# Contributing

This document describes how you can contribute to Telegram Desktop. Please read it carefully.

\*\*Table of Contents\*\*

\* [What contributions are accepted](#what-contributions-are-accepted)

\* [Build instructions](#build-instructions)

\* [Pull upstream changes into your fork regularly](#pull-upstream-changes-into-your-fork-regularly)

\* [How to get your pull request accepted](#how-to-get-your-pull-request-accepted)

\* [Keep your pull requests limited to a single issue](#keep-your-pull-requests-limited-to-a-single-issue)

\* [Squash your commits to a single commit](#squash-your-commits-to-a-single-commit)

\* [Don't mix code changes with whitespace cleanup](#dont-mix-code-changes-with-whitespace-cleanup)

\* [Keep your code simple!](#keep-your-code-simple)

\* [Test your changes!](#test-your-changes)

\* [Write a good commit message](#write-a-good-commit-message)

## What contributions are accepted

We highly appreciate your contributions in the matter of fixing bugs and optimizing the Telegram Desktop source code and its documentation. In case of fixing the existing user experience please push to your fork and [submit a pull request][pr].

Wait for us. We try to review your pull requests as fast as possible.

If we find issues with your pull request, we may suggest some changes and improvements.

Unfortunately we \*\*do not merge\*\* any pull requests that have new feature implementations, translations to new languages and those which introduce any new user interface elements.

If you have a translations-related contribution, check out [Translations platform][translate].

Telegram Desktop is not a standalone application but a part of [Telegram project][telegram], so all the decisions about the features, languages, user experience, user interface and the design are made inside Telegram team, often according to some roadmap which is not public.

## Build instructions

See the [README.md][build\_instructions] for details on the various build

environments.

## Pull upstream changes into your fork regularly

Telegram Desktop is advancing quickly. It is therefore critical that you pull upstream changes into your fork on a regular basis. Nothing is worse than putting in a days of hard work into a pull request only to have it rejected because it has diverged too far from upstream.

To pull in upstream changes:

git remote add upstream https://github.com/telegramdesktop/tdesktop.git

git fetch upstream master

Check the log to be sure that you actually want the changes, before merging:

git log upstream/master

Then rebase your changes on the latest commits in the `master` branch:

git rebase upstream/master

After that, you have to force push your commits:

git push --force

For more info, see [GitHub Help][help\_fork\_repo].

## How to get your pull request accepted

We want to improve Telegram Desktop with your contributions. But we also want to provide a stable experience for our users and the community. Follow these rules and you should succeed without a problem!

### Keep your pull requests limited to a single issue

Pull requests should be as small/atomic as possible. Large, wide-sweeping changes in a pull request will be \*\*rejected\*\*, with comments to isolate the specific code in your pull request. Some examples:

\* If you are making spelling corrections in the docs, don't modify other files.

\* If you are adding new functions don't '\*cleanup\*' unrelated functions. That cleanup belongs in another pull request.

#### Squash your commits to a single commit

To keep the history of the project clean, you should make one commit per pull request.

If you already have multiple commits, you can add the commits together (squash them) with the following commands in Git Bash:

1. Open `Git Bash` (or `Git Shell`)

2. Enter following command to squash the recent {N} commits: `git reset --soft HEAD~{N} && git commit` (replace `{N}` with the number of commits you want to squash)

3. Press <kbd>i</kbd> to get into Insert-mode

4. Enter the commit message of the new commit

5. After adding the message, press <kbd>ESC</kbd> to get out of the Insert-mode

6. Write `:wq` and press <kbd>Enter</kbd> to save the new message or write `:q!` to discard your changes

7. Enter `git push --force` to push the new commit to the remote repository

For example, if you want to squash the last 5 commits, use `git reset --soft HEAD~5 && git commit`

### Don't mix code changes with whitespace cleanup

If you change two lines of code and correct 200 lines of whitespace issues in a file the diff on that pull request is functionally unreadable and will be \*\*rejected\*\*. Whitespace cleanups need to be in their own pull request.

### Keep your code simple!

Please keep your code as clean and straightforward as possible.

Furthermore, the pixel shortage is over. We want to see:

\* `opacity` instead of `o`

\* `placeholder` instead of `ph`

\* `myFunctionThatDoesThings()` instead of `mftdt()`

### Test your changes!

Before you submit a pull request, please test your changes. Verify that Telegram Desktop still works and your changes don't cause other issue or crashes.

### Write a good commit message

\* Explain why you make the changes. [More infos about a good commit message.][commit\_message]

\* If you fix an issue with your commit, please close the issue by [adding one of the keywords and the issue number][closing-issues-via-commit-messages] to your commit message.

For example: `Fix #545`

[//]: # (LINKS)

[telegram]: https://telegram.org/

[help\_fork\_repo]: https://help.github.com/articles/fork-a-repo/

[help\_change\_commit\_message]: https://help.github.com/articles/changing-a-commit-message/

[commit\_message]: http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html

[pr]: https://github.com/telegramdesktop/tdesktop/compare

[build\_instructions]: https://github.com/telegramdesktop/tdesktop/blob/master/README.md#build-instructions

[closing-issues-via-commit-messages]: https://help.github.com/articles/closing-issues-via-commit-messages/

[translate]: https://translations.telegram.org