# Contributing

Greenplum is maintained by a core team of developers with commit rights to the main gpdb repository on GitHub. At the same time, we are very eager to receive contributions from anybody in the wider Greenplum community. This section covers all you need to know if you want to see your code or documentation changes be added to Greenplum and appear in the future releases.

## Getting started

Greenplum is developed on GitHub, and anybody wishing to contribute to it will have to have a GitHub account and be familiar with Git tools and workflow. It is also recommend that you follow the developer's mailing list since some of the contributions may generate more detailed discussions there.

Once you have your GitHub account, fork this repository so that you can have your private copy to start hacking on and to use as source of pull requests.

Anybody contributing to Greenplum has to be covered by either the Corporate or the Individual Contributor License Agreement. If you have not previously done so, please fill out and submit the Contributor License Agreement. Note that we do allow for really trivial changes to be contributed without a CLA if they fall under the rubric of obvious fixes. However, since our GitHub workflow checks for CLA by default you may find it easier to submit one instead of claiming an "obvious fix" exception.

## Licensing of Greenplum contributions

If the contribution you're submitting is original work, you can assume that Pivotal will release it as part of an overall Greenplum release available to the downstream consumers under the Apache License, Version 2.0. However, in addition to that, Pivotal may also decide to release it under a different license (such as PostgreSQL License to the upstream consumers that require it. A typical example here would be Pivotal upstreaming your contribution back to PostgreSQL community (which can be done either verbatim or your contribution being upstreamed as part of the larger changeset).

If the contribution you're submitting is NOT original work you have to indicate the name of the license and also make sure that it is similar in terms to the Apache License 2.0. Apache Software Foundation maintains a list of these licenses under Category A. In addition to that, you may be required to make proper attribution in the NOTICE file file similar to these examples.

Finally, keep in mind that it is NEVER a good idea to remove licensing headers from the work that is not your original one. Even if you are using parts of the file that originally had a licensing header at the top you should err on the side of preserving it. As always, if you are not quite sure about the licensing implications of your contributions, feel free to reach out to us on the developer mailing list.

## Coding guidelines

Your chances of getting feedback and seeing your code merged into the project greatly depend on how granular your changes are. If you happen to have a bigger change in mind, we highly recommend engaging on the developer's mailing list first and sharing your proposal with us before you spend a lot of time writing code. Even when your proposal gets validated by the community, we still recommend doing the actual work as a series of small, self-contained commits. This makes the reviewer's job much easier and increases the timeliness of feedback.

When it comes to C and C++ parts of Greenplum, we try to follow PostgreSQL Coding Conventions. In addition to that:

For C and perl code, please run pgindent if necessary as specified in [README.gpdb](/src/tools/pgindent/README.gpdb).

All Python code must pass Pylint.

All Go code must be formatted according to gofmt.

We recommend using `git diff --color` when reviewing your changes so that you don't have any spurious whitespace issues in the code that you submit.

All new functionality that is contributed to Greenplum should be covered by regression tests that are contributed alongside it. If you are uncertain on how to test or document your work, please raise the question on the gpdb-dev mailing list and the developer community will do its best to help you.

At the very minimum you should always be running make installcheck-world to make sure that you're not breaking anything.

## Changes applicable to upstream PostgreSQL

If the change you're working on touches functionality that is common between PostgreSQL and Greenplum, you may be asked to forward-port it to PostgreSQL. This is not only so that we keep reducing the delta between the two projects, but also so that any change that is relevant to PostgreSQL can benefit from a much broader review of the upstream PostgreSQL community. In general, it is a good idea to keep both code bases handy so you can be sure whether your changes may need to be forward-ported.

## Submission timing

To improve the odds of the right discussion of your patch or idea happening, pay attention to what the community work cycle is. For example, if you send in a brand new idea in the beta phase of a release, we may defer review or target its inclusion for a later version. Feel free to ask on the mailing list to learn more about the Greenplum release policy and timing.

## Patch submission

Once you are ready to share your work with the Greenplum core team and the rest of the Greenplum community, you should push all the commits to a branch in your own repository forked from the official Greenplum and send us a pull request.

For now, we require all pull requests to be submitted against the main master branch, but over time, once there are many supported open source releases of Greenplum in the wild, you may decide to submit your pull requests against an active release branch if the change is only applicable to a given release.

## Validation checks and CI

Once you submit your pull request, you will immediately see a number of validation checks performed by our automated CI pipelines. There also will be a CLA check telling you whether your CLA was recognized. If any of these checks fails, you will need to update your pull request to take care of the issue. Pull requests with failed validation checks are very unlikely to receive any further peer review from the community members.

Keep in mind that the most common reason for a failed CLA check is a mismatch between an email on file and an email recorded in the commits submitted as part of the pull request.

If you cannot figure out why a certain validation check failed, feel free to ask on the developer's mailing list, but make sure to include a direct link to a pull request in your email.

## Patch review

A submitted pull request with passing validation checks is assumed to be available for peer review. Peer review is the process that ensures that contributions to Greenplum are of high quality and align well with the road map and community expectations. Every member of the Greenplum community is encouraged to review pull requests and provide feedback. Since you don't have to be a core team member to be able to do that, we recommend following a stream of pull reviews to anybody who's interested in becoming a long-term contributor to Greenplum. As Linus would say "given enough eyeballs, all bugs are shallow".

One outcome of the peer review could be a consensus that you need to modify your pull request in certain ways. GitHub allows you to push additional commits into a branch from which a pull request was sent. Those additional commits will be then visible to all of the reviewers.

A peer review converges when it receives at least one +1 and no -1s votes from the participants. At that point you should expect one of the core team members to pull your changes into the project.

Greenplum prides itself on being a collaborative, consensus-driven environment. We do not believe in vetoes and any -1 vote casted as part of the peer review has to have a detailed technical explanation of what's wrong with the change. Should a strong disagreement arise it may be advisable to take the matter onto the mailing list since it allows for a more natural flow of the conversation.

At any time during the patch review, you may experience delays based on the availability of reviewers and core team members. Please be patient. That being said, don't get discouraged either. If you're not getting expected feedback for a few days add a comment asking for updates on the pull request itself or send an email to the mailing list.

## Direct commits to the repository

On occasion you will see core team members committing directly to the repository without going through the pull request workflow. This is reserved for small changes only and the rule of thumb we use is this: if the change touches any functionality that may result in a test failure, then it has to go through a pull request workflow. If, on the other hand, the change is in the non-functional part of the code base (such as fixing a typo inside of a comment block) core team members can decide to just commit to the repository directly.