## Contributing [documentation](https://github.com/git-tfs/git-tfs/tree/master/doc)

We'll try to review and merge documentation changes quickly.

If you see something easy to change, you can probably even make the change in your browser!

We are a small team and each of us have a different tfs history with different edge cases... Each fix or new use case supported will be much appreciated and will permit to build a tool usable for a maximum of people. Do not hesitate to contribute. Every little help will be appreciated!!

## Contributing code

0. Open an [issue](https://github.com/git-tfs/git-tfs/issues) if you want to add a new feature. That way, we could give you some advices, tell you where to look for in the source code,...

1. In your git-tfs clone, run `git config core.autocrlf true` so that all the line endings are unix line endings when you commit.

2. \*\*Read the source.\*\* It's hopefully not that intimidating.

3. \*\*Check for issues.\*\* There are a few issues that are low-hanging fruit. Feel free to pick them.

4. \*\*Ask questions.\*\* Feel free to drop me a line, or ask a question over at the [google group](http://groups.google.com/group/git-tfs-dev/), or join `#git-tfs` on FreeNode.

5. Verify that your editor is configured to use 4 spaces instead of tabs. You could even install the [EditorConfig Extension for VisualStudio](http://visualstudiogallery.msdn.microsoft.com/c8bccfe2-650c-4b42-bc5c-845e21f96328) ([such plugins exists for other editors](http://editorconfig.org/#download)), and the good space configuration will be set automatically when opening the git-tfs solution.

6. Run the build task `.\build.ps1 -Target "FormatCode"` before committing to keep code formatting consistent (and make pull request review easier and quicker!)

## Pull Requests

Here are some tips on creating a pull request:

1. Write that awesome code. :sparkles:

2. Make sure the existing unit tests don't break. We try to keep the unit tests

[easy to run](https://github.com/git-tfs/git-tfs/blob/master/doc/running-the-unit-tests.md).

3. We like new unit tests. If you can unit test your code, do so.

One of the pain points of the current git-tfs codebase is that parts of it are very difficult to unit test.

It's slowly getting less painful to unit test. One thing to try is to write an integration test

that runs git-tfs with the VsFake driver, similar to how the clone tests are written.

4. If you modify code in any of the client adapters (GitTfs.Vs\*), please try it out with as many versions of the TFS client libraries as you can.

When you submit the pull request, include a note about which versions you have tried to compile with, and which ones you have tested with.

5. Include [documentation](https://github.com/git-tfs/git-tfs/tree/master/doc) for externally-visible changes.

6. Update the [release notes for the next version](https://github.com/git-tfs/git-tfs/tree/master/doc/release-notes).