# How to contribute

We are glad you are reading this, because we need volunteer developers

to help this project come to fruition.

If you don't have anything you are working on we have a

[\*\*list of newbie friendly issues\*\*](https://github.com/Harvey-OS/harvey/issues?utf8=%E2%9C%93&q=is%3Aissue+is%3Aopen+label%3A%22help+wanted%22+label%3A%22good+first+issue%22)

you can help out with.

If you haven't already, come find us on our

[mailing list](https://groups.google.com/forum/#!forum/harvey). We

want you working on things you're excited about.

Harvey, like most other open source projects, has a

[Code of Conduct](https://github.com/Harvey-OS/harvey/wiki/Code-of-Conduct)

that it expects its contributors and core team members to adhere to.

Here are some important resources:

\* Our [mailing list](https://groups.google.com/forum/#!forum/harvey)

is where the development discussion happens.

\* [Man Pages](https://sevki.io/harvey/sys/man/1/0intro) are where

you can documentation from head of the repository.

\* The [Issue Tracker](https://github.com/Harvey-OS/harvey/issues) is

our day-to-day project management space.

## Testing

We make use Travis-CI and make sure we can build your pull-requests

before we can accept your contributions.

## Submitting a patch

Harvey uses Github Pull Requests to accept contributions.

1. Clone the repository:

`git clone https://github.com/Harvey-OS/harvey.git`.

It is also possible to use `git` instead of `https` if you have an

SSH public key stored on Github:

`git clone git@github.com:Harvey-OS/harvey.git`.

This makes submitting contributions easier. For the rest of this

manual we assume to use https.

2. Check out a feature branch for your work on by executing:

`git checkout -b feature-name`.

For example, @keedon selected the branch name `statscrash` for

issue #70.

3. Make changes

4. Commit with a descriptive message and [signed-off-by:](https://github.com/docker/Harvey-OS/harvey/main/CONTRIBUTING.md#sign-your-work):

```

$ git commit -m -s "A brief summary of the commit

>

> A paragraph describing what changed and its impact."

```

For a representative example, look at @keedon's

[commit message](https://github.com/keedon/harvey/commit/09fe3a21fa8b42088bc8ad83287928e9e7cc96ef)

for issue #70 mentioned above. You can also use graphical git

tools such as `git gui` if you like.

5. Fork the repo (only once).

![harvey-os\_harvey\_\_a\_fresh\_take\_on\_plan\_9](https://cloud.githubusercontent.com/assets/429977/13457174/099fb5cc-e067-11e5-83ce-f65aa966a4a9.png)

6. Add the repo as a remote (every time you clone the repository)

`git remote add yourname https://github.com/yourusername/harvey.git`

where "yourname" is your github login name.

`git remote -v` should look like this:

```

$ git remote -v

yourname https://github.com/yourname/harvey.git (fetch)

yourname https://github.com/yourname/harvey.git (push)

origin https://github.com/Harvey-OS/harvey.git (fetch)

origin https://github.com/Harvey-OS/harvey.git (push)

```

7. Push your branch to your forked repository:

`git push yourname feature-name`

8. Create a pull request by going to

`https://github.com/yourname/harvey/pull/new/feature-name`

or clicking the PR button

![sevki\_harvey\_\_a\_64\_bit\_operating\_system\_based\_on\_plan\_9\_from\_bell\_labs\_and\_nix\_\_under\_gpl](https://cloud.githubusercontent.com/assets/429977/13457635/79359350-e069-11e5-987b-1b4fccc45372.png)

9. Add details of what you have worked on and your motivation.

![comparing\_harvey-os\_master\_\_\_sevki\_travis-trials\_ \_harvey-os\_harvey](https://cloud.githubusercontent.com/assets/429977/13457683/aa2a423a-e069-11e5-84cc-1173e33264cb.png)

When you send a pull request, we greatly appreciate if you include

`emu` output and additional tests: we can always use more test

coverage. Please follow our coding conventions and make sure all

of your commits are atomic in the sense of having one feature per

commit.

