# Table of contents

Contents

[Table of contents 1](#_Toc13494296)

[Introduction 1](#_Toc13494297)

[Basic operations 2](#_Toc13494298)

[Add File 2](#_Toc13494299)

[Commit File 5](#_Toc13494300)

[Push (no conflicts) 8](#_Toc13494301)

[Pull (no conflicts) 12](#_Toc13494302)

[Appendix A 14](#_Toc13494303)

[Appendix B 15](#_Toc13494304)

[Troubleshooting 15](#_Toc13494305)

[Advanced Troubleshooting 17](#_Toc13494306)

[Appendix C 19](#_Toc13494307)

# Introduction

This is how-to style document on how to use Git.

His dedicated audience is new Git user.

In general it has Windows-oriented, but it should be pretty straight-forward to apply it on another platforms.

If you have some SVN knowledge, please see Appendix A.

For troubleshooting see Appendix B. It uses command line command to resolve some troubles that you can encounter with.

There some choices and assumptions that was made deliberately and they not discussed in the document. There is no assumption, however, on your remote repository, it can be any Git provider such as GitHub, GitLab, etc.

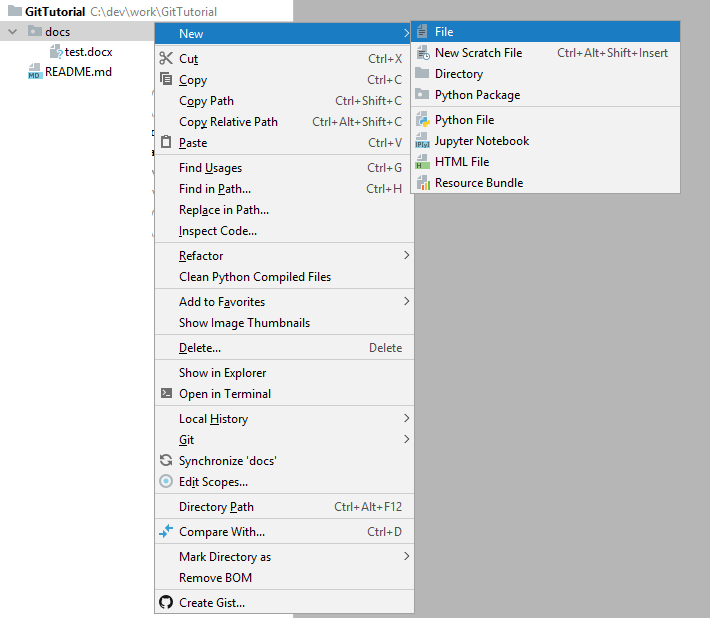
PyCharm is used as git client through this document. See Appendix C for details. All examples are deliberately provided on PyCharm.

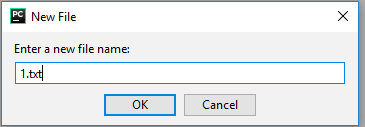
TBD

# Basic operations

## Add File

* If you are adding new file through Pycharm, it will be automatically added to Git (you will be able to commit it without any addition actions).





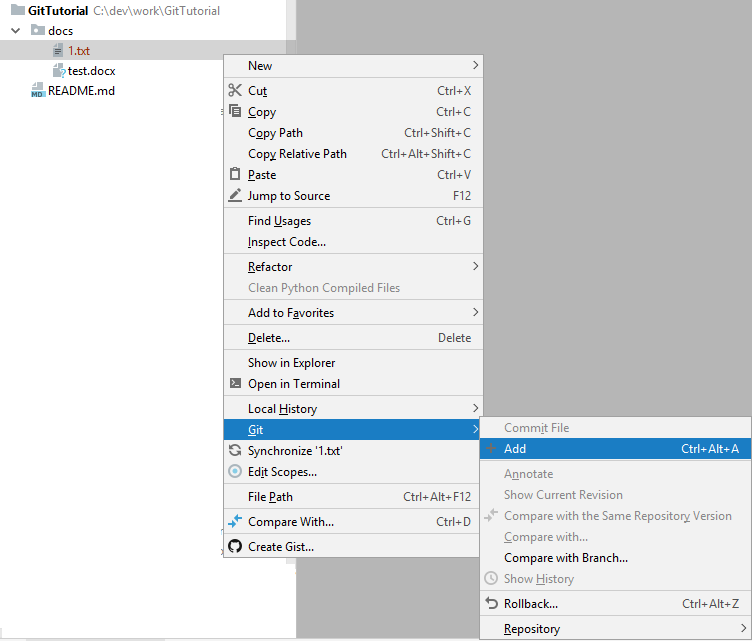
In such a case you just do nothing.

