# Contributing to Gunicorn

Want to hack on Gunicorn? Awesome! Here are instructions to get you

started. They are probably not perfect, please let us know if anything

feels wrong or incomplete.

## Contribution guidelines

### Pull requests are always welcome

We are always thrilled to receive pull requests, and do our best to

process them as fast as possible. Not sure if that typo is worth a pull

request? Do it! We will appreciate it.

If your pull request is not accepted on the first try, don't be

discouraged! If there's a problem with the implementation, hopefully you

received feedback on what to improve.

We're trying very hard to keep Gunicorn lean and focused. We don't want it

to do everything for everybody. This means that we might decide against

incorporating a new feature. However, there might be a way to implement

that feature \*on top of\* Gunicorn.

### Discuss your design on the mailing list

We recommend discussing your plans [on the mailing

list](http://gunicorn.org/#community) before starting to code -

especially for more ambitious contributions. This gives other

contributors a chance to point you in the right direction, give feedback

on your design, and maybe point out if someone else is working on the

same thing.

### Create issues...

Any significant improvement should be documented as [a github

issue](https://github.com/benoitc/gunicorn/issues) before anybody starts

working on it.

### ...but check for existing issues first!

Please take a moment to check that an issue doesn't already exist

documenting your bug report or improvement proposal. If it does, it

never hurts to add a quick "+1" or "I have this problem too". This will

help prioritize the most common problems and requests.

### Conventions

Don't comment on closed issues or PRs, instead open a new issue and link it to

the old one.

Fork the repo and make changes on your fork in a feature branch:

- If it's a bugfix branch, name it XXX-something where XXX is the number

of the issue

- If it's a feature branch, create an enhancement issue to announce your

intentions, and name it XXX-something where XXX is the number of the

issue.

Submit unit tests for your changes. Python has a great test framework built

in; use it! Take a look at existing tests for inspiration. Run the full

test suite on your branch before submitting a pull request.

Make sure you include relevant updates or additions to documentation

when creating or modifying features.

If you are adding a new configuration option or updating an existing one,

please do it in `gunicorn/config.py`, then run `make -C docs html` to update

`docs/source/settings.rst`.

Write clean code.

Pull requests descriptions should be as clear as possible and include a

reference to all the issues that they address.

Code review comments may be added to your pull request. Discuss, then

make the suggested modifications and push additional commits to your

feature branch. Be sure to post a comment after pushing. The new commits

will show up in the pull request automatically, but the reviewers will

not be notified unless you comment.

Before the pull request is merged, make sure that you squash your

commits into logical units of work using `git rebase -i` and `git push

-f`. After every commit the test suite should be passing. Include

documentation changes in the same commit so that a revert would remove

all traces of the feature or fix.

Commits that fix or close an issue should include a reference like

`Closes #XXX` or `Fixes #XXX`, which will automatically close the issue

when merged.

Add your name to the THANKS file, but make sure the list is sorted and

your name and email address match your git configuration. The THANKS

file is regenerated occasionally from the git commit history, so a

mismatch may result in your changes being overwritten.

## Decision process

### How are decisions made?

Short answer: with pull requests to the gunicorn repository.

Gunicorn is an open-source project under the MIT License with an open

design philosophy. This means that the repository is the source of truth

for EVERY aspect of the project, including its philosophy, design,

roadmap and APIs. \*If it's part of the project, it's in the repo. It's

in the repo, it's part of the project.\*

As a result, all decisions can be expressed as changes to the

repository. An implementation change is a change to the source code. An

API change is a change to the API specification. A philosophy change is

a change to the relevant documentation. And so on.

All decisions affecting gunicorn, big and small, follow the same 3 steps:

\* Step 1: Open a pull request. Anyone can do this.

\* Step 2: Discuss the pull request. Anyone can do this.

\* Step 3: Accept or refuse a pull request. The relevant maintainer does this (see below "Who decides what?")

### Who decides what?

So all decisions are pull requests, and the relevant maintainer makes

the decision by accepting or refusing the pull request. But how do we

identify the relevant maintainer for a given pull request?

Gunicorn follows the timeless, highly efficient and totally unfair system

known as [Benevolent dictator for

life](http://en.wikipedia.org/wiki/Benevolent\_Dictator\_for\_Life), with

Benoit Chesneau (aka benoitc), in the role of BDFL. This means that all

decisions are made by default by me. Since making every decision myself

would be highly unscalable, in practice decisions are spread across

multiple maintainers.

The relevant maintainer for a pull request is assigned in 3 steps:

\* Step 1: Determine the subdirectory affected by the pull request. This might be src/registry, docs/source/api, or any other part of the repo.

\* Step 2: Find the MAINTAINERS file which affects this directory. If the directory itself does not have a MAINTAINERS file, work your way up the the repo hierarchy until you find one.

\* Step 3: The first maintainer listed is the primary maintainer who is assigned the Pull Request. The primary maintainer can reassign a Pull Request to other listed maintainers.

### I'm a maintainer, should I make pull requests too?

Primary maintainers are not required to create pull requests when

changing their own subdirectory, but secondary maintainers are.

### Who assigns maintainers?

benoitc.

### How can I become a maintainer?

\* Step 1: learn the component inside out

\* Step 2: make yourself useful by contributing code, bugfixes, support etc.

\* Step 3: volunteer on the irc channel (#gunicorn@freenode)

Don't forget: being a maintainer is a time investment. Make sure you

will have time to make yourself available. You don't have to be a

maintainer to make a difference on the project!

### What are a maintainer's responsibility?

It is every maintainer's responsibility to:

\* 1) Expose a clear roadmap for improving their component.

\* 2) Deliver prompt feedback and decisions on pull requests.

\* 3) Be available to anyone with questions, bug reports, criticism etc. on their component. This includes irc, github requests and the mailing list.

\* 4) Make sure their component respects the philosophy, design and roadmap of the project.

### How is this process changed?

Just like everything else: by making a pull request :)