### Iterating on your work

- A Github pull request or a Gerrit CL roughly serve the same purpose:

they are feedback loops for people to give feedback and for you to

iterate on your work in response to that feedback, so be ready to

repeat steps!

- When iterating on your work continue committing to the same branch

and push the changes up to your fork. Github will track the changes

and update the pull request accordingly.

- Most of the time, when you are ready to submit your pull request,

someone else will have merged something to main, at which point

your branch will have been outdated, GitHub provides a convenient

way of updating your branch right from your pull request.

![build\_files\_by\_sevki\_ \_pull\_request\_\_53\_ \_harvey-os\_harvey](https://cloud.githubusercontent.com/assets/429977/13457994/4d9a3118-e06b-11e5-9898-f8574b5ce11d.png)

- When you click that button, GitHub will update your branch, at which

point you will have to `git pull yourname feature-name` and update

your local repo before committing more changes.

- If your changes conflict with something that was merged into the

branch, you will have to resolve the changes manually before

submitting the changes.

### Keeping up to date with the main branch

If you're working in a branch that is outdated with respect to the

master branch, just do a `git pull --rebase`. This will put your

changes after the pull. In the case that there would be conflicts, you

will have to solve them manually, but they are marked with something

like "`>>>>>HEAD`" and git will tell you about which files are in

conflict.

#### More information

- [How to use Pull Requests](http://help.github.com/pull-requests/)

- [GitHub Flow](https://guides.github.com/introduction/flow/)

- [Hub: A command line application for GH Flow](https://hub.github.com)

- [Video: Pull Requests](https://www.youtube.com/watch?v=81uKcXZoQ2A)

- [Video: Workflows](https://www.youtube.com/watch?v=EwWZbyjDs9c)

- [Github Desktop Client](https://desktop.github.com/)

- [Video: Undo, Redo & Rebase Your Git History](https://www.youtube.com/watch?v=W39CfI3-JFc)

- [Video: Git Rebase](https://www.youtube.com/watch?v=SxzjZtJwOgo)

### Sign your work

The sign-off is a simple line at the end of the explanation for the patch. Your

signature certifies that you wrote the patch or otherwise have the right to pass

it on as an open-source patch. The rules are pretty simple: if you can certify

the below (from [developercertificate.org](http://developercertificate.org/)):

```

Developer Certificate of Origin

Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.

1 Letterman Drive

Suite D4700

San Francisco, CA, 94129

Everyone is permitted to copy and distribute verbatim copies of this

license document, but changing it is not allowed.

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I

have the right to submit it under the open source license

indicated in the file; or

(b) The contribution is based upon previous work that, to the best

of my knowledge, is covered under an appropriate open source

license and I have the right under that license to submit that

work with modifications, whether created in whole or in part

by me, under the same open source license (unless I am

permitted to submit under a different license), as indicated

in the file; or

(c) The contribution was provided directly to me by some other

person who certified (a), (b) or (c) and I have not modified

it.

(d) I understand and agree that this project and the contribution

are public and that a record of the contribution (including all

personal information I submit with it, including my sign-off) is

maintained indefinitely and may be redistributed consistent with

this project or the open source license(s) involved.

```

Then you just add a line to every git commit message:

Signed-off-by: Joe Smith <joe.smith@email.com>

Use your real name (sorry, no pseudonyms or anonymous contributions.)

If you set your `user.name` and `user.email` git configs, you can sign your

commit automatically with `git commit -s`.

## Coding conventions

If you read the code you should get a hang of it but a loose listing

of our

[Style-Guide](https://github.com/Harvey-OS/harvey/wiki/Style-Guide)

exists, we recommend you check it out.

We have also automated the process via

[clang-format](http://clang.llvm.org/docs/ClangFormat.html)

so before you submit a change please format your diff.

[How to clang-format a diff](http://clang.llvm.org/docs/ClangFormat.html#script-for-patch-reformatting)

\_Adopted from [Open Government Contribution Guidelines](https://github.com/opengovernment/opengovernment/blob/master/CONTRIBUTING.md)\_