# Contributing to API Platform

First of all, thank you for contributing, you're awesome!

To have your code integrated in the API Platform project, there are some rules to follow, but don't panic, it's easy!

## Reporting Bugs

If you happen to find a bug, we kindly request you to report it. However, before submitting it, please:

\* Check the [project documentation available online](https://api-platform.com/docs/)

Then, if it appears that it's a real bug, you may report it using GitHub by following these 3 points:

\* Check if the bug is not already reported!

\* A clear title to resume the issue

\* A description of the workflow needed to reproduce the bug

> \_NOTE:\_ Don't hesitate giving as much information as you can (OS, PHP version extensions...)

## Pull Requests

### Writing a Pull Request

First of all, you must decide on what branch your changes will be based depending of the nature of the change.

See [the dedicated documentation entry](https://api-platform.com/docs/extra/releases/).

To prepare your patch directly in the `vendor/` of an existing project (convenient to fix a bug):

1. Remove the existing copy of the library: `rm -Rf vendor/api-platform/core`

2. Reinstall the lib while keeping Git metadata: `composer install --prefer-source`

3. You can now work directly in `vendor/api-platform/core`, create a new branch: `git checkout -b my\_patch`

4. When your patch is ready, fork the project and add your Git remote: `git remote add <your-name> git@github.com:<your-name>/core.git`

5. You can now push your code and open your Pull Request: `git push <your-name> my\_patch`

Alternatively, you can also work with the test application we provide:

cd tests/Fixtures/app

./console assets:install --symlink

symfony serve

# or if you prefer using the PHP built-in web server

php -S localhost:8000 -t public/

### Matching Coding Standards

The API Platform project follows [Symfony coding standards](https://symfony.com/doc/current/contributing/code/standards.html).

But don't worry, you can fix CS issues automatically using the [PHP CS Fixer](https://cs.sensiolabs.org/) tool:

php-cs-fixer.phar fix

And then, add the fixed file to your commit before pushing.

Be sure to add only \*\*your modified files\*\*. If any other file is fixed by cs tools, just revert it before committing.

### Sending a Pull Request

When you send a PR, just make sure that:

\* You add valid test cases (Behat and PHPUnit).

\* Tests are green.

\* You make a PR on the related documentation in the [api-platform/docs](https://github.com/api-platform/docs) repository.

\* You make the PR on the same branch you based your changes on. If you see commits

that you did not make in your PR, you're doing it wrong.

\* Also don't forget to add a comment when you update a PR with a ping to [the maintainers](https://github.com/orgs/api-platform/people), so he/she will get a notification.

\* Squash your commits into one commit (see the next chapter).

### Tests

On `api-platform/core` there are two kinds of tests: unit (`phpunit`) and integration tests (`behat`).

Both `phpunit` and `behat` are development dependencies and should be available in the `vendor` directory.

#### PHPUnit and Coverage Generation

To launch unit tests:

vendor/bin/phpunit --stop-on-failure -vvv

If you want coverage, you will need the `pcov` PHP extension and run:

vendor/bin/phpunit --coverage-html coverage -vvv --stop-on-failure

Sometimes there might be an error with too many open files when generating coverage. To fix this, you can increase the `ulimit`, for example:

ulimit -n 4000

Coverage will be available in `coverage/index.html`.

#### Behat

The command to launch Behat tests is:

php -d memory\_limit=-1 ./vendor/bin/behat --suite=default --stop-on-failure --format=progress

If you want to launch Behat tests for MongoDB, the command is:

APP\_ENV=mongodb php -d memory\_limit=-1 ./vendor/bin/behat --suite=mongodb --stop-on-failure --format=progress

To get more details about an error, replace `--format=progress` by `-vvv`.

## Squash your Commits

If you have 3 commits, start with:

git rebase -i HEAD~3

An editor will be opened with your 3 commits, all prefixed by `pick`.

Replace all `pick` prefixes by `fixup` (or `f`) \*\*except the first commit\*\* of the list.

Save and quit the editor.

After that, all your commits will be squashed into the first one and the commit message will be the first one.

If you would like to rename your commit message, type:

git commit --amend

Now force push to update your PR:

git push --force-with-lease

# License and Copyright Attribution

When you open a Pull Request to the API Platform project, you agree to license your code under the [MIT license](LICENSE)

and to transfer the copyright on the submitted code to Kévin Dunglas.

Be sure to you have the right to do that (if you are a professional, ask your company)!

If you include code from another project, please mention it in the Pull Request description and credit the original author.

# Releases

This section is dedicated to maintainers.

1. Update the JavaScript dependencies by running `./update-js.sh` (always check if it works in a browser)

2. Update the `CHANGELOG.md` file (be sure to include Pull Request numbers when appropriate)

3. Create a signed tag: `git tag -s vX.Y.Z -m "Version X.Y.Z"`

4. Create a release using GitHub's UI and copy the changelog

5. Create a new release of `api-platform/api-platform`