This package provides automatic code style checking and formatting for Laravel applications and packages. Your code is formatted following Laravel's code style guide.

The package adds the php-cs-fixer tool and a community maintained ruleset to your application. The ruleset is a best effort attempt to match the code style the Laravel framework itself uses. Check out an example to see what the code style looks like.

You might want to use this package if you are writing a Laravel application, package or tutorial and you want to match the framework's code style.

Require this package with composer. It is recommended to only require the package for development.

```
composer require matt-allan/laravel-code-style --dev
```

The service provider will be automatically registered using package discovery.

If you don't use auto-discovery you should add the service provider to the providers array in ***config/app.php.***

```
php artisan vendor:publish --
provider="MattAllan\LaravelCodeStyle\ServiceProvider"
```

Once the package is installed you should publish the configuration.

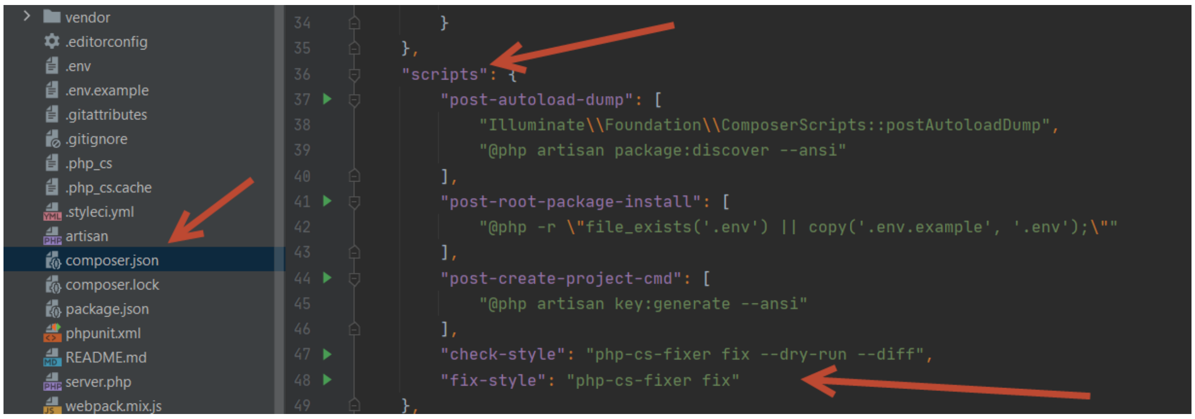
```
php artisan vendor:publish --
provider="MattAllan\LaravelCodeStyle\ServiceProvider"
```

To make checking and fixing code style easier for contributors to your project it's recommended to add the commands as a composer script.

The following example allows anyone to check the code style by calling composer check-style and to fix the code style with *composer fix-style.*

Note: go to **composer.json** and find scripts tags and add code below:

```
{
    // ...
    "scripts": {
        "check-style": "php-cs-fixer fix --dry-run --diff",
        "fix-style": "php-cs-fixer fix"
    }
}
```



```
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.php_cs
.php_cs.cache
.styleci.yml
artisan
composer.json
composer.lock
package.json
phpunit.xml
README.md
server.php
webpack.mix.js
```

```
        }
    },
    "scripts": {
        "post-autoload-dump": [
            "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
            "@php artisan package:discover --ansi"
        ],
        "post-root-package-install": [
            "@php -r \"file_exists('.env') || copy('.env.example', '.env');\""
        ],
        "post-create-project-cmd": [
            "@php artisan key:generate --ansi"
        ],
        "check-style": "php-cs-fixer fix --dry-run --diff",
        "fix-style": "php-cs-fixer fix"
    },
```

#Configuration

The default configuration is published as **.php_cs** in the project root. You can customize this file to change options such as the paths searched or the fixes applied.

Note: go to project root to fine **.php_cs** and open to configuration, the code below is automatically to public and root project when you artisan **vendor:publish**

```
<?php

require __DIR__.'/vendor/autoload.php';
require __DIR__.'/bootstrap/app.php';

return (new MattAllan\LaravelCodeStyle\Config())
    ->setFinder(
        PhpCsFixer\Finder::create()
            ->in(app_path())
            ->in(config_path())
            ->in(database_path('factories'))
            ->in(database_path('seeders'))
            ->in(resource_path('lang'))
            ->in(base_path('routes'))
            ->in(base_path('tests'))
    )
    ->setRules([
        '@Laravel' => true,
        '@Laravel:risky' => true
    ])->setRiskyAllowed(true);
```

Rules

By default only the @Laravel preset is enabled. This preset enforces the PSR-2 standard as well as nearly 100 other rules such as ordering use statements alphabetically and requiring trailing commas in multiline arrays.

A **@Laravel:risky** preset is also available. The **@Laravel:risky** preset enables rules that may change code behavior. To enable risky rules you need to add the preset and set isRiskyEnabled to true.

```
return (new MattAllan\LaravelCodeStyle\Config())
        ->setFinder(
            // ...
        )
        ->setRules([
            '@Laravel' => true,
            '@Laravel:risky' => true,
        ])
        ->setRiskyAllowed(true);
```

before you composer run-script to check style you need to config hook and the composer.json file like code bellow:

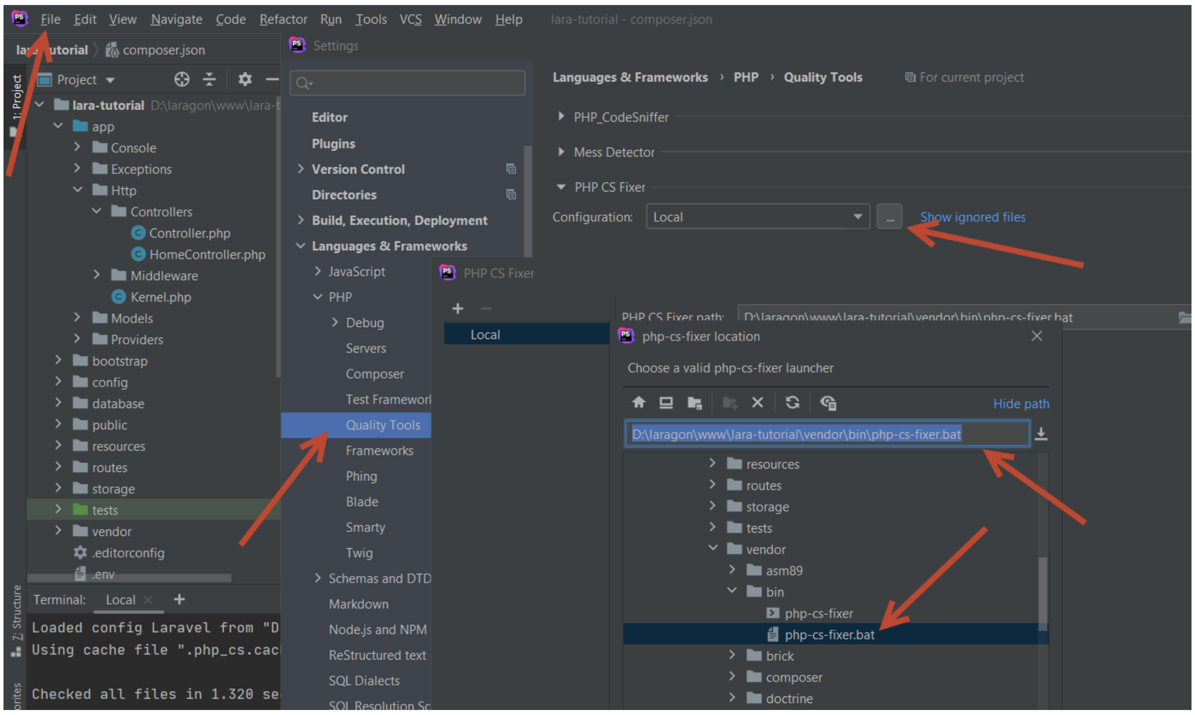
```
 "extra": {
       //
        "hooks": {
            "pre-commit": [
                "composer run-script check-style",
            ]
        }
    },
```

You can use command line to check and fix code style

```
composer run-script fix-style
composer run-script check-style
```

if you use PhpStorp you can configure on the setting like bellow:

file>settings->language & framework > Quality Tools> php cs fixer > local > go find path php-cs-fixer.bat in /vendor/ben/php-cs-fixer.bat in you project then click validate after that click ok to apply.



Configuration: Local

PHP CS Fixer path: D:\laragon\www\lara-tutorial\vendor\bin\php-cs-fixer.bat

Choose a valid php-cs-fixer launcher

D:\laragon\www\lara-tutorial\vendor\bin\php-cs-fixer.bat

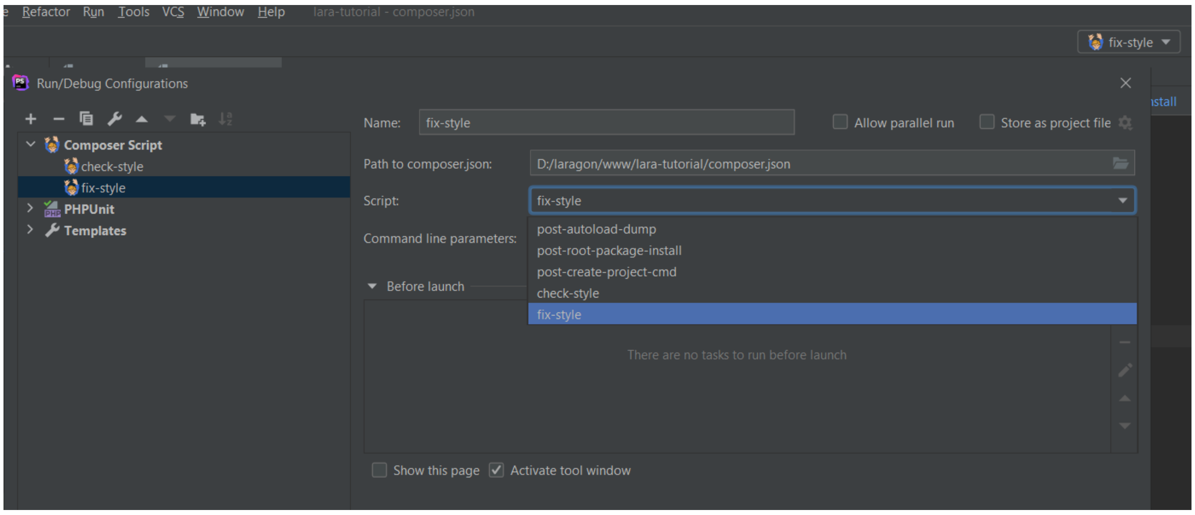
Terminal: Local

```
Loaded config Laravel from "D
Using cache file ".php_cs.cac

Checked all files in 1.320 se
```

if you want to run without command line just go to configure in phpstorp bellow:

edit configuration>composer script> create name and fine script name in script section;



Name: fix-style

Path to composer.json: D:/laragon/www/lara-tutorial/composer.json

Script: fix-style