ci
```


github.com/ZeekInteractive/longhornphp-tools-workshop/actions/runs/668
Summary
Jobs
build
Run details
Usage
Workflow file
Search logs

build
Started 14s ago

Setup cache environment 7s

```
57 Processing triggers for man-db (2.10.2-1) ...
58 NEEDRESTART-VER: 3.5
59 NEEDRESTART-KCUR: 6.2.0-1015-azure
60 NEEDRESTART-KEXP: 6.2.0-1015-azure
61 NEEDRESTART-KSTA: 1
62
63 ✓ libaio-dev Added libaio-dev
```

Cache extensions 0s

```
1 ▶ Run actions/cache@v3
11 Cache not found for input keys: Linux-jammy-8.2-
   19143e7058473044a31860d1000836f0aefc70b0f486220ef978535a8a9f5085-20220831, Linux-jammy-8.2-
   19143e7058473044a31860d1000836f0aefc70b0f486220ef978535a8a9f5085-20220831
```

Setup PHP 2s

```
1 ▶ Run shivammathur/setup-php@v2
10 /usr/bin/bash /home/runner/work/_actions/shivammathur/setup-php/v2/src/scripts/run.sh
11 ==> Setup PHP
```

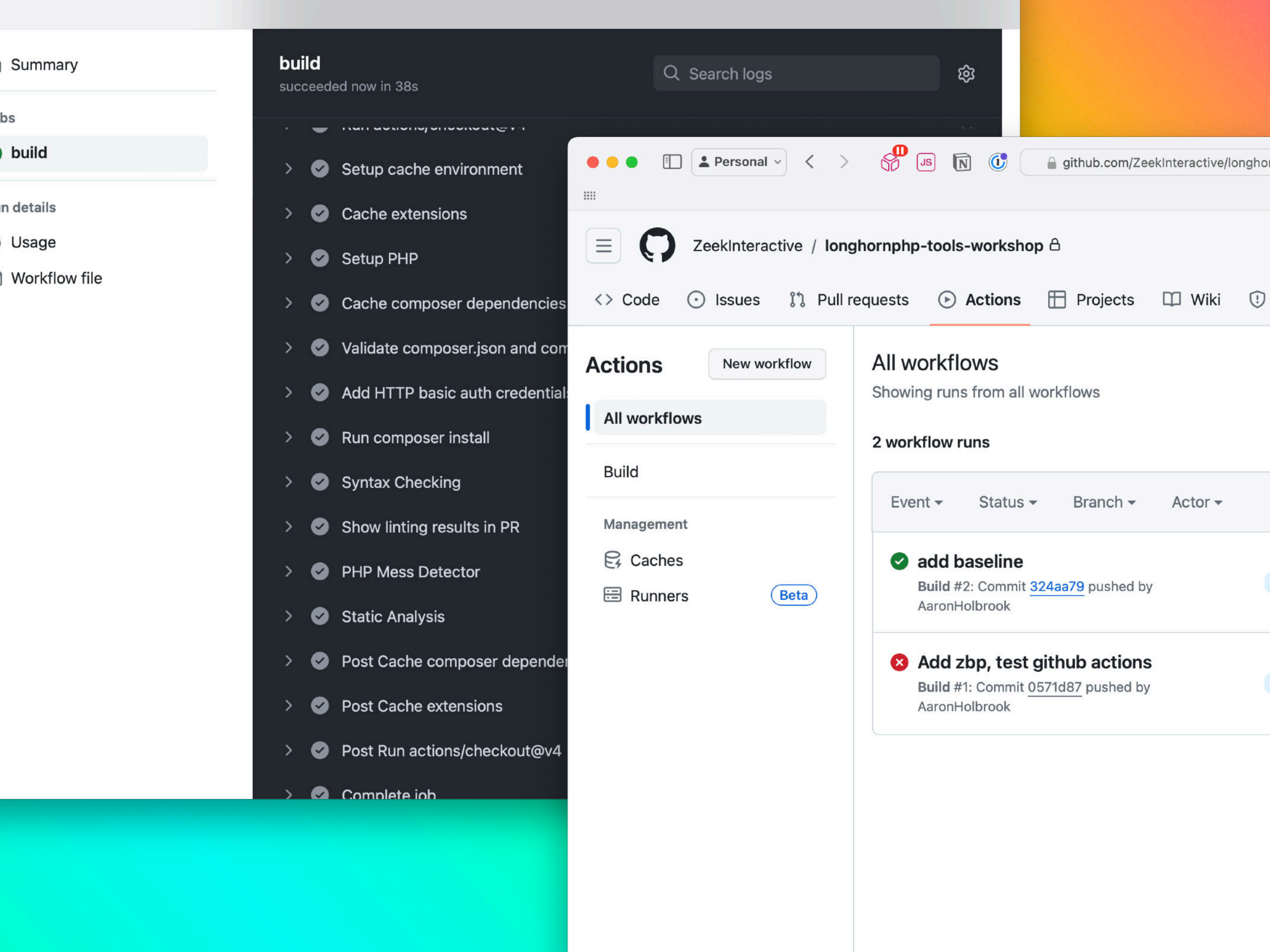
Cache composer dependencies

Validate composer.json and composer.lock

Add HTTP basic auth credentials

Run composer install

Syntax Checking



