# How to Contribute #

Thank you for your interest in contributing to the Apache Thrift project! Information on why and how to contribute is available on the Apache Software Foundation (ASF) web site. In particular, we recommend the following to become acquainted with Apache Contributions:

\* [Contributors Tech Guide](http://www.apache.org/dev/contributors)

\* [Get involved!](http://www.apache.org/foundation/getinvolved.html)

\* [Legal aspects on Submission of Contributions (Patches)](http://www.apache.org/licenses/LICENSE-2.0.html#contributions)

## GitHub pull requests ##

This is the preferred method of submitting changes. When you submit a pull request through github,

it activates the continuous integration (CI) build systems at Appveyor and Travis to build your changesxi

on a variety of Linux and Windows configurations and run all the test suites. Follow these requirements

for a successful pull request:

1. All significant changes require an [Apache Jira THRIFT Issue](http://issues.apache.org/jira/browse/THRIFT) ticket. Trivial changes such as fixing a typo or a compiler warning do not.

1. All pull requests should contain a single commit per issue, or we will ask you to squash it.

1. The pull request title must begin with the Jira THRIFT ticket identifier if it has an associated ticket, for example:

THRIFT-9999: an example pull request title

1. Commit messages must follow this pattern for code changes (deviations will not be merged):

THRIFT-9999: [summary of fix, one line if possible]

Client: [language(s) affected, comma separated, for example: "cpp,erl,perl"]

Instructions:

1. Create a fork in your GitHub account of http://github.com/apache/thrift

1. Clone the fork to your development system.

1. Create a branch for your changes (best practice is issue as branch name, e.g. THRIFT-9999).

1. Modify the source to include the improvement/bugfix, and:

\* Remember to provide \*tests\* for all submitted changes!

\* Use test-driven development (TDD): add a test that will isolate the bug \*before\* applying the change that fixes it.

\* Verify that you follow [Thrift Coding Standards](/docs/coding\_standards) (you can run 'make style', which ensures proper format for some languages).

\* [\*optional\*] Verify that your change works on other platforms by adding a GitHub service hook to [Travis CI](http://docs.travis-ci.com/user/getting-started/#Step-one%3A-Sign-in) and [AppVeyor](http://www.appveyor.com/docs). You can use this technique to run the Thrift CI jobs in your account to check your changes before they are made public. Every GitHub pull request into Thrift will run the full CI build and test suite on your changes.

1. Squash your changes to a single commit. This maintains clean change history.

1. Commit and push changes to your branch (please use issue name and description as commit title, e.g. "THRIFT-9999: make it perfect"), with the affected languages on the next line of the description.

1. Use GitHub to create a pull request going from your branch to apache:master. Ensure that the Jira ticket number is at the beginning of the title of your pull request, same as the commit title.

1. Wait for other contributors or committers to review your new addition, and for a CI build to complete.

1. Wait for a committer to commit your patch. You can nudge the committers if necessary by sending a message to the [Apache Thrift mailing list](https://thrift.apache.org/mailing).

## If you want to build the project locally ##

For Windows systems, see our detailed instructions on the [CMake README](/build/cmake/README.md).

For Windows Native C++ builds, see our detailed instructions on the [WinCPP README](/build/wincpp/README.md).

For unix systems, see our detailed instructions on the [Docker README](/build/docker/README.md).

## If you want to review open issues... ##

1. Review the [GitHub Pull Request Backlog](https://github.com/apache/thrift/pulls). Code reviews are open to all.

2. Review the [Jira issue tracker](http://issues.apache.org/jira/browse/THRIFT). You can search for tickets relating to languages you are interested in or currently using with thrift, for example a Jira search (Issues -> Search For Issues) query of ``project = THRIFT AND component in ("Erlang - Library") and status not in (resolved, closed)`` will locate all the open Erlang Library issues.

## If you discovered a defect... ##

1. Check to see if the issue is already in the [Jira issue tracker](http://issues.apache.org/jira/browse/THRIFT).

1. If not, create a ticket describing the change you're proposing in the Jira issue tracker.

1. Contribute your code changes using the GitHub pull request method:

## Contributing via Patch ##

To create a patch from changes in your local directory:

git diff > ../THRIFT-NNNN.patch

then wait for contributors or committers to review your changes, and then for a committer to apply your patch. This is not the preferred way to submit changes and incurs additional overhead for committers who must then create a pull request for you.

## GitHub recipes for Pull Requests ##

Sometimes commmitters may ask you to take actions in your pull requests. Here are some recipes that will help you accomplish those requests. These examples assume you are working on Jira issue THRIFT-9999. You should also be familiar with the [upstream](https://help.github.com/articles/syncing-a-fork/) repository concept.

### Squash your changes ###

If you have not submitted a pull request yet, or if you have not yet rebased your existing pull request, you can squash all your commits down to a single commit. This makes life easier for the committers. If your pull request on GitHub has more than one commit, you should do this.

1. Use the command ``git log`` to identify how many commits you made since you began.

2. Use the command ``git rebase -i HEAD~N`` where N is the number of commits.

3. Leave "pull" in the first line.

4. Change all other lines from "pull" to "fixup".

5. All your changes are now in a single commit.

If you already have a pull request outstanding, you will need to do a "force push" to overwrite it since you changed your commit history:

git push -u origin THRIFT-9999 --force

A more detailed walkthrough of a squash can be found at [Git Ready](http://gitready.com/advanced/2009/02/10/squashing-commits-with-rebase.html).

### Rebase your pull request ###

If your pull request has a conflict with master, it needs to be rebased:

git checkout THRIFT-9999

git rebase upstream master

(resolve any conflicts, make sure it builds)

git push -u origin THRIFT-9999 --force

### Fix a bad merge ###

If your pull request contains commits that are not yours, then you should use the following technique to fix the bad merge in your branch:

git checkout master

git pull upstream master

git checkout -b THRIFT-9999-take-2

git cherry-pick ...

(pick only your commits from your original pull request in ascending chronological order)

squash your changes to a single commit if there is more than one (see above)

git push -u origin THRIFT-9999-take-2:THRIFT-9999

This procedure will apply only your commits in order to the current master, then you will squash them to a single commit, and then you force push your local THRIFT-9999-take-2 into remote THRIFT-9999 which represents your pull request, replacing all the commits with the new one.