# Contribute

? Thanks for thinking about contributing to nock! We, the maintainers, are glad you're here and will be excited to help you get started if you have any questions. For now, here are some basic instructions for how we manage this project.

Please note that this project is released with a [Contributor Code of Conduct](./CODE\_OF\_CONDUCT.md). By participating in this project you agree to abide by its terms.

\*\*Table of Contents\*\*

<!-- toc -->

- [Branching](#branching)

- [Backport a fix / feature to a previous version](#backport-a-fix--feature-to-a-previous-version)

- [Submit a Beta / Next release](#submit-a-beta--next-release)

- [Formatting & linting](#formatting--linting)

- [Commit Message conventions](#commit-message-conventions)

- [Generate README TOC](#generate-readme-toc)

- [Running tests](#running-tests)

- [Running only a single test](#running-only-a-single-test)

- [Release Process](#release-process)

- [GitHub Apps](#github-apps)

- [Becoming a maintainer](#becoming-a-maintainer)

- [Who is funding work on Nock?](#who-is-funding-work-on-nock)

- [Priority Support](#priority-support)

- [Is there a chat where I can talk to other maintainers and users?](#is-there-a-chat-where-i-can-talk-to-other-maintainers-and-users)

<!-- tocstop -->

## Branching

\*\*TLDR: Most pull requests should be opened against the `beta` branch.\*\*

We use [semantic-release](https://github.com/semantic-release/semantic-release) to automate the release of new versions based on semantic commit messages as described above. Depending on the branch that a new pull request is merged into, it is published to different versions using different [npm dist-tags](https://docs.npmjs.com/cli/dist-tag).

- `main`: Publish with `@latest`, or if the commits have been merged into `next` branch before, update the `@latest` dist-tag to point at the new version

- `next`: Publish with `@next`

- `beta`: Publish to `X.0.0-beta.Y`, where `X` is the next breaking version and `Y` is number that gets increased with each release.

- `[Version].x`: For example `10.x`, where `10` is the major release number. No breaking changes are allowed on this branch. Publishes fix/feature versions to `[Version]` using a `@release-@[version].x` release tag

### Backport a fix / feature to a previous version

If you want to \*\*backport a fix\*\* to version 10, and the last version released on version 10 is `10.0.4`, then you have to follow these steps

1. Create the `[Version].x` branch unless it already exists. If you don’t have write access, ask one of the maintainers to do it for you. In this example, the command is `git checkout -b 10.x v10.0.4`.

2. Push the branch to GitHub: `git push -u origin $(git symbolic-ref --short HEAD)`

3. Create a new branch based on `10.x`

4. Make your changes

5. Submit the pull request against the `10.x` branch

### Submit a Beta / Next release

Create a new branch based off `beta` or `next`, depending on what you want. Make your changes and submit them to the same branches. If either `beta` or `next` is outdated, ask a maintainer to re-create them from `main`

## Formatting & linting

This project formats its source code using Prettier. The most enjoyable way to

use Prettier is to let it format code for you when you save. You can [integrate

it into your editor][integrate prettier].

[integrate prettier]: https://prettier.io/docs/en/editors.html

If you don't want to integrate with your editor, you can run `npm run prettier`

instead.

Semantic issues are checked with ESLint via `npm run lint`.

## Commit Message conventions

`nock` releases are automated using [semantic-release](https://github.com/semantic-release/semantic-release).

To automatically calculate the correct version number as well as changelogs,

three commit message conventions need to be followed

- Commit bug fixes with `fix: ...` or `fix(scope): ...` prefix in commit subject

- Commit new features with `feat: ...` or `feat(scope): ...` prefix in commit subject

- Commit breaking changes by adding `BREAKING CHANGE:` in the commit body

(not the subject line)

Other helpful conventions are

- Commit test files with `test: ...` or `test(scope): ...` prefix

- Commit changes to `package.json`, `.gitignore` and other meta files with

`chore(filename-without-ext): ...`

- Commit changes to README files or comments with `docs: ...`

- Code style changes with `style: standard`

Don't stress this; if necessary the maintainers can fix commit message(s) of a pull request using the `squash & merge` button.

## Generate README TOC

Make sure to update the README's table of contents whenever you update the README using the following npm script.

```

$ npm run toc

```

## Running tests

All of the tests work offline. None of them should hit the network.

```

$ npm test

```

### Running only a single test

When you're writing a test for a specific purpose, it may be useful to only run that test. In order to do this, you can pass a `only` option as the second argument to the [`tap`](https://www.node-tap.org/) testrunner. For instance in `tests/example.js`, like so:

```js

test('description', { only: true }, t => {

// ...

})

```

And then:

```sh

$ tap --only tests/example\_file.js

```

## Release Process

All of our releases are automated using [semantic-release](https://github.com/semantic-release/semantic-release). The commit messages pushed to the main branch trigger new releases. Semantic-release requires that commits follow certain conventions, [described above](#commit-message-conventions). semantic-release creates a GitHub release, adds release notes and publishes the new version to npm. This is why we do not store release notes in the [`CHANGELOG`](CHANGELOG.md) file - they're already on GitHub.