Summary

bs

build

n details

Usage

Workflow file

build

succeeded now in 38s

Search logs

```
Run actions/checkout@v4
Setup cache environment
Cache extensions
Setup PHP
Cache composer dependencies
Validate composer.json and com
Add HTTP basic auth credential
Run composer install
Syntax Checking
Show linting results in PR
PHP Mess Detector
Static Analysis
Post Cache composer depende
Post Cache extensions
Post Run actions/checkout@v4
Complete job
```

Personal

github.com/ZeekInteractive/longhor

ZeekInteractive / longhornphp-tools-workshop

Code Issues Pull requests Actions Projects Wiki

Actions

New workflow

All workflows

Build

Management

Caches

Runners Beta

All workflows

Showing runs from all workflows

2 workflow runs

Event Status Branch Actor

✓ add baseline

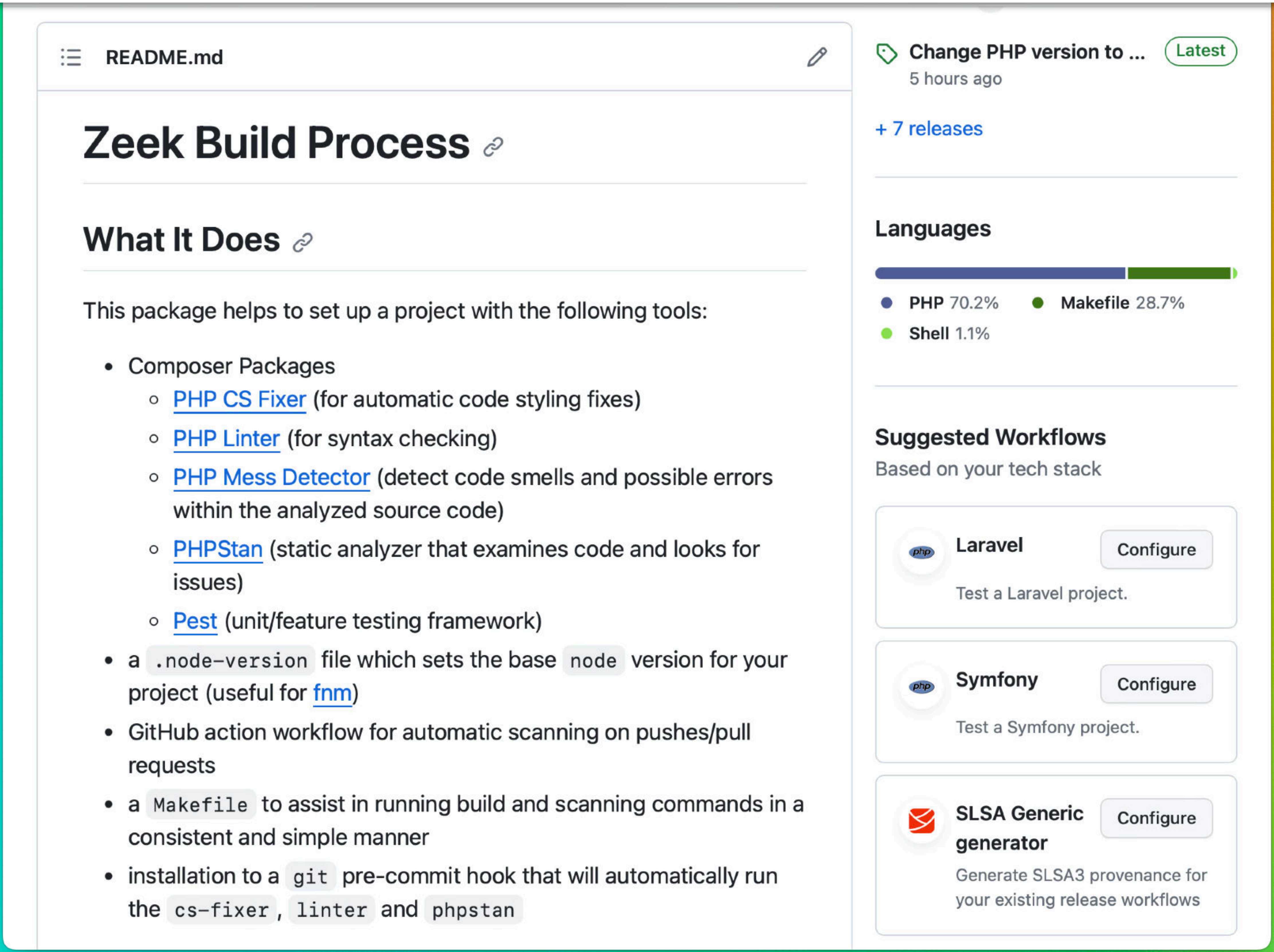
Build #2: Commit 324aa79 pushed by AaronHolbrook

✗ Add zbp, test github actions

Build #1: Commit 0571d87 pushed by AaronHolbrook

The Zeek Build Process

github.com/ZeekInteractive/zeek-build-process



README.md

