# Contributing to SSHKit

\* Use [Stack Overflow][so] for Capistrano-related how-to questions and support

\* [\*\*Don't\*\* push your pull request](http://www.igvita.com/2011/12/19/dont-push-your-pull-requests/)

\* [\*\*Do\*\* write a good commit message](http://365git.tumblr.com/post/3308646748/writing-git-commit-messages)

## Getting help

Most users encounter SSHKit primarily by using Capistrano. If you have a question about using

SSHKit in context of Capistrano, please use the [capistrano tag on Stack Overflow][so].

If you are using SSHKit directly, or if you think you've found a bug in SSHKit, please open a

[GitHub issue](https://github.com/capistrano/sshkit/issues).

[so]: http://stackoverflow.com/questions/tagged/capistrano

## Ruby versions

You can see the current ruby versions we support in [.travis.yml](.travis.yml).

Obviously, any language features you use must be available in the oldest version we support.

Therefore, it's often helpful to develop / test against the oldest version to avoid accidentally

using unsupported features.

## Tests

SSHKit has a unit test suite and a functional test suite. Some functional tests run against

[Vagrant](https://www.vagrantup.com/) VMs. If possible, you should make sure that the

tests pass for each commit by running `rake` in the sshkit directory. This is in case we

need to cherry pick commits or rebase. You should ensure the tests pass, (preferably on

the minimum and maximum ruby version), before creating a PR.

Pull requests are much more likely to be accepted if you write a tests for the new functionality

you are adding. If you are fixing a bug, it would be great if you could add a test to

expose the bug you are fixing to show that the behaviour is fixed by your changes.

We use [Travis CI](https://travis-ci.org/capistrano/sshkit) to run the tests on different

ruby versions.

\*\*The Travis build does not run the functional tests,

so make sure all the tests pass locally before creating your PR.\*\*

## Coding guidelines

This project uses [RuboCop](https://github.com/bbatsov/rubocop) to enforce standard Ruby coding

guidelines. Currently we run RuboCop's lint rules only, which check for readability issues

like indentation, ambiguity, and useless/unreachable code.

\* Test that your contributions pass with `rake lint`

\* The linter is also run as part of the full test suite with `rake`

\* Note the Travis build will fail and your PR cannot be merged if the linter finds errors

## Changelog

Most changes should have an accompanying entry in the [CHANGELOG](CHANGELOG.md), unless they

are minor documentation / config changes. This is incredibly important so that our users can

see the affect of new versions without having to trawl through the commits.

## Breaking changes

We adhere to [semver](http://semver.org/) so breaking changes will require a major release.

For breaking changes, it would normally be helpful to discuss them by raising a 'Proposal' issue

or PR with examples of the new API you're proposing

[before doing a lot of work](https://www.igvita.com/2011/12/19/dont-push-your-pull-requests/).

Bear in mind that breaking changes may require many hundreds / thousands of users to update their

code.

You can use the [BREAKING\_API\_WISHLIST](BREAKING\_API\_WISHLIST.md) to record issues / PRs where

API changes have been discussed, and not implemented.