# Contributing to gitsh

## Contributing a feature

We love pull requests from everyone. By participating in this project, you

agree to abide by the thoughtbot [code of conduct].

[code of conduct]: https://thoughtbot.com/open-source-code-of-conduct

Here's a quick guide to begin contributing:

1. Clone the repo:

git clone https://github.com/thoughtbot/gitsh.git

2. Ensure \*\*GNU Readline\*\* and \*\*GNU Autotools\*\* are installed,

e.g. on macOS you would run:

brew install readline

brew install automake

3. Build the generated files. Some Ruby files in `gitsh` are generated by the

build system, and the tests won't run without them:

cd gitsh

./autogen.sh

bundle

RUBY=$(which ruby) ./configure

make

Note that setting `RUBY=$(which ruby)` will use your current Ruby version.

This isn't recommended for installing gitsh for day-to-day use, but is

recommended for development.

The `./configure` step will search for GNU Readline, but might fail if GNU

Readline isn't installed on your system, or if it finds an incompatible

implementation first. If this happens, you can set the `CPPFLAGS` and

`LDFLAGS` environment variables to tell gitsh where to find Readline.

For example, on macOS with Readline installed via Homebrew, you will need

something like this:

RUBY=$(which ruby) CPPFLAGS='-I/usr/local/opt/readline/include' \

LDFLAGS='-L/usr/local/opt/readline/lib' ./configure

4. Run the tests. We only take pull requests with passing tests, and it's great

to know that you have a clean slate:

make check

5. Add a test for your change. Only refactoring and documentation changes

require no new tests. If you are adding functionality or fixing a bug, we

need a test!

6. Make the test pass.

7. Fork the repo, push to your fork, and submit a pull request.

At this point you're waiting on us. We like to at least comment on, if not

accept, pull requests within three business days. We may suggest some changes or

improvements or alternatives.

Some things that will increase the chance that your pull request is accepted:

\* Include tests that fail without your code, and pass with it.

\* Update the documentation, especially the man page, whatever is affected by

your contribution.

\* Follow the [thoughtbot style guide][style-guide].

And in case we didn't emphasize it enough: we love tests!

## Testing

We use the autotools structure for running tests. To run the full suite,

use the `check` target:

make check

You can run a subset of the tests by file name:

env TESTS="spec/integration/tab\_completion\_spec" make -e check

The full test output is available in `test-suite.log`, and partial

output is available in the log for the test itself (e.g.

`spec/integration/arguments\_spec.log`).

## Manual testing

To run your cloned version of gitsh locally, simply run:

./bin/gitsh

## Releasing a new version

gitsh is packaged and installed using GNU autotools.

0. Make sure you're starting from a clean slate:

make distclean

git checkout master

1. Update the version number in `configure.ac`.

2. Update the `configure` script, `Makefile`, and other dependencies:

./autogen.sh

./configure

3. Commit your changes to `configure.ac`, `INSTALL`, and any other files that

were modified by the version bump:

git add .

git commit -m "Bump version: X.Y.Z"

git push

4. Build and publish the release:

make release\_build

make release\_push

make release\_clean

Alternatively, you can use a single command that will run them for you. If

anything goes wrong, this will be harder to debug:

make release

## Regular maintenance

### Updating supported Ruby versions

When a new version of Ruby is released, or an old version of Ruby reaches

end-of-life, we should update gitsh's requirements to match.

1. Update the build system (`configure.ac`):

- Change the `AX\_PROG\_RUBY\_VERSION` call to the minimum supported version.

- Change the list of possible binary names passed to `AC\_PATH\_PROGS` to

reflect the supported versions, e.g. if Ruby 2.6 is supported, then

`ruby26` should be included in the list.

2. Update the CI configuration in `.travis.yml`.

3. Update references to the minimum supported version in the documentation:

- Update the install instructions template in `INSTALL.in`.

- Update the generated `INSTALL` instructions (`./configure && make`).

4. Update the Ruby dependency in package manager templates to the minimum

supported version:

- `homebrew/gitsh.rb.in`

- `arch/PKGBUILD.in`

### Updating gem dependencies

Gems can be updated in the usual way:

bundle update

When updating Rubygems for gitsh, there are a few things to consider:

- The Gemfile is only used by developers and maintainers. When gitsh is

distributed, all of the gems' files are included in the distribution.

- Use the minimum supported Ruby version when updating gems to avoid

installing versions that are incompatible with older Rubies.

[style-guide]: https://github.com/thoughtbot/guides/tree/master/style#ruby