Zeek Build Process

What It Does

This package helps to set up a project with the following tools:

- Composer Packages
  - PHP CS Fixer (for automatic code styling fixes)
  - PHP Linter (for syntax checking)
  - PHP Mess Detector (detect code smells and possible errors within the analyzed source code)
  - PHPStan (static analyzer that examines code and looks for issues)
  - Pest (unit/feature testing framework)
- a `.node-version` file which sets the base `node` version for your project (useful for fnm)
- GitHub action workflow for automatic scanning on pushes/pull requests
- a `Makefile` to assist in running build and scanning commands in a consistent and simple manner
- installation to a `git` pre-commit hook that will automatically run the `cs-fixer`, `linter` and `phpstan`

Change PHP version to ... Latest
5 hours ago

+ 7 releases

Languages

● PHP 70.2% ● Makefile 28.7%
● Shell 1.1%

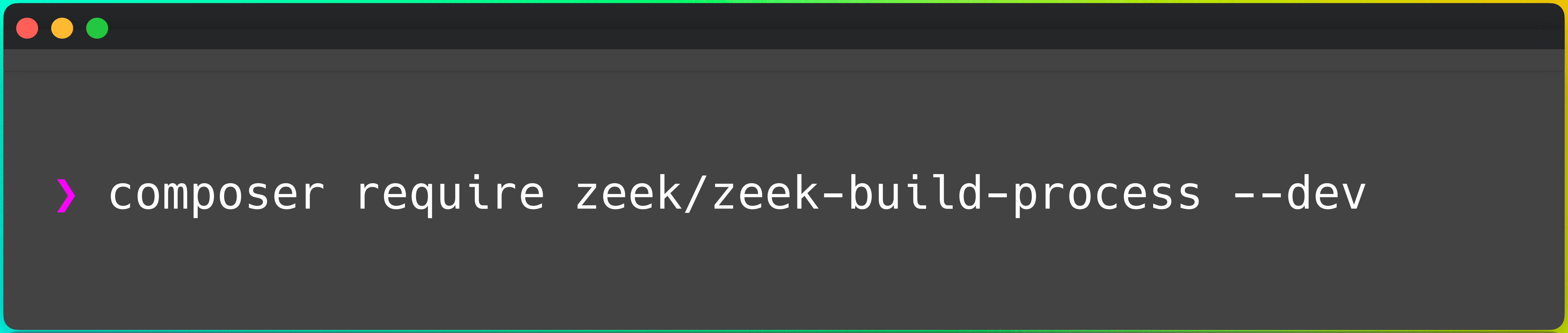
Suggested Workflows
Based on your tech stack

Laravel — Configure
Test a Laravel project.

