# How to contribute

The easiest way to contribute is to open an issue and start a discussion.

Then we can decide if and how a feature or a change could be implemented and if you should submit a pull requests with code changes.

Also read this first: [Being a good open source citizen](https://hackernoon.com/being-a-good-open-source-citizen-9060d0ab9732#.x3hocgw85)

## General feedback and discussions

Please start a discussion on the [core repo issue tracker](https://github.com/quartznet/quartznet/issues).

## Building

Run `build.cmd` or `build.sh` from the command line. This builds and runs tests.

## Bugs and feature requests?

Please log a new issue in the GitHub repo.

## Other discussions

https://gitter.im/quartznet/quartznet and https://groups.google.com/forum/#!forum/quartznet

## Filing issues

The best way to get your bug fixed is to be as detailed as you can be about the problem.

Providing a minimal project with steps to reproduce the problem is ideal.

Here are questions you can answer before you file a bug to make sure you're not missing any important information.

1. Did you read the [documentation](https://www.quartz-scheduler.net/documentation/index.html)?

2. Did you include the snippet of broken code in the issue?

3. What are the \*EXACT\* steps to reproduce this problem (including source/destination types, mapping configuration and execution)?

GitHub supports [markdown](http://github.github.com/github-flavored-markdown/), so when filing bugs make sure you check the formatting before clicking submit.

## Contributing code and content

Make sure you can build the code. Familiarize yourself with the project workflow and our coding conventions. If you don't know what a pull request is read this article: https://help.github.com/articles/using-pull-requests.

Before submitting a feature or substantial code contribution please discuss it with the team and ensure it follows the product roadmap. Here's a list of blog posts that are worth reading before doing a pull request:

\* [Open Source Contribution Etiquette](http://tirania.org/blog/archive/2010/Dec-31.html) by Miguel de Icaza

\* [Don't "Push" Your Pull Requests](http://www.igvita.com/2011/12/19/dont-push-your-pull-requests/) by Ilya Grigorik.

\* [10 tips for better Pull Requests](http://blog.ploeh.dk/2015/01/15/10-tips-for-better-pull-requests/) by Mark Seemann

\* [How to write the perfect pull request](https://github.com/blog/1943-how-to-write-the-perfect-pull-request) by GitHub

Here's a few things you should always do when making changes to the code base:

\*\*Commit/Pull Request Format\*\*

```

Summary of the changes (Less than 80 chars)

- Detail 1

- Detail 2

#bugnumber (in this specific format)

```

\*\*Tests\*\*

- Tests need to be provided for every bug/feature that is completed.

- Tests only need to be present for issues that need to be verified by QA (e.g. not tasks).

- If there is a scenario that is far too hard to test there does not need to be a test for it.

- "Too hard" is determined by the team as a whole.