# How to Contribute to TCMalloc

We'd love to accept your patches and contributions to this project. There are

just a few small guidelines you need to follow.

NOTE: If you are new to GitHub, please start by reading [Pull Request

howto](https://help.github.com/articles/about-pull-requests/)

## Contributor License Agreement

Contributions to this project must be accompanied by a Contributor License

Agreement. You (or your employer) retain the copyright to your contribution;

this simply gives us permission to use and redistribute your contributions as

part of the project. Head over to <https://cla.developers.google.com/> to see

your current agreements on file or to sign a new one.

You generally only need to submit a CLA once, so if you've already submitted one

(even if it was for a different project), you probably don't need to do it

again.

## Guidelines for Pull Requests

\* All submissions, including submissions by project members, require review.

We use GitHub pull requests for this purpose. Consult

[GitHub Help](https://help.github.com/articles/about-pull-requests/) for more

information on using pull requests.

\* If you are a Googler, it is preferable to first create an internal CL and

have it reviewed and submitted. The code propagation process will deliver

the change to GitHub.

\* Create \*\*small PRs\*\* that are narrowly focused on \*\*addressing a single concern\*\*.

When PRs try to fix several things at a time, if only one fix is considered

acceptable, nothing gets merged and both author's & review's time is wasted.

Create more PRs to address different concerns and everyone will be happy.

\* Provide a good \*\*PR description\*\* as a record of \*\*what\*\* change is being

made and \*\*why\*\* it was made. Link to a GitHub issue if it exists.

\* Don't fix code style and formatting unless you are already changing that line

to address an issue. Formatting of modified lines may be done using

`git clang-format`. PRs with irrelevant changes won't be merged. If you do

want to fix formatting or style, do that in a separate PR.

\* Unless your PR is trivial, you should expect there will be reviewer comments

that you'll need to address before merging. We expect you to be reasonably

responsive to those comments, otherwise the PR will be closed after 2-3 weeks

of inactivity.

\* Maintain \*\*clean commit history\*\* and use \*\*meaningful commit messages\*\*.

PRs with messy commit history are difficult to review and won't be merged.

Use `rebase -i upstream/master` to curate your commit history and/or to

bring in latest changes from master (but avoid rebasing in the middle of a

code review).

\* Keep your PR up to date with upstream/master (if there are merge conflicts,

we can't really merge your change).

\* \*\*All tests need to be passing\*\* before your change can be merged. We

recommend you \*\*run tests locally\*\* (see below)

\* Exceptions to the rules can be made if there's a compelling reason for doing

so. That is - the rules are here to serve us, not the other way around, and

the rules need to be serving their intended purpose to be valuable.

## TCMalloc Committers

The current members of the TCMalloc engineering team are the only committers at

present.

## Community Guidelines

This project follows

[Google's Open Source Community Guidelines](https://opensource.google.com/conduct/).