Symfony — Configure
Test a Symfony project.

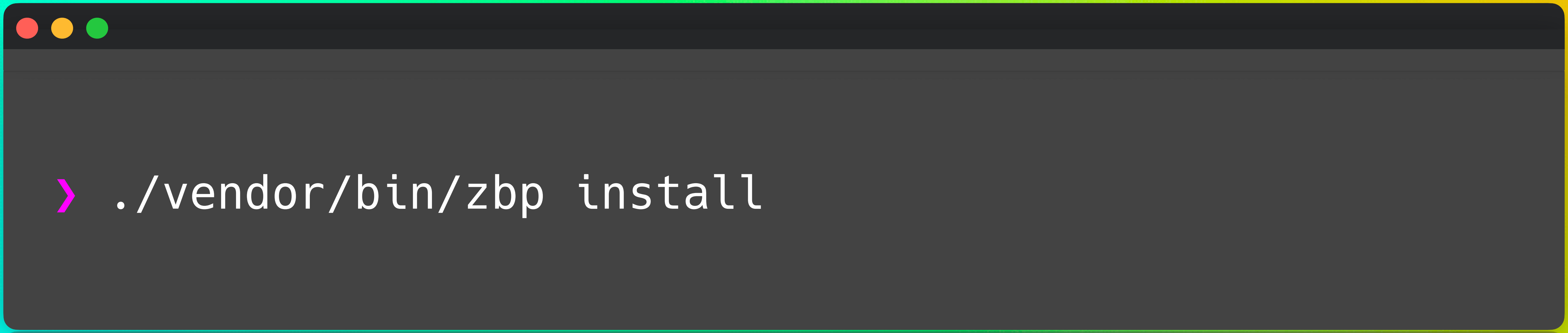
SLSA Generic generator — Configure
Generate SLSA3 provenance for your existing release workflows

https://github.com/ZeekInteractive/zeek-build-process



```
❯ composer require zeek/zeek-build-process --dev
```

https://github.com/ZeekInteractive/zeek-build-process



```
❯ ./vendor/bin/zbp install
```

https://github.com/ZeekInteractive/zeek-build-process

HANDS ON!

github.com/ZeekInteractive/longhornphp-tools-workshop

ZEEK

Join our Team!

- 🌐 Flexible Work Environment
- 💡 Innovative Projects
- 🌱 Growth and Development Opportunities
- ⚖️ Work-Life Balance
- 🏡 100% remote
- 📜 Seasoned company history with top talent
- 💰 Competitive Compensation
- 🛌 Flexible Fridays Program
- 🏝️ Flexible PTO
- 🩺 401k, Health, Dental, Vision Insurance
- 🎉 Fun as a Core Value: We believe life's too long to be so serious– enjoy the journey with us!

Inspired or curious?
Reach out and let's discuss further!

aaron@zeek.com

Scan To Explore Opportunities