Note: If you copy/move file with Pycharm it will be also automatically added to Git and will be available for commit alter.

This are typical cases of working with files. Pycharm will “git add” it for you.

* Occasionally, you will want to add files outside Pycharm.
* For instance, you will copy this file from another location in Windows.
* Or you’re creating new file in CMD\Terminal.
* Another use-case, when Pycharm is not added this file. See in commit section, how do you know.

Right-click on file that you want to add (you can select directory/project instead) -> Git -> Add.



Note: ”Commit file” is not available (it is greyed out).

Note: It is safe to add file multiple times; it will do nothing (after first time).

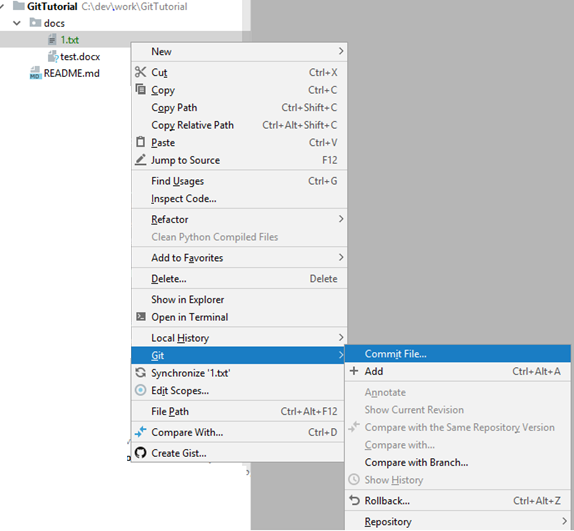
## Commit File

In order to commit file, it should be added first.

Note: In Pycharm it will be usually added (to index) automatically.

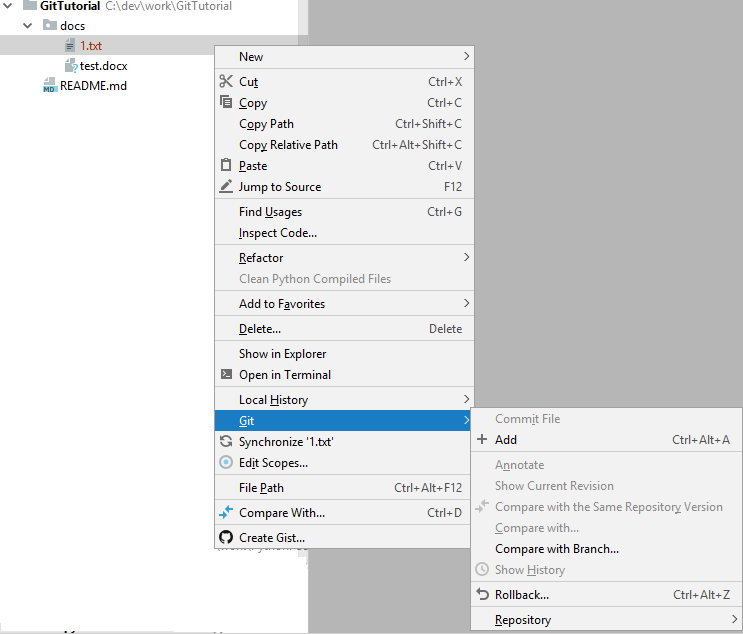
In order to commit file, you can

right-click on file that you want to commit (you can select directory/project instead) -> Git -> Commit.



* How can you check that file you are about to commit was added (to index)? One way is simply try to commit:

right-click on file that you want to commit (you can select directory/project instead) -> Git -> if commit is not available (is greyed out) than you should add the file (directory/project) first.



