How to contribute

We definitely welcome your patches and contributions to gRPC! Please read the gRPC organization's governance rules and contribution guidelines before proceeding.

If you are new to github, please start by reading Pull Request howto

Legal requirements

In order to protect both you and ourselves, you will need to sign the Contributor License Agreement.

Guidelines for Pull Requests

How to get your contributions merged smoothly and quickly.

- Create **small PRs** that are narrowly focused on **addressing a single concern**. We often times receive PRs that are trying to fix several things at a time, but 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.

- The grpc package should only depend on standard Go packages and a small number of exceptions. If your contribution introduces new dependencies which are NOT in the list, you need a discussion with gRPC-Go authors and consultants.

- For speculative changes, consider opening an issue and discussing it first. If you are suggesting a behavioral or API change, consider starting with a gRFC proposal.

- 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.

    - `make all` to test everything, OR
    - `make vet` to catch vet errors
    - `make test` to run the tests
    - `make testrace` to run tests in race mode

- Exceptions to the rules can be made if there's a compelling reason for doing so.