# Contributing to Protocol Buffers

We welcome your contributions to protocol buffers. This doc describes the

process to contribute patches to protobuf and the general guidelines we

expect contributors to follow.

## Before You Start

We accept patches in the form of github pull requests. If you are new to

github, please read [How to create github pull requests](https://help.github.com/articles/about-pull-requests/)

first.

### Contributor License Agreements

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.

\* If you are an individual writing original source code and you're sure you

own the intellectual property, then you'll need to sign an [individual CLA](https://cla.developers.google.com/about/google-individual?csw=1).

\* If you work for a company that wants to allow you to contribute your work,

then you'll need to sign a [corporate CLA](https://cla.developers.google.com/about/google-corporate?csw=1).

### Coding Style

This project follows [Google’s Coding Style Guides](https://github.com/google/styleguide).

Before sending out your pull request, please familiarize yourself with the

corresponding style guides and make sure the proposed code change is style

conforming.

## Contributing Process

Most pull requests should go to the master branch and the change will be

included in the next major/minor version release (e.g., 3.6.0 release). If you

need to include a bug fix in a patch release (e.g., 3.5.2), make sure it’s

already merged to master, and then create a pull request cherry-picking the

commits from master branch to the release branch (e.g., branch 3.5.x).

For each pull request, a protobuf team member will be assigned to review the

pull request. For minor cleanups, the pull request may be merged right away

after an initial review. For larger changes, you will likely receive multiple

rounds of comments and it may take some time to complete. We will try to keep

our response time within 7-days but if you don’t get any response in a few

days, feel free to comment on the threads to get our attention. We also expect

you to respond to our comments within a reasonable amount of time. If we don’t

hear from you for 2 weeks or longer, we may close the pull request. You can

still send the pull request again once you have time to work on it.

Once a pull request is merged, we will take care of the rest and get it into

the final release.

## Pull Request Guidelines

\* 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.

We often receive PRs that are trying to fix several things at a time, but 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.

\* For speculative changes, consider opening an issue and discussing it first.

If you are suggesting a behavioral or API change, make sure you get explicit

support from a protobuf team member before sending us the pull request.

\* 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. 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 before creating your PR to catch breakages early on.

Ultimately, the green signal will be provided by our testing infrastructure.

The reviewer will help you if there are test failures that seem not related

to the change you are making.

## Reviewer Guidelines

\* Make sure that all tests are passing before approval.

\* Apply the "release notes: yes" label if the pull request's description should

be included in the next release (e.g., any new feature / bug fix).

Apply the "release notes: no" label if the pull request's description should

not be included in the next release (e.g., refactoring changes that does not

change behavior, integration from Google internal, updating tests, etc.).

\* Apply the appropriate language label (e.g., C++, Java, Python, etc.) to the

pull request. This will make it easier to identify which languages the pull

request affects, allowing us to better identify appropriate reviewer, create

a better release note, and make it easier to identify issues in the future.