# Contributing

Your contributions are always welcome!

## Process

### Environment

For development with `git-secret` you should have these tools:

- git

- bash

- bundler

- gawk

- gnupg (or gnupg2), see below if not packaged by your distribution/OS (i.e. MacOS)

- ruby

- sha256sum (on freebsd and MacOS `shasum` is used instead)

- [shellcheck](https://github.com/koalaman/shellcheck)

To test `git-secret` using [test-kitchen](https://kitchen.ci/), which is optional and uses docker to test on multiple distributions,

you will also need:

- [docker](https://www.docker.com/)

- [test-kitchen](https://kitchen.ci/)

The below only required if dealing with manuals, `gh-pages` or releases:

- ruby, ruby-dev

- [aspell](http://aspell.net/), to check your changes for spelling errors

### Environment MacOS

- install [Homebrew](https://brew.sh/)

- install gnupg2 with `brew install gnupg2`

#### For docker/test-kitchen (optional, for testing multiple distros locally using docker)

- install Docker for Mac

- install ruby2.6 and kitchen dependencies with

brew install rbenv ruby-build rbenv-vars;

rbenv install 2.6.3; rbenv rehash; rbenv global 2.6.3;

(You can also use `rvm` instead of `rbenv`, but brew packages `rbenv` for you.)

then use

gem install bundler kitchen-ansible serverspec kitchen-docker kitchen-verifier-serverspec

If you have trouble getting test-kitchen and docker working on your mac to test git-secret with, see #534

or let us know by filing an issue.

### Getting started

1. Create your own or pick an opened issue from the [tracker][tracker]. Take a look at the [`help-wanted` tag][help-wanted]

2. Fork the git-secret repo and then clone the repository using a command like `git clone https://github.com/${YOUR\_NAME}/git-secret.git`

3. Make sure that everything works on the current platform by running `make test`.

You can also try the experimental `SECRETS\_TEST\_VERBOSE=1 make test`, which will

show you a lot of debug output while the tests are running.

Note that 'experimental' features may change or be removed in a future version of `git-secret`.

4. If you want to test on multiple operating systems, [Run local CI tests](#running-local-ci-tests) (optional; this will

automatically happen on [Travis-CI](https://travis-ci.org/sobolevn/git-secret) when you submit a PR).

Running the CI tests locally is optional. The tests will happen automatically on Travis-CI

when you create a PR for `git-secret`, and again when any PR is merged.

To verify functionality on supported platforms use `bundle exec kitchen verify --test-base-path="$PWD/.ci/integration"`.

See `[test-kitchen](https://kitchen.ci/) and `kitchen help verify` for more info about using `kitchen verify`.

### Code style

New features and changes should aim to be as clear, concise, simple, and consistent

1. clear: make it as obvious as possible what the code is doing

2. concise: your PR should be as few characters (not just lines) of changes as \_reasonable\_.

However, generally choose clarity over being concise.

Clarity and conciseness can be in conflict with each other. But

it's more important for the code to be understandable than for it to be small.

Therefore favor writing clear code over making shorter diffs in your PRs.

3. simple: this dovetails with the previous two items.

git-secret is a security product, so it's best to have the code be easy to understand.

This also aids future development and helps minimize bugs.

4. consistent: Write code that is consistent with the surrounding code and the rest of the git-secret code base.

Every code base has its own conventions and style that develop and accrete over time.

Consistency also means that the inputs and outputs of git-secret should be as consistent as reasonable

with related Unix and git tools, and follow the 'rule of least surprise',

also known as the 'principle of least astonishment': <https://en.wikipedia.org/wiki/Principle\_of\_least\_astonishment>

We wrote this to clarify our thinking about how git-secret should be written. Of course, these are philosophical goals,

not necessities for releasing code, so balancing these four ideals \_perfectly\_ is both unwarranted and impossible.

### Writing PRs

If you're planning a large change to `git-secret` (for example, a lot of lines/characters of diffs, affecting multiple commands,

changing/adding a lot of behavior, or adding multiple command-line options), it's best to discuss the changes in an Issue first.

