

CI/CD

Automate and Optimize



PUG Torino

- [PHP User Group Torino](#) is a group of web developers interested in the PHP language (and not only)
- We are part of [GrUSP](#) association
- On [meetup.com](#) we are about 621 members
- We have also a [mailing list](#) with more than 100 members
- [Toolbox Coworking](#) is sponsoring the group



TOOLBOX
COWORKING





Daniele Barbaro

Tech Lead

Class of '84, workaholic, engineer, in an eternal struggle to conquer the world.

Since 20 whatever, PHP has been his weapon of choice for conquest.



<https://daniele.barbaro.online>



Why we are here:



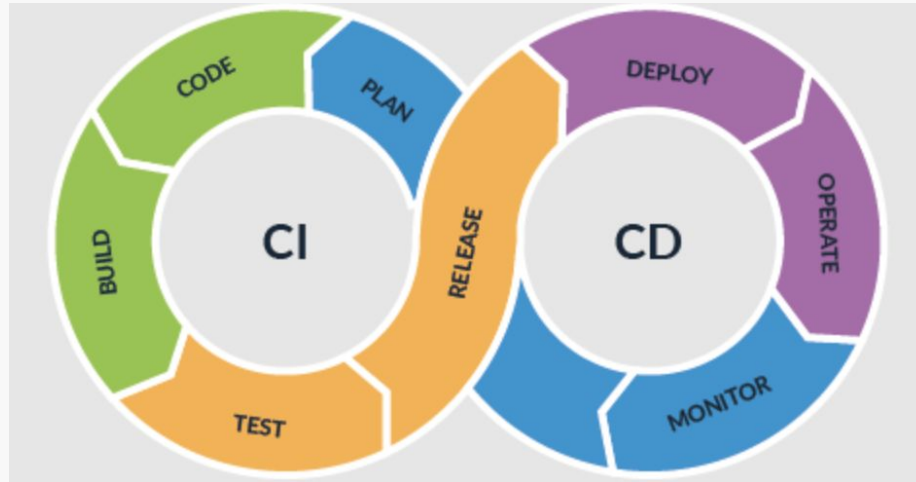


Why we are here:

- Understand what CI/CD means
- Learn the importance of using these tools
- Identify the right tools for each project
- Recognize the value of automating, optimizing processes, and testing
- Know how to approach a refactor



Continuous Integration (CI) Continuous Deployment/Delivery (CD)





Continuous Integration (CI)

Automated testing process and continuous integration of code changes

- Every code change is automatically tested.
- Changes are merged into the main branch if they pass the tests.





Continuous Deployment/Delivery (CD)

Automated deployment process of applications to production environments.

- Every change that passes continuous integration tests is automatically deployed.
- Eliminates the need for manual intervention for deployments.



Tell me why. Now.



Tell me why. Now. Why CI/CD?

- Dramatically reduce human impact on repetitive actions
- Improve code quality
- Quickly identify issues
- Release new features rapidly
- Enhance responsiveness to market demands





In a nutshell:

- **Consistency** - Always deployable
- **Efficiency** - No human repetitive actions
- **Quality** - Always tested



Workflow /



<https://github.com/marketplace>

Pipeline



<https://gitlab.com/pipeline-components>





Workflow

It is a series of jobs defined in a YAML configuration file that specifies what should happen when a certain event occurs in the repository.

- **Events:** These are the triggers that activate the workflow. They can be events such as a push or a pull request, etc.
(<https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows>)
- **Job:** A job is a set of steps that run on a virtual machine or a container. A job can include shell commands, scripts.
- **Steps:** These are the individual instructions within a job. Each step can execute a command or use an action.

```
name: CI

on: [push]

jobs:
  ci-test:
    runs-on: ubuntu-latest

    steps:
      - name: "Say Hello"
        run: echo "Hello
Daniele!"
```



Workflow

```
name: tests

on:
  push:
    branches:
      - master
      - develop
  pull_request:
    branches:
      - master

jobs:
  php-tests:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Install PHP
        uses: "shivammathur/setup-php@v2"

      - name: Run composer
        run: composer install --prefer-dist --no-progress --no-ansi --no-
interaction
      - name: Run unit tests
        run: vendor/bin/phpunit
```





Code Quality - Php Stan

PHPStan examines the entire codebase for obvious and elusive bugs, even in those rarely executed if statements that are certainly not covered by tests.

<https://phpstan.org/try>





Code Quality



```
- name: Run static analysis checks
  run: |
    vendor/bin/phpstan analyse src --error-format=checkstyle | cs2pr
```

*cs2pr: Annotate a Pull Request based on a Checkstyle XML

<https://github.com/staabm/annotate-pull-request-from-checkstyle>



Code Quality

27

28 + `private function ensureIsValidEmail($email): void`

✖ Check failure on line 28 in src/Email.php



GitHub Actions / php-tests

Method Danielebarbaro\CiCdSample\Email::ensureIsValidEmail() has parameter

29

{





Code Linting - Php cs

PHP CS Fixer analyzes the entire codebase for style and formatting issues, ensuring that your code conforms to established standards.

