# Contributing to MikroORM

As an open source project, the best way to support MikroORM is to contribute to its development.

You can start with reporting new issues, improving the docs, or - if you feel confident enough

to dive into the source code - send a pull request!

\*\*Any help is much appreciated!\*\*

## Got a Question or Problem?

First take a look at documentation, then try to go thru existing issues. If you have a general

question about how something works, better place to ask is our

[Slack channel](https://join.slack.com/t/mikroorm/shared\_invite/enQtNTM1ODYzMzM4MDk3LTBmZDNlODBhYjcxNGZlMTkyYzJmODAwMDhjODc0ZTM2MzQ2Y2VkOGM0ODYzYTJjMDRiZDdjMmIxYjI2OTY0Y2U)

or [Stack overflow](https://stackoverflow.com/tags/mikro-orm/).

## Found a bug?

If you find a bug in the source code, you can help by [submitting an issue](https://github.com/mikro-orm/mikro-orm/issues/new)

or even better, by [submitting a Pull Request](https://github.com/mikro-orm/mikro-orm/pulls) with a fix.

## Missing a feature?

You can \*request\* a new feature by [submitting an issue](https://github.com/mikro-orm/mikro-orm/issues/new) to this GitHub

Repository. If you would like to \*implement\* a new feature, please submit an issue with a proposal for your work first,

so we can discuss what is the best way to implement, as well as to be sure nobody else works on that already.

## Submission guidelines

### Submitting an issue

Before you submit an issue, please search the issue tracker, maybe an issue for your problem already exists and the discussion

might inform you of workarounds readily available.

Please provide steps to reproduce for found bug or ideally fork the repository and add failing test that demonstrates what

is wrong. This will help to understand and fix the issue faster.

### Submitting a pull request

Before you submit your pull request consider the following guidelines:

- Search [GitHub](https://github.com/mikro-orm/mikro-orm/pulls) for an open or closed PR

that relates to your submission. You don't want to duplicate effort.

- Fork the project, install NPM dependencies and start docker to have all databases ready.

[run-rs](https://www.npmjs.com/package/run-rs) is used to manage mongodb replica set.

On windows you might need to execute `run-rs` directly as well as adjust some configuration,

refer to their docs for more information.

```sh

yarn

docker-compose up -d

yarn run-rs

```

- Run tests before you start working, to be sure they all pass and your setup is working correctly:

```sh

yarn test

```

- This project aims to have 100% code coverage, so be sure to \*\*include appropriate test cases\*\*.

- Follow defined [coding standard](#coding-standard), use `yarn lint` command to check it.

- Commit your changes using a descriptive commit message that follows defined [commit message conventions](#commit).

Adherence to these conventions is necessary because release notes are automatically generated from these messages.

- Push the code to your forked repository and create a pull request on GitHub.

- If somebody from project contributors suggest changes then:

- Make the required updates.

- Re-run all test suites to ensure tests are still passing.

- Rebase your branch and force push to your GitHub repository (this will update your Pull Request). Basically you can

use `git commit -a --amend` and `git push --force origin my-fix-branch` in order to keep single commit in the feature

branch.

That's it! Thank you for your contribution!

## <a name="coding-standard"></a> Coding standard

To ensure consistency throughout the source code, keep these rules in mind as you are working:

- All features or bug fixes \*\*must be tested\*\*, ideally by both unit tests and integration tests.

- If you are implementing new feature or extending public API, you should \*\*document it\*\*.

- Follow defined rules in [tslint.json](tslint.json). All these will be checked by GitHub Actions

when you submit your PR.

Some highlights:

- use 2 spaces for indentation

- always use semicolons

- use single quotes where possible

- do not use `public` keyword (allowed only in constructor)

- prefer `const` over `let` (and do not use `var`)

## Commit Message Guidelines

The project have very precise rules over how git commit messages can be formatted. This leads to

\*\*more readable messages\*\* that are easy to follow when looking through the \*\*project history\*\*.

But also, git history is used to \*\*generate the change log\*\*.

The commit message format is borrowed from Angular projects and you can find

[more details in this document][commit-message-format].

## CI

We automatically run all pull requests through [GitHub Actions](https://github.com/mikro-orm/mikro-orm/actions).

- The test suite is ran against MongoDB, MySQL, PostgreSQL and SQLite on all supported Node.JS versions (`yarn test`).

- The code is validated against our Coding Standard (`yarn lint`).

- The code is checked by a static analysis tool ([codeclimate.com](https://codeclimate.com/github/mikro-orm/mikro-orm)).

If you break the tests, we cannot merge your code, so please make sure that your code is working before opening a pull request.

Also try to not introduce new code complexity and duplication issues. You can run the

static analysis tool locally - to do so, [follow this guideline][code-climate-guide].

[commit-message-format]: https://docs.google.com/document/d/1QrDFcIiPjSLDn3EL15IJygNPiHORgU1\_OOAqWjiDU5Y/edit#

[code-climate-guide]: https://github.com/codeclimate/codeclimate/blob/master/README.md