# Contributing to Overcommit

## Bug Reports

\* Ensure that your issue [has not already been reported][1]. It may already be

fixed!

\* Include the steps you carried out to produce the problem.

\* Include the behavior you observed along with the behavior you expected, and

why you expected it.

\* Try setting the `OVERCOMMIT\_DEBUG` environment variable to enable the display

of additional verbose output from executed commands.

\* Include the stack trace and any debugging output reported by Overcommit.

## Feature Requests

We welcome feedback with or without pull requests. If you have an idea for how

to improve the tool, great! All we ask is that you take the time to write a

clear and concise explanation of what need you are trying to solve. If you have

thoughts on \_how\_ it can be solved, include those too!

The best way to see a feature added, however, is to submit a pull request.

## Pull Requests

\* Before creating your pull request, it's usually worth asking if the code

you're planning on writing will actually be considered for merging. You can

do this by [opening an issue][1] and asking. It may also help give the

maintainers context for when the time comes to review your code.

\* Ensure your [commit messages are well-written][2]. This can double as your

pull request message, so it pays to take the time to write a clear message.

\* Add tests for your feature. You should be able to look at other tests for

examples, especially if you're contributing a pre-commit hook.

Speaking of tests, we use `rspec`, which can be run like so:

```bash

bundle exec rspec

```

\* Submit your pull request!

All pull requests will be tested against [Travis CI][3], where the following

commands are run against multiple versions of Ruby:

```bash

bundle exec rspec

bundle exec overcommit --run

```

Ensuring your changes pass for the above commands before submitting your pull

request will save you time having to fix those changes. Better yet, if you

[install Overcommit](README.md#installation) hooks into your forked repo, a lot

of these checks will be done automatically for you!

### Naming Hooks

Hooks should be named in camel case format (e.g. `RuboCop`) with acronyms only

capitalizing the first letter in the series (e.g. SCSS Lint becomes `ScssLint`).

If a tool has a specific capitalization that is odd, follow that capitalization.

For example, `Scalastyle` is written with a lowercase "s" rather than

camel-cased as `ScalaStyle`, so the `Scalastyle` hook follows that convention.

Exceptions to this rule are tools that begin with a lowercase

letter&mdash;these should be capitalized.

Lastly, unless a tool has a particularly unique or descriptive name, include

an additional prefix to help categorize it (e.g. `Java` in `JavaCheckstyle`),

so it is easier for others to find hooks in the [README](README.md).

The reasoning for this perhaps odd naming scheme is to strike a balance between

consistency, familiarity for those who already know the tool, and Overcommit's

ability to deduce the name of a hook from its filename and vice versa.

[1]: https://github.com/sds/overcommit/issues

[2]: https://medium.com/brigade-engineering/the-secrets-to-great-commit-messages-106fc0a92a25

[3]: https://travis-ci.org/

## Code of conduct

This project adheres to the [Open Code of Conduct][code-of-conduct]. By

participating, you are expected to honor this code.

[code-of-conduct]: https://github.com/civiccc/code-of-conduct