# We :heart: Your Contributions

Thank you for taking your time to contribute to Rubinius. This guide will help you spend your time well, and help us keep Rubinius focused and making valuable progress.

Above all, life is short. Make it sweet. Be kind. :smile:

Now, let's dive into the beautiful :boom: messes. Of the many things that can go wrong, most fall into a pretty small set of categories:

1. Build & Packaging

1. OS & Platform

1. Ruby Compatibility

1. C-API Compatibility

1. Process Abort | Hang

1. Performance

1. Rubinius Capabilities

1. Feature Requests

Each of these has specific notes that will help you in the guides below, but there are a few things that all contributors should keep in mind.

## General Notes

It can feel intimidating to contribute to a complex project, but it can also be exhilerating and fun. These general notes will help everyone participate in this communal activity.

1. Follow the [Code of Conduct](https://rubinius.com/code-of-conduct/)

1. There are no "getting started"-type tickets or labels because everyone is encouraged to dive into whatever interests them. There's no expectation that you "work you way up" by doing documentation or clean-up tasks. Just pick something you're curious about and start working on it.

1. The Rubinius experiment has its own perspective, it's not trying to immitate another project. Be open to learning about it.

1. Small changes make huge differences. We will happily accept a PR making a single character change if it helps move forward. Don't wait to have everything working.

1. What you choose to spend your time on is a powerful vote about what is important. Tickets that don't have any code contributed for them will be closed after one month with an explanation and reference to this guide.

1. Check the closed issues before opening your issue. If an issue was closed previously because no code was contributed, a new issue without code will be closed as a duplicate.

1. Try to follow the existing style of the code (there's a lot of code). When in doubt, just ask.

1. Life is hard. Make it a bit softer. Be kind. :smile:

The focus for Rubinius in the near term is on the following capabilities:

1. Instruction set

1. Debugger

1. Profiler

1. Just-in-time compiler

1. Concurrency

1. Garbage collector

Contributions in the form of PRs for any of these areas are appreciated. Once these capabilities are more developed, all the categories below will be much easier to work on.

## Build & Packaging

The current build system depends on having Ruby installed. This dependency is being removed. Once it is, Rubinius will be much easier to package for various operating systems and packaging systems.

Issues for building or packaging on macOS and Ubuntu (Rubinius runs CI on Travis on Ubuntu) will be fixed.

Pull requests to fix issues for building or packaging on other operating systems or package managers will be happily accepted, but issues without code fixes will be closed as noted above.

Tag your issue with "Build & Packaging".

## OS & Platform

The same guidelines apply to operating system and platform issues as for build and packaging issues above.

Tag your issue with "OS & Platform".

## Ruby Compatibility

Rubinius will continue to maintain compatibility with the current stable version of Matz's Ruby Implementation (MRI) to the extent possible.

To support Ruby compatibility, we need specs and code. If you contribute code, be aware of any licensing issues that may affect the code you contribute.

Tag your issue with "Ruby Language Compatibility".

## C-API Compatibility

In general, the MRI C-API for Rubinius is deprecated, but that doesn't mean it will be going anywhere soon.

Compatibility for C-extensions will continue to be evaluated on a case-by-case basis. However, C-extensions that depend on functions that examine or manipulate built-in MRI data structures will never be supported.

Tag your issue with "C-API Compatibility".

## Process Abort | Hang

Much of the internals of Rubinius have been mostly or completely rewritten in the past couple years. This includes the garbage collector, concurrency facilities, Fibers, much of the instruction set, and a migration away from "primitive" functions that implement Ruby features.

Tag your issue with "Process Abort | Hang".

## Performance

Performance is a complex affair, but it generally reduces to a simple concept: To go faster, do less.

Until the core capabilities listed above are more developed, most performance work is very difficult. But we gladly accept PRs that fix performance issues.

Tag your issue with "Performance".

## Rubinius Capabilities

See the list above of the near term focus for Rubinius.

Tag your issue with "System Capabilities".

## Feature Requests

In general, Rubinius enthusiastically supports experimentation and welcomes ideas for features that enhance developer ability to create amazing software.

Some features won't be quite right for Rubinius, but the best way to know is to have working code we can play with and react to, so consider adding some code (even a little bit) to your feature request. If feature requests don't get any code votes, the issue will be closed according to the policy listed above.

Tag your issue with "Feature Request".

## We Missed Something :sweat\_smile:

If you don't see what you need here, come ask us a question :point\_right: <a href="https://gitter.im/rubinius/rubinius?utm\_source=badge&utm\_medium=badge&utm\_campaign=pr-badge"><img src="https://badges.gitter.im/Join%20Chat.svg" /></a>

Thank you! :rocket: :rainbow: :unicorn: