# Contributing to Concurrent Ruby

You want to contribute? Thank you! Concurrent Ruby is work of [many contributors](https://github.com/ruby-concurrency/concurrent-ruby/graphs/contributors). You're encouraged to submit [pull requests](https://github.com/ruby-concurrency/concurrent-ruby/pulls), [propose features and discuss issues](https://github.com/ruby-concurrency/concurrent-ruby/issues). When in doubt, ask a question in the [Concurrent Ruby gitter.im chatroom](https://gitter.im/ruby-concurrency/concurrent-ruby).

#### Find Something to Work on

If you want to contribute but aren't sure what to work on, there are tasks marked with [\*\*looking-for-contributor\*\*](https://github.com/ruby-concurrency/concurrent-ruby/issues?q=is%3Aissue+is%3Aopen+label%3Alooking-for-contributor) label. Complete list of tasks can be found on [issues page](https://github.com/ruby-concurrency/concurrent-ruby/issues). We appreciate your help.

Before starting, feel free to chat with us on [gitter](https://gitter.im/ruby-concurrency/concurrent-ruby).

#### Fork the Project

Fork the [project on Github](https://github.com/ruby-concurrency/concurrent-ruby) and check out your copy.

```

git clone https://github.com/contributor/concurrent-ruby.git

cd concurrent-ruby

git remote add upstream https://github.com/ruby-concurrency/concurrent-ruby.git

```

#### Create a Topic Branch

Make sure your fork is up-to-date and create a topic branch for your feature or bug fix.

```

git checkout master

git pull upstream master

git checkout -b my-feature-branch

```

#### Bundle Install, Build and Test

Ensure that you can build the project and run tests.

Please note that Concurrent Ruby includes native Java and (optional) C extensions. When building you will need the appropriate build tools for your platform. If you don't have the build tools installed that's OK. The tests can still be run. A few tests may fail, but these are just the tests that verify that the extensions exist. You can usually ignore these locally if you are working on the pure Ruby parts of the library (which is most of the lib).

```

bundle install

bundle exec rake

```

#### Write Tests

Try to write a test that reproduces the problem you're trying to fix or describes a feature that you want to build. Add to [specs](https://github.com/ruby-concurrency/concurrent-ruby/tree/master/spec).

We definitely appreciate pull requests that highlight or reproduce a problem, even without a fix.

#### Write Code

Implement your feature or bug fix.

Make sure that `bundle exec rake` completes without errors.

#### Follow the Guidelines

There are a few guidelines which we follow when adding features. Consider that submissions which do not follow these guidelines will require modification before acceptance.

\* \*\*No downstream dependencies:\*\* Concurrent Ruby is a foundational library used by major projects like [Rails](http://rubyonrails.org/). Our downstream dependencies become everyone's dependencies. Because we cannot guarantee that downstream projects meet our development standards, it's best for everyone if we simply aviod dependencies.

\* \*\*Do not monkey patch Ruby:\*\* Changing Ruby for our convenience affects every gem in every project that uses Concurrent Ruby. Monkey patching Ruby may change the behavior of other libraries in unexpected ways and destabilize projects which depend on us.

\* \*\*Do not pollute the global namespace:\*\* Putting all our code within the `Concurrent` module guarantees that there will be no namespace collisions with other gems or the projects which depend on us.

\* \*\*No global state:\*\* We are removing global state we have.

\* \*\*Minimize per-object configuration:\*\* Ruby makes programmers happy. One of Ruby's charms is its simplicity. Concurrent Ruby aims to mirror this simplicity. Advanced configuration options are encouraged when they provide value, but every abstraction should have reasonable defaults that meet the needs of most users.

\* \*\*Provide explicit behavior and guarantees:\*\* Our APIs should be concrete and clearly define what they do (and don't do). Users of Concurrent Ruby should never be surprised by unexpected behavior or be given guarantees we cannot keep.

\* \*\*Eat our own dog food:\*\* Concurrent Ruby provides a rich set of low-level (internal and public) classes which provide strong guarantees and are optimized for performance across platforms. All our high-level abstractions should make use of these tools.

#### Write Documentation

Document any external behavior in the [README](README.md).

#### Commit Changes

Make sure git knows your name and email address:

```

git config --global user.name "Your Name"

git config --global user.email "contributor@example.com"

```

Writing good commit logs is important. A commit log should describe what changed and why.

```

git add ...

git commit

```

#### Push

```

git push origin my-feature-branch

```

#### Make a Pull Request

Go to https://github.com/contributor/concurrent-ruby and select your feature branch. Click the 'Pull Request' button and fill out the form. Pull requests are usually reviewed within a few days.

#### Rebase

If you've been working on a change for a while, rebase with upstream/master.

```

git fetch upstream

git rebase upstream/master

git push origin my-feature-branch -f

```

#### Update CHANGELOG

Update the [CHANGELOG](CHANGELOG.md) with a description of what you have changed.

#### Check on Your Pull Request

Go back to your pull request after a few minutes and see whether it passed muster with Travis-CI and AppVeyor. Everything should look green, otherwise fix issues (amending your commit).

Please note that testing concurrency is hard. Very hard. We have a few tests that occasionally fail due (mostly) to incorrect synchronization within the test itself. If everything passes locally but you see an error on CI, it's possibly you've become victim to one of the tests. Don't worry, the Concurrent Ruby team reviews the tests output of all failed CI runs and will let you know if the failing test is unrelated to your commit.

#### Thank You

Please do know that we really appreciate and value your time and work. We love you, really.

And also, a special thank you to the [Volt](https://github.com/voltrb/volt) team for writing an awesome guide for contributors. We have borrowed liberally from theirs.