# How to contribute to Helpy

Welcome, and thanks for contributing to Helpy. Together we are building

the best customer support platform in the world. Here are some of the ways

you can contribute:

\*\*If you find a security bug, do not report it through GitHub. Please send an

e-mail to [hello@helpy.io](mailto:hello@helpy.io)

instead. See [SECURITY.md](/SECURITY.md) for more.a\*\*

- Report or fix Bugs- Before reporting a new issue, please be sure that the issue wasn't already

reported or fixed by searching on GitHub through our [issues](https://github.com/helpyio/helpy/issues).

When creating a new issue, be sure to include a \*\*title and clear description\*\*,

as much relevant information as possible, and either detailed steps to recreate the

issue or a test case example - Helpy has a lot

of moving parts and it's functionality can be affected by third party gems, so

we need as much context and details as possible to identify what might be broken

for you.

- Refactoring- There are many opportunities to refactor code within Helpy to

make it more concise, faster, more readable, remove duplication, etc.

- Improve test coverage- As with any large and growing codebase, test coverage

is not always as good as it could be. Help improving test coverage is always

welcome and will help you learn how Helpy works. We use Minitest exclusively.

- Translate the project- The community has already translated Helpy into 20+

languages, but there are many more waiting. We need help getting Helpy

translated into as many locales as possible! Please see the guide to

translation for more details.

- Build new features. There is a backlog of new features that the community would

like to see built. Check out our roadmap for more insight on this, and if you would like

to take one on, please get in touch with us to make sure someone is not already

working on it.

Before sending a new Pull Request, take a look on existing Pull Requests and Issues

to see if the proposed change or fix has been discussed in the past, or if the

change was already implemented but not yet released.

We expect new Pull Requests to include enough tests for new or changed behavior,

and we aim to maintain everything as most backwards compatible as possible,

reserving breaking changes to be shipped in major releases when necessary.

If your Pull Request includes new or changed behavior, be sure that the changes

are beneficial to a wide range of use cases or it's an application specific change

that might not be so valuable to other applications. Some changes can be introduced

as a new `Helpy-something` gem instead of belonging to the main codebase.

#General Guidelines:

- Join us on Slack. Let me know you wish to contribute.

- Make your PRs granular. They should only include one piece of functionality per PR.

- Check the roadmap. If you want to build a feature, contact us to make sure

no one else is already working on it

- You must provide passing test coverage. We use minitest, see

http://www.rubypigeon.com/posts/minitest-cheat-sheet/?utm\_source=rubyweekly&utm\_medium=email

- You also must expose functionality to the API. We use Grape. API methods should be

tested as well.

- If your feature/bug fix/enhancement adds or changes text in the project, please create

i18n strings in en.yml and any other locales you can.

- We are hugely concerned with user experience, and a nice UI. Oftentimes that means

we may take what you have contributed and “dress it up” or ask you to do the same.