<https://cs.symfony.com/>





Code Style



```
- name: Check code style
  run: |
    vendor/bin/php-cs-fixer fix src --dry-run --format=checkstyle | cs2pr
```





Code Style Workflow

```
name: Check & fix styling

on: [push]

jobs:
  php-cs-fixer:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2
        with:
          ref: ${ github.head_ref }

      - name: Run PHP CS Fixer
        uses: docker://oskarstark/php-cs-fixer-ga
        with:
          args: --config=.php_cs.dist.php --allow-risky=yes

      - name: Commit changes
        uses: stefanzweifel/git-auto-commit-action@v4
        with:
          commit_message: Fix styling
```



Test Workflow



```
name: run-tests

on:
  push:
    branches:
      - master
      - develop
  pull_request:
    branches:
      - master

jobs:
  test:
    runs-on: ${ matrix.os }

    strategy:
      fail-fast: true
      matrix:
        os: [ubuntu-latest, windows-latest]
        php: [8.3, 8.2]
        laravel: [ '10.*', '11.*' ]
        stability: [prefer-lowest, prefer-stable]
        include:
          - laravel: 10.*
            testbench: 8.*
          - laravel: 11.*
            testbench: 9.*

    name: P${{ matrix.php }} - L${{ matrix.laravel }} - ${{ matrix.stability }} - ${{ matrix.os }}

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Setup PHP
        uses: shivammathur/setup-php@v2
        with:
          php-version: ${{ matrix.php }}
          extensions: dom, curl, libxml, mbstring, zip, pcntl, pdo, sqlite, pdo_sqlite, bcmath, soap, intl, gd, exif, iconv, imagick, fileinfo, xdebug
          coverage: xdebug

      - name: Setup problem matchers
        run: |
          echo "::add-matcher::${{ runner.tool_cache }}/php.json"
          echo "::add-matcher::${{ runner.tool_cache }}/phpunit.json"

      - name: Install Matrix dependencies
        run: |
          composer require "laravel/framework:${{ matrix.laravel }}" "orchestra/testbench:${{ matrix.testbench }}" --no-interaction --no-update
          composer update --${{ matrix.stability }} --prefer-dist --no-interaction

      - name: Execute tests
        run: vendor/bin/phpunit
```













Gitlab Comparison

```
phpunit:
  stage: test
  image: registry.gitlab.com/pipeline-components/phpunit:latest
  variables:
    OS: "ubuntu-latest"
    LARAVEL_VERSION: "11.*"
    STABILITY: "prefer-stable"
    TESTBENCH: "9.*"
  script:
    - curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
    - composer install
    - composer require "laravel/framework:$LARAVEL_VERSION" "orchestra/testbench:$TESTBENCH" --no-interaction
  --no-update
    - composer update --$STABILITY --prefer-dist --no-interaction
    - ./vendor/bin/phpunit --no-coverage

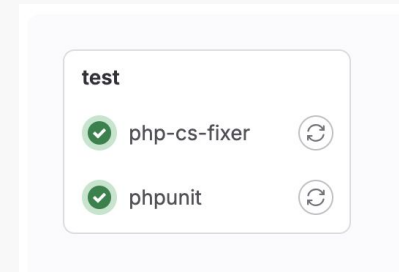
php-cs-fixer:
  stage: test
  image: registry.gitlab.com/pipeline-components/php-cs-fixer:latest
  script:
    - curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
    - composer install
    - ./vendor/bin/php-cs-fixer fix src --dry-run --diff --verbose --show-progress=dots
```



GitHub

<div> All checks have passed</div> <div>1 skipped and 33 successful checks</div>		Hide all checks
	dependabot-auto-merge / dependabot (pull_request_target) Skipped	Details
	 Check & fix styling / php-cs-fixer (push) Successful in 5s	Details
	 run-tests / P8.3 - L10.* - prefer-lowest - ubuntu-latest (pull_request) Successful in 19s	Details
	 run-tests / P8.3 - L10.* - prefer-lowest - ubuntu-latest (push) Successful in 20s	Details
	 run-tests / P8.3 - L10.* - prefer-stable - ubuntu-latest (pull_request) Successful in 16s	Details
	 run-tests / P8.3 - L10.* - prefer-stable - ubuntu-latest (push) Successful in 13s	Details

Gitlab





Demo Time





Refactor & Pipeline

[Rector](#) analyzes the entire codebase and automatically applies refactoring and updates, improving code quality and ensuring it is always up-to-date with best practices.

It helps with:

- PHP upgrades,
- framework updates,
- code quality improvement.





Workflow Ninja Mode

- Creation of frontend assets (e.g., npm build for Tailwind).
- Docker integration in CI.

Deployment automation:

- Tagging and releasing code.
- Using GitHub scripts for automated releases.
- Uploading build artifacts.



Workflow Ninja Mode

- expressions
- contexts
- events



<https://docs.github.com/en/actions/learn-github-actions/expressions>

<https://docs.github.com/en/actions/learn-github-actions/contexts>

<https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows>

```
name: Example Workflow

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Run step if condition is true
        if: github.event_name == 'push'
        run: echo "This step runs only on push events to the main
branch."

      - name: Always run this step
        run: echo "This step always runs."
```



Tools

App/Src:

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

Template:

- <https://github.com/VincentLanglet/Twig-CS-Fixer>
- <https://github.com/bdelespierre/laravel-blade-linter>

Analysis:

- <https://github.com/phpstan/phpstan>

Refactor:

- <https://github.com/rectorphp/rector>
- <https://github.com/driftingly/rector-laravel/>



Grazie ❤️



<https://daniele.barbaro.online>

Repo:

- <https://github.com/danielebarbaro/laravel-vat-eu-validator>
- <https://github.com/danielebarbaro/ci-cd-sample>