We use @nockbot as a separate account for releases, because npm tokens cannot be scoped to a single package. This improves our security model in case of a data breach involving npm tokens. @nockbot's credentials were set up by @gr2m; contact him if for any reason you need to change this in the future.

## GitHub Apps

We use several GitHub apps to help maintain this repository. While we would like to address every issue and while we would like to be on hand to support every person, Nock is pretty much entirely volunteer run, and we simply don't have the time to do everything. Please don't be offended if an automated app posts in your issue! We're doing what we can with with we have.

Currently, we use the [Stale](https://github.com/apps/stale) and [Lock](https://github.com/apps/lock) apps to mark old issues as stale, and to lock issues which have been closed to stop drive-by comments. You can see the configuration files for these in [.github/](.github).

## Becoming a maintainer

So you want to do more than file a bug or submit a PR? Awesome!

Nock is actively interested in having more maintainers. That means that we would love to have you (yes, you) get more involved if you want to! We don't have strict tests that you need to pass to become a maintainer. Instead, all we want is to find people who are frequent contributors, understand what Nock does, and are eager to help make this tool better.

Here are some things you can do today to actively show the Nock team that you're interested in helping out in the long term:

- \*\*Triage issues!\*\* We have more issues than we have bandwidth to deal with. For some of these issues, there are no labels, no comments suggesting how the issue could be resolved, and no follow-up after months or years of silence. It would be great if the issues actively reflected the state of the project. Go through old issues and suggesting labels in comments, ask for more information, and generally help out with the resolution process. This would be a great help!

- \*\*Review PRs.\*\* We have a lot of open PRs! Some of these are probably ready to merge, and some of them probably need more work. Any extra eyes on PRs are encouraged. Comment on code patterns you think need work, suggest extra tests, and let us know if a PR 'LGTM' ("looks good to me"). The more reviewers we have, the faster we can merge issues, the better this project becomes.

- \*\*Help out!\*\* If someone files a bug and no one has responded yet, see if you can resolve it! Suggest PRs, or file them yourself. While most contributors are going to only be interested in their own bugs, great maintainers help out with other people's bugs. This is one of the best ways to become an expert at Nock (and Node.js, JavaScript, or pretty much any project) - helping others.

- \*\*Write docs.\*\* Are our docs missing something important? Are they not clear? Could they be better? Open a PR!

- \*\*Suggest examples.\*\* Right now, we have a few examples, but we could always have more. Submit small example files and tutorials. At some point, we'll have to work on a better way to display these - for now, it's great to show others how to solve difficult mocking problems easily with Nock.

- \*\*Refactor.\*\* This is one of the hardest things to do, but one of the most useful. Go through the code, and find examples where it could be written better - with better variable names, more useful abstractions, and more elegant patterns. Taking away ten lines of code that are unnecessary is more valuable than submitting a hundred new lines, sometimes. Open a PR or a comment and defend your position; ask for feedback.

Once you've been around for a bit, ask a current Maintainer - one of [the team members](https://github.com/orgs/nock/people) - whether you can be elevated to Maintainer status and given permissions to close issues and merge PRs. We're interested in how well you know what Nock is about, and how involved you are in the community - not where you're from, how good your English is, or whether or not you can pass a whiteboard test blindfolded. If you think that you've been helpful, let us know. We're friendly, promise. :)

## Who is funding work on Nock?

Nock is entirely self-funded by [our Open Collective](https://opencollective.com/nock). Open Collective allows us to get contributions from users and companies, without giving away our open source IP. Think of it like a kickstarter, or an old-fashioned system of art patrons.

If you'd like to give money to support Nock, please do go to our profile and check it out.

## Priority Support

As part of backing on Open Collective, we tag some issues as a priority if the user who opens them is also a backer, using the [Open Collective Bot](https://github.com/marketplace/open-collective-bot). In general, as maintainers we will give priority to support people who are working on these issues. The money in the Open Collective goes to the entire project, however, and maintainers are volunteers.

## Is there a chat where I can talk to other maintainers and users?

We don't have a chat for a variety of reasons - mainly, because our maintainers' time is limited, and we don't want to be on the hook for enforcing codes of conduct or for checking in every day. However, we encourage you to open issues about anything related to Nock! Issues really are the best way to keep the conversation focused on development.

We often have video calls to pair program or to hack together on Nock. Join us! Look at the issues for a notification about these, and, if there isn't one planned, open an issue and suggest one.

On that note, thanks for reading this far. Go mock some tests now, there's nothing left to read here. :)