Also it's often best to implement larger or complex changes as a series of planned-out, smaller PRs,

each making a small set of specific changes. This facilitates discussions of implementation, which often come to light

only after seeing the actual code used to perform a task.

As mentioned above, we seek to be consistent with surrounding git and Unix tools, so when writing changes to git-secret,

think about the input, output, and command-line options that similar Unix commands use.

Our favor toward traditional Unix and git command-style inputs and outputs can also mean it's appropriate to

lean heavily on git and widely-used Unix command features instead of re-implementing them in code.

### Development Process

1. Firstly, you should setup git-secret's development git hooks with `make install-hooks`

This will copy the hooks from utils/hooks into .git/hooks/pre-commit and .git/hooks/post-commit

2. Make changes to the git secret files that need to be changed

3. When making changes to any files inside `src/`, for changes to take effect you will need to rebuild the `git-secret` script with `make clean && make build`

4. Run [`shellcheck`][shellcheck] against all your changes with `make lint`.

You should also check your changes for spelling errors using 'aspell -c filename'.

5. Add an entry to CHANGELOG.md, referring to the related issue # if appropriate

6. Change the .ronn file(s) in man/man1 and man/man7 to document your changes if appropriate

7. Now, add all your files to the commit with `git add --all` and commit changes with `git commit`.

Write a good commit message which explains your work

8. When running `git commit` the tests will run automatically, your commit will be canceled if they fail.

You can run the tests manually with `make clean build test`.

If you want to make a commit and not run the pre- and post-commit hooks, use 'git commit -n'

9. Push to your repository, and make a pull-request against `master` branch. It's ideal to have one commit per pull-request,

but don't worry, it's easy to `squash` PRs into a small number of commits when they're merged.

### Branches

We have two long-live branches: `master` for the git-secret code and man pages, and `gh-pages` for the static web site.

Development looks like this:

> `your-branch` -> `master`

- `master` branch is protected, so only fully tested code goes there. It is also used to create a new `git` tag and a `github` release

By convention, you can name your branches like `issue-###-short-description`, but that's not required.

The `gh-pages` branch is used for the pages at `git-secret.io`. See 'Release Process' below.

### Continuous integration

Local CI is done with the help [`test-kitchen`](http://kitchen.ci/). `test-kitchen` handles multiple test-suites on various platforms.

You can run our CI tests locally, but it is not strictly required in order to do development or testing of git-secret. When you have

`test-kitchen` installed, `bundle exec kitchen list` will output the list of test suites to be run against supported platforms.

Cloud CI is done with the help of [Travis-CI](https://travis-ci.org/sobolevn/git-secret), which handles testing on multiple environments using

- `Docker`-based jobs or so-called 'integration tests', which create a local release, install it with the package manager and then run unit-tests and system checks

- `OSX` jobs, which handle basic unit-tests on `MacOS` (Travis still calls MacOS 'OSX')

- Native `travis` jobs, which handle basic unit-tests and style checks

### Running local ci-tests with test-kitchen

Ci-tests are only necessary if you want to test git-secret on multiple OS'es using docker and test-kitchen,

like we do on travis-ci.

1. Install required gems with `bundle install`.

2. Run ci-tests with `bundle exec kitchen verify --test-base-path="$PWD/.ci/integration"`

### Writing tests

`git-secret` uses [bats-core](https://github.com/bats-core/bats-core) for testing.

See the files in tests/ and the `bats-core` documentation for details.

Because the output of many commands can be affected by the SECRETS\_VERBOSE environment

variable (which enables verbosity), it's best not to expect a particular number of lines of

output from commands.

### Release process

The release process is defined in the `git`-hooks and `.travis.yml`.

When creating a commit inside the `master` branch (it is usually a documentation and changelog update with the version bump inside `src/version.sh`) the

pre-commit and post-commit hooks will trigger three events.

- `pre-commit`: run the test suite will be locally

- `pre-commit`: generate and update the manuals and add them to the current commit with `make build-man`

- `post-commit`: trigger `make build-gh-pages`, which will update and push manuals to the [git-secret site][git-secret-site].