## Push (no conflicts)

After you commit your changes, you want them to be available to everybody. This is what “git push” does. Before, “git push”, all your commits are “visible” only for you.

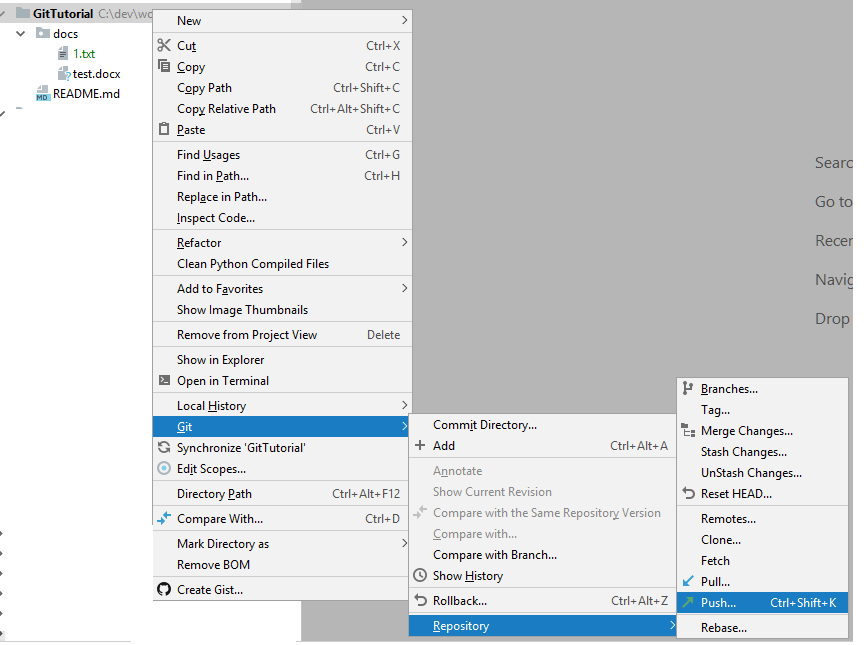
Note: Typically, you should do “git pull” first.

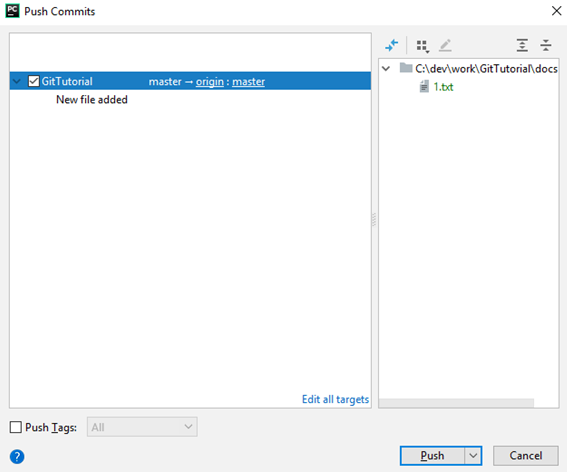
Note: Anyway, in this section I’m assuming that there are no conflicts in “git push”. Conflict resolution is walked through in “Merge” section.

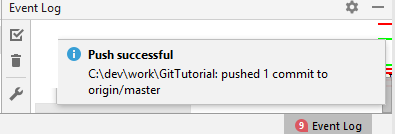
Right-click project -> Git -> Repository ->Push

In the popup window check the project you want to Push. Click on “Push” button.

You should see in the right left corner in the floating window “Push successful”.





