Install Git for Windows

<https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>

From the book <https://git-scm.com/book/en/v2/>

**BASICS**

check the list of settings

git config --list

one time setup user and email

git config --global user.name "John Doe"

git config --global user.email "johndoe@example.com"

remove variables from –global

git config --global --unset usern.name

check value of a variable (in this case the default git editor)

git config core.editor

get help (long and short)

git help config

git config –h

clone repository

git clone https://github.com/caxtla/test

or

git clone <https://github.com/caxtla/test> foldername

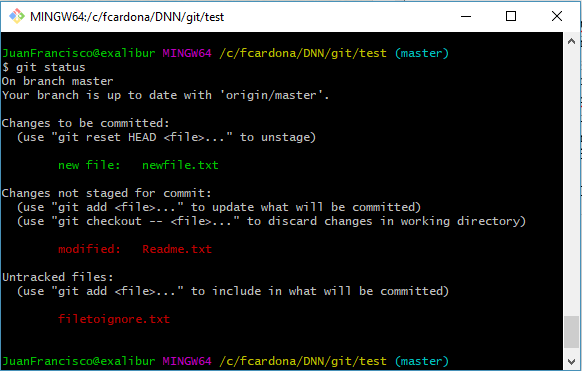


Get status of the folder

git status

New files added will be shown as Untracked (red color) and they will not be committed until you add them with add

git add newfile.txt



After added the files will be seen as “Changes to be committed” (green color). Git add dirpath will add the dir and all files inside recursively.

Modified files will be shown as “Changes not staged for commit”. To mark as staged files use also git add file (git add = include in the next commit). If after add you do another change then the file will be shown in both staged and not staged, so another add is required to commit the last change.

Short version of status

git status -s

To ignore files create a file .gitignore, placed in the root folder and it applies recursively to all subfolders, but you can have also files inside subfolders to customize each folder

For examples check <https://github.com/github/gitignore>

To check changes in files of what has been changed but not stated use

git diff

or to check staged changes

git diff --staged

To commit changes use:

git commit

and the default editor will be open to add comment. Or Use the following to add comment:

git commit -m "Commit message"

To skip staging (to avoid get add file for each tracked file changed), use -a:

git commit -a -m "Committ without add"

To remove a file use rm

git rm filetoremove.txt

If you remove the file only from the file system it will be seen in git as removed but it will not be removed from the repository, it requires to use the git rm command. The git rm also deletes the file from the file system.

If you added a file to the repository and then you want to delete it from the repository but keep it in your file system then use the command

git rm --cached keepinfilesystem.txt

To rename files:

git mv file.txt newname.txt

To see a history of commits use:

git log

Also helpful is –p (show changes) and -2 (show last 2 changes)

git log -p -2

To override the last commit with the current snapshot use

git commit --amend

To unstage a file (it was git add but no yet git commit) then use:

git reset HEAD file.txt

To recover a file which has been changed:

git checkout -- file.txt

To show remote repositories use:

git remote -v

Or just git remote for a shorter version

Instead of cloning a repository you can add manually a repository with a name using (make sure you are in folder with a git repository using “git init”):

git remote add fctest https://github.com/caxtla/test

The format is git remote add <name> <url>. Later you can use <name> instead of <url>. When you do a git clone, the name by default is “origin”

To get data the repository use git fetch

git fetch fctest

git fetch <name> (or <url>). This command will download the files but it will not change your local files, you need to merge them.

Instead of fetch you can use pull to download the changes and merge them with your current branch, so the files are visible in your file system. To merge use:

git merge origin

or

git merge origin/master

There is a shortcut when you have a tracking-branch, using the git pull command. But first we need to create a remote tracking-branch, using ‘git checkout –b <newbranch>’ <remote>/<remotebranch>

git checkout -b copytest fctest/master

After this doing git pull will download all files

To push your changes use git push <remote> <branch>

git push origin master

To see more information about the <remotes> use ‘git remote show <remote>’

git remote show origin

To rename

git remote remove oldname newname

And to delete

git remote remove remotename

**BRANCHES**

<https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>

In git each commits points to a tree file which points to all files changes (snapshot) and it also points to its previous commit



The default branch in git is “master”. A branch is a pointer to a commit and multiple branches can point to the same commit. HEAD points always to your current branch. To find out where it is pointing use:

git log --oneline --decorate

Usually HEAD points to master

To create a branch use git branch <branchname>

git branch testing

HEAD is still pointing to master

To select a different branch use git checkout <branchname>

git checkout testing

Any change here will be added to the testing branch, and HEAD points to testing. After commit if you switch back to master with ‘git checkout master’ you will not see changes from testing branch. Then after commit to master the following scenario is shown:



You can create a branch and checkout at the same time with -b

git checkout –b newbranch

This is a shortcut of git branch and git checkout. You can switch to a different branch only if you have no changes in your current branch

Check all history with:

git log --oneline --decorate --graph --all

Git control of changes

<https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>



To merge a fast-forward branch into the master:

git checkout master

git merge hotfix

If <hotfix> is the next commit then master will point to the commit (fast-forward)

Once the branch is merged in master it can be deleted with –d:

git branch -d hotfix

Basic merging



git checkout master

git merge iss53

Git compares 3 snapshots: master, iss53 and the previous common to both to do this merge and then creates a new branch with all changes



At this point the branch iss53 can be deleted as it was already merged in the master. If there are conflicts during a merge then it will not work. Use git status to see the conflicts. Conflicts can be fixed manually (use git add <file> to confirm merge) or the mergetool can be used:

git mergetool

To show the list of branches:

git mergetool

The current branch will have a \* at the beginning

To find out the last commit for each branch:

git branch -v

To find out branches which have been merged or not merged with the current branch use:

git branch –merged

git branch –no-merged

git branch –no-merged anyotherbranch

The last option is to check state of a different branch without checking it out

Centralized workflow



1. /etc/gitconfig file: Contains values applied to every user on the system and all their repositories. If you pass the option --system to git config, it reads and writes from this file specifically. (Because this is a system configuration file, you would need administrative or superuser privilege to make changes to it.)
2. ~/.gitconfig or ~/.config/git/config file: Values specific personally to you, the user. You can make Git read and write to this file specifically by passing the --global option, and this affects all of the repositories you work with on your system.
3. config file in the Git directory (that is, .git/config) of whatever repository you’re currently using: Specific to that single repository. You can force Git to read from and write to this file with the --local option, but that is in fact the default. (Unsurprisingly, you need to be located somewhere in a Git repository for this option to work properly.)