# Contributing

Contributions are \*\*welcome\*\* and will be fully \*\*credited\*\*.

We accept contributions via Pull Requests on [Github](https://github.com/forkcms/forkcms).

## Guidelines

- \*\*[PSR-2 Coding Standard](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md)\*\* - The easiest way to apply the conventions is to install [PHP CS Fixer](https://github.com/FriendsOfPHP/PHP-CS-Fixer) and run it with `--level=psr2`.

- \*\*Document any change in behaviour\*\* - Make sure the `README.md`, `CHANGELOG.md`, `UPGRADE\_<upcoming\_version>` and any other relevant documentation are kept up-to-date.

- \*\*One pull request per feature\*\* - If you want to do more than one thing, send multiple pull requests.

- \*\*Send coherent history\*\* - Make sure each individual commit in your pull request is meaningful. If you had to make multiple intermediate commits while developing, please squash them before submitting.

- \*\*Add tests\*\* - We'd really like to get more tests, so PR's containing tests are most welcome.

## Running Tests

You must have Fork CMS installed for the functional tests to be able to run.

> You need a database with the same credentials as in your parameters.yml file, but the name suffixed with \_test to be able to run the (functional) tests

``` bash

# run all tests (so on an installed fork)

bin/phpunit

# only run unit tests (requires no database setup)

bin/phpunit --testsuite=unit

```

## How to submit pull requests

To be able to submit pull requests, you should have (a little) knowledge of git. The most useful commands will be included, but it's advised to read [Pro git](http://git-scm.com/book/en/v2). It's one of the most complete guides.

### Creating your own Fork

To start working on a pull request, you first need to create your own version of our repository, which is called a "fork" in GitHub. You can do this by pushing the "Fork" button on the right top of the repository page.

### Getting a local Fork instance

After creating your own \*fork\* of Fork CMS, you should fetch the code to your computer. You can do this using these commands:

```bash

# Go to the directory where you want to store Fork, for example ~/Sites/forkcms

cd ~/Sites/forkcms

# Clone your version of Fork in the current directory (mind the dot at the end)

# If you get permission denied errors, read https://help.github.com/articles/generating-ssh-keys/#platform-all

git clone git@github.com:<your-github-user-name>/forkcms.git .

# Install the dependencies of Fork CMS

# you will need composer: https://getcomposer.org/doc/00-intro.md#globally

composer install

```

That's it, you now have the newest version of the code of Fork CMS installed locally!

You'll possibly need to start a local (apache) server to be able to access Fork with your browser on a local address.

### Creating a branch

A best practice in git is creating a new branch for each new feature. We mostly name prefix our branchnames with feature- or bugfix- to show the type of PR, and then a name with dashes (- signs) instead of spaces.

You can do it using this command:

```bash

# Create and go to a branch in one step!

git checkout -b <name-of-your-branch>

```

This is in fact a shortcut for these two commands:

```bash

# Create a new branch

git branch <name-of-your-branch>

# Go to this branch

git checkout <name-of-your-branch>

```

### Work on your feature/bugfix

Now it's time to open up your editor and write the code for the contribution you wish to make!

Please read our contributing guidelines earlier in this document.

Make sure to commit your code at certain times with a meaningful message.

```bash

# Check what has changed in your branch

git status

# Use this until all files you wish to commit have been added

git add <path-to-your-file>

# Use this to add all changed files to your commit

git add .

# Commit your changes

git commit

```

The `git commit` command will pop up your editor and ask you to write a message. If you rather write your message in your command line, you can use `git commit -m "My meaningful commit message"`

### Pushing your code to your Forked repository

All your commits are stored locally on your computer. To make sure you they are available on github, you have to push your code.

```bash

# Push your code to the remote

git push origin <your-branch-name>

```

### Submitting your pull request

We advise you to run the tests locally before creating a pull request.

Creating a pull request is really easy on GitHub. Just go to [our repository on GitHub](https://github.com/forkcms/forkcms) after pushing your code and a yellow bar will appear with the question if you want to create a new pull request.

If this yellow bar isn't available, you can also go to "pull request", click on "New pull request", and select your branch to be used.

Write some explanation of what has changed and why in the description. If it's related to an issue, it's nice to include the issue number.

### Waiting for approval

After creating a pull request, we'll go trough your code, to make sure everything works as stated, and our quality norms are met. If necessary, we'll give you feedback on how you can improve your pull request.

After handling feedback, we'll thank you for your help, merge the changes to our master branch, and your change will be included in the next release!

If it's a security fix, we'll even create a new release right away.

### Squashing commits

Sometimes, you can be asked to squash your commits. This is mostly the case when your pull request contains a lot of commits for smaller changes.

Squashing commits will change your commit history to contain less commits (or do a lot of other funky stuff). It can be done this way:

```bash

# squash the last 5 commits

git rebase -i HEAD~5

```

If your pull request contained another amount of commits, you can change the number in this command.

An editor will open up containing the hashes and the commit messages of your last 5 commits. It should look like this:

```

pick c171aae Write tests for the new ModulesSettings class.

pick f6cf11c Rename the modules\_settings to fork.settings.

pick 66a7f2b Refactor using rename method.

pick 123491d Deprecate the module settings methods in the BackendModel

pick cd64652 Deprecate the modulesettings methods in the frontend model.

```

If you want to put all changes in one commit, you can just leave "pick" on the first line, and change the word "pick" on the other lines to "squash".

This way, git will know it has to merge all changes of the commits into the first one.

```

pick c171aae Write tests for the new ModulesSettings class.

squash f6cf11c Rename the modules\_settings to fork.settings.

squash 66a7f2b Refactor using rename method.

squash 123491d Deprecate the module settings methods in the BackendModel

squash cd64652 Deprecate the modulesettings methods in the frontend model.

```

After saving and closing your editor, git will start rebasing it. A new editor will pop up, to enable you to change the commit message.

After saving your new commit message, you'll need to force push your changes to your branch.

```bash

# overwrite the git history on the remote too

git push -f origin <your-branch-name>

```

Et voila, your pull request has been changed!

If you want to learn more about the possibilities of squashing, you can find a lot of info here: <http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html>.

## Support

If you need some more help, or if you want to discuss about a feature, you can do this in multiple ways.

\* Create a GitHub issue <https://github.com/forkcms/forkcms/issues>

\* Join us on Slack: <https://fork-cms.herokuapp.com>

\* Create a WIP pull request. We can discuss how we can work further on it.