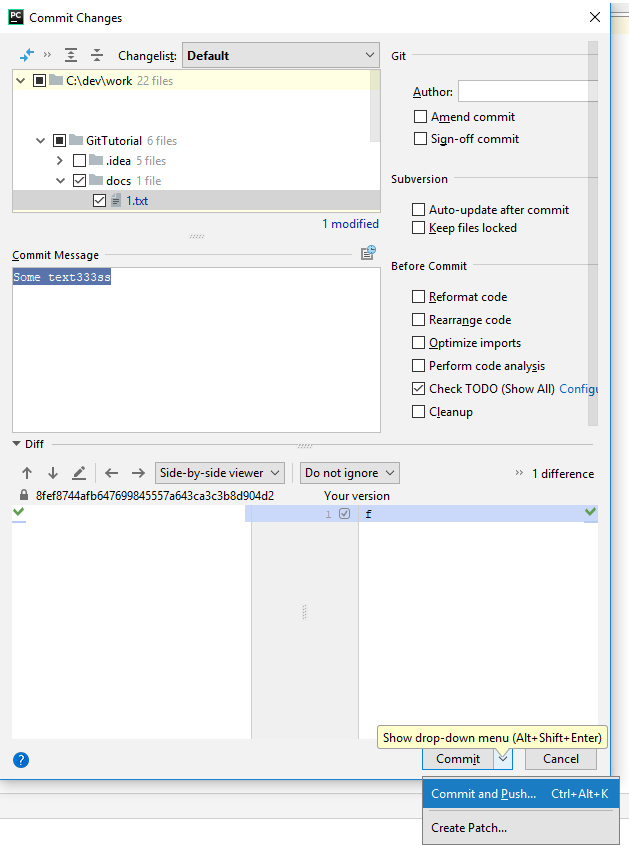


In the push commits window, you see all the Git project. In every Git project you see message of every commit that you’re about to push. On the right you can see from what files each commits consists of.

Note: If push failed due the conflict it is better to abort push. See “merge” section for details.

Note: You can also commit&push together. This is not atomic operation, it will be translated to “git commit” and “git push” (last one can fail if there is conflict).

In the “commit changes” window near the commit button, you can click on “v” and you will see sub-menu in which you can chose commiy&push.



## Pull (no conflicts)

This operation takes changes that somebody else did and integrate them to your project.

Note: In this section I’m assuming that there are no conflicts in “git pull”. Conflict resolution is walked through in “Merge” section.

Note: It is advised to do “git pull” once in a while. See “merge” section for more details.

# Appendix A

This section contains **rough** “translation” of familiar terms from SVN to Git.

This may be useful if you know SVN and start to use Git. Note, that in any place in the document, except introduction, there is no assumption on SVN knowledge.

* Trunk in SVN is roughly corresponds to master on Git.
* Update (without conflicts) in SVN – to pull.
* Commit (without conflicts) in SVN to (add file)+commit+push.
* Checkout of entire repository in SVN is clone (there is also another way).
* Branch on SVN is chain of commits. Merge of 2 branches is simply created new commit that copies changes from another branch. This is “hard branches”.  
  Branches in Git are not “hard branches”, *they are only pointers on commits*.
* Commit in Git can have more than one parent.

More advanced (you can skip it, especially in your first read):

* Commit history saved in different way than in SVN. I will point only on 1 of them:
* On SVN history is stored on your central repository. Each operation on it, requires call to it. On SVN each local repository has full copy of the history. Each operation on it are local.
* Update (with conflicts) in SVN – pull + resolve/merge conflicts+commit. Commit is used to conclude merge.
* Commit (with conflicts) in SVN (typically due SVN commit which fails as “out of date”, svn update resolve the issue, svn update typically it works without conflicts, you can commit after this) – (commit can’t case conflicts, you are working locally) git push + resolve conflicts.

# Appendix B

## Troubleshooting

* If you close merge resolution windows and when you want to commit your merge changes, you don’t see any file that change

Type in the terminal/cmd

git commit

See next section for more details.

* If you successfully resolve all merge conflicts, but you receive error message “You have not concluded your merge (MERGE\_HEAD exists)”

Type in the terminal/cmd

git commit

or to abort merge

git merge --abort

Note: If you should absolutely sure that all conflicts where resolved, you can try also.

del /q /s /f .git\MERGE\*

Or you can type

rm -fr .git/MERGE\*

See <https://stackoverflow.com/a/21829654>

* If Pull failed, you should merge you changes first and then

Type in the terminal/cmd

git commit

Close opened editor, ignore it.

You should have additional commit “Merge branch ‘master’ of”.

And push your changes in regular way.

Note: See also Troubleshouthing version above for another option. In this option you have auto-generate message for merge commit.

## Advanced Troubleshooting

This subsection some dangerous command that should be used only if you know what are you doing.