- `post-commit`: new `git` tag (such as v0.3.1) will be automatically created if the version is changed, using something like

```bash

if [[ "$NEWEST\_TAG" != "v${SCRIPT\_VERSION}" ]]; then

git tag -a "v${SCRIPT\_VERSION}" -m "version $SCRIPT\_VERSION"

fi

```

#### About GnuPG

Here are some links to gnupg documentation that might be useful for those working with git-secret:

- [GnuPG PDF Documentation](https://www.gnupg.org/documentation/manuals/gnupg.pdf)

- [GnuPG doc/DETAILS File](https://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob;f=doc/DETAILS)

#### Travis releases

After you commit a tag that matches the pattern '^v' and the tests succeed, scripts run on [Travis-CI](https://travis-ci.org/sobolevn/git-secret)

will publish new `deb` and `rpm` packages to [`bintray`][bintray].

(If you wish to override a previous release (\*be careful, this is discouraged\*) you will need to add `"override": 1` into `matrixParams`, see `deb-deploy.sh` and `rpm-deploy.sh`)

#### Manual releases

Releases to `brew` are made manually, and involve opening a PR on the [Homebrew Core](https://github.com/Homebrew/homebrew-core) repo .

To get started, see the

[Homebrew docs about Formulae-related PRs](https://docs.brew.sh/How-To-Open-a-Homebrew-Pull-Request#formulae-related-pull-request)

and `brew bump-formula-pr --help`

#### Dockerhub releases

[`Dockerhub`][Dockerhub] contains `Docker` images with different OSes used for testing. It is updated via a `github` webhook on commit into `master`.

[tracker]: https://github.com/sobolevn/git-secret/issues

[help-wanted]: https://github.com/sobolevn/git-secret/issues?q=is%3Aissue+is%3Aopen+label%3A%22help+wanted%22

[shellcheck]: https://github.com/koalaman/shellcheck

[git-secret-site]: http://git-secret.io

[bintray]: https://bintray.com/sobolevn

[Dockerhub]: https://hub.docker.com/r/sobolevn/git-secret/

### Downstream Packages

There are several distributions and packaging systems that may already have git-secret packaged for your distribution (although sometimes their versions are not the most current, and we recommend all users upgrade to 0.2.5 or above).

### Notes to Downstream Packagers (Those who make packages for specific OSes/distributions)

First of all, thank you for packaging git-secret for your platform! We appreciate it.

We also would like to welcome you to collaborate or discuss any issues, ideas or thoughts you have about

git-secret by submitting [issue report](https://github.com/sobolevn/git-secret/issues)

(which can also be feature requests) or

[pull requests](https://help.github.com/en/articles/creating-a-pull-request)

via the git repo at

[git-secret on github](https://github.com/sobolevn/git-secret)

Please let us know if there are any changes you'd like to see to the source,

packaging, testing, documentation, or other aspect of git-secret.

We look forward to hearing from you.

## Financial contributions

We also welcome financial contributions in full transparency on our [open collective](https://opencollective.com/git-secret).

Anyone can file an expense. If the expense makes sense for the development of the community, it will be "merged" in the ledger of our open collective by the core contributors and the person who filed the expense will be reimbursed.

## Credits

### Contributors

Thank you to all the people who have already contributed

to git-secret via commits to our git repository!

<a href="http://github.com/sobolevn/git-secret/graphs/contributors"><img src="https://opencollective.com/git-secret/contributors.svg?width=890" /></a>

### Backers

Thank you to all our backers! [[Become a backer](https://opencollective.com/git-secret#backer)]

<a href="https://opencollective.com/git-secret#backers" target="\_blank"><img src="https://opencollective.com/git-secret/backers.svg?width=890"></a>

### Sponsors

Thank you to all our sponsors! (please ask your company to also support this open source project by [becoming a sponsor](https://opencollective.com/git-secret#sponsor))

<a href="https://opencollective.com/git-secret/sponsor/0/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/0/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/1/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/1/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/2/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/2/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/3/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/3/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/4/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/4/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/5/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/5/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/6/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/6/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/7/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/7/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/8/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/8/avatar.svg"></a>

<a href="https://opencollective.com/git-secret/sponsor/9/website" target="\_blank"><img src="https://opencollective.com/git-secret/sponsor/9/avatar.svg"></a>