* If Pull failed, you should merge you changes first and then

Make some little change to any file, commit your changes (you should have additional commit “Merge branch ‘master’ of”), commit&push your changes and undo you change.

* If you want to move remote branch (for example origin/master), move your local branch to the desired point (Reset current branch to Here…) and then type

git push --force origin master

Note: This operation should be extremely rare. Do this with care.

Note: Change master to your local current branch name if needed.

Note: If you do this on long lived branch (such as master) you should say to everybody to pull from this repository (to reflect your change in their local repository)

* Update git. See <https://stackoverflow.com/a/48924212>

git update-git-for-windows

* If you have following error message:

fatal: unable to access 'https://somepath/ProjectName.git/': SSL certificate problem: self signed certificate

You have two options (see <https://stackoverflow.com/a/9008394> ):

1. Open cmd/terminal and type

git config http.sslVerify false

This will disable SSL certificate checking the repository in which you type this command only.

1. Open cmd/terminal and type

git config --global http.sslVerify false

This will disable SSL certificate checking globally.

Note: Seems the --global option IS needed when a repo is NOT yet checked out (can't set options for a repo that doesn't exist yet locally). One can always turn it back on after.

# Appendix C

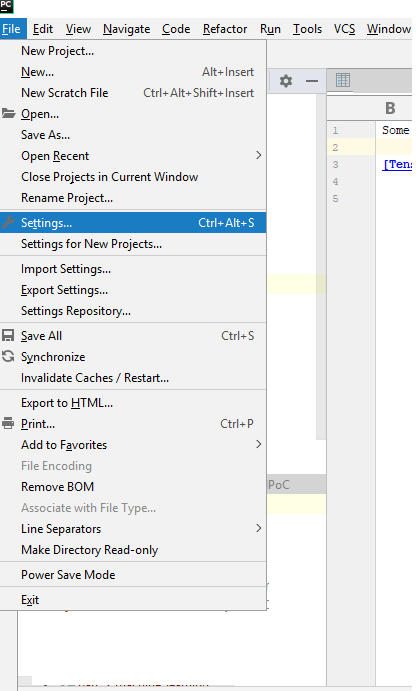
This appendix contains step to reproduce my local environment.

1. Install git client.

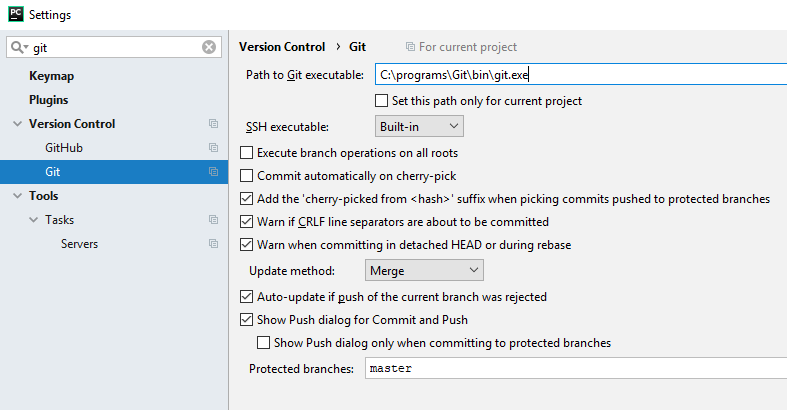
* Go to <https://git-scm.com/downloads> and download appropriate one. I use 2.22.0.windows.1 (latest for the day this document was written).
* Install it.
* You can see Appendix B->Advance Troubleshooting->Update git section if you need to update it.
* Add path to git.exe to your Path environment variable (or it’s equivalent on another OS).

1. PyCharm 2018.2.4 (Communnity edition) is used as Git UI in this document.

* See <https://www.jetbrains.com/pycharm/download/previous.html>
* In Pycharn open Settings. File->Settings



Ensure that you have following settings:



Note:

* Path to Git executable may differ on your machine.
* Ensure that “Update method“ is Merge
* Ensure that check-box “Warn when committing in detached HEAD or during rebase is checked.
* (you can also do git init, and git remote add origin, create local master, and git push -u origin master)