Git - Commands

1. You can view all of your settings and where they are coming from using:

**$ git config --list --show-origin**

1. Set your user name and email address globally:

**$ git config --global user.name "<username>"**

**$ git config --global user.email <email>**

1. Set your user name and email address for a specific project:

**$ git config user.name "<username>"**

**$ git config user.email <email>**

1. Set up text editor for git:

**$ git config --global core.editor <editor-name>**

1. Checking configuration settings:

**$ git config --list**

1. You can also check what Git thinks a specific key’s value is by typing:

**$ git config <key>**

1. If you ever need help while using Git, there are three equivalent ways to get the comprehensive manual page (manpage) help for any of the Git commands:

**$ git help <verb>**

**$ git <verb> --help**

**$ man git-<verb>**

1. Creating a git repository:

**$ git init**

1. Cloning an existing repository:

**$ git clone <url>**

**$ git clone <url> <filename>**

1. Checking status of files:

**$ git status**

1. Tracking new files:

**$ git add <filename>**

**$ git add . ~track all files**

1. Short status:

**$ git status -s**

**$ git status --short**

1. See changes between modified & unstaged file:

**$ git diff**

1. See changes between staged & last commit:

**$ git diff –-staged**

**$ git diff --cached**

1. See difference in an external diff showing program:

**$ git difftool --tool=<tool>**

1. To list the available diff showing programs:

**$ git difftool --tool-help**

1. Simple commit:

**$ git commit**

1. Put the diffs in the commit:

**$ git commit -v**

1. Commit with message inline:

**$ git commit -m “<message>”**

1. Skip the staging area and direct commit:

**$ git commit -a -m “<message>”**

1. Removing files:

**$ git rm <filename>**

**$ git rm <directories , file-globe-patterns>**

1. Force removal:

**$ git rm -f <filename>**

1. Untrack a file:

**$ git rm –-cached <filename>**

1. Renaming a file:

**$ git mv <old-filename> <new-filename>**

1. Viewing commit history:

**$ git log**

1. History with difference:

**$ git log -p**

**$ git log --patch**

1. History with difference and limit:

**$ git log -p -<n> ~ n: positive integer**

1. History with stats & shortstats:

**$ git log --stat**

**$ git log --shortstat**

1. Formatted History:

**$ git log --pretty=<option>**

**~ option: oneline, short, full, fuller, format**

**$ git log --pretty=format:”format-options”**

1. Formatted History with graph:

**$ git log --pretty=<option> --graph**

1. Show the list of files modified after the commit information:

**$ git log --name-only**

1. Show the list of files affected with added/ modified / deleted information as well:

**$ git log --name-status**

1. Show only the first few characters of the SHA-1 checksum instead of all 40:

**$ git log --abbrev-commit**

1. Display the date in a relative format (for example, “2 weeks ago”) instead of using the full date format:

**$ git log --relative-date**

1. Limiting log output:

**$ git log --since=<n>.<days-weeks-years>**

**$ git log --until=<n>.<days-weeks-years>**

**$ git log --before=<n>.<days-weeks-years>**

**$ git log --after=<n>.<days-weeks-years>**

**$ git log --since=”<specific-date , relative date>”**

1. Only show commits in which the author entry matches the specified string:

**$ git log –author <author-name>**

1. Only show commits in which the committer entry matches the specified string:

**$ git log --committer <committer-name>**

1. Only show commits with a commit message containing the string:

**$ git log --grep “<string>”**

1. Only show commits adding or removing code matching the string:

**$ git log -S “<string>”**

1. Prevent the display of merge commits:

**$ git log --no-merges**

1. If you want to redo that commit, make the additional changes you forgot, stage them, and commit again:

**$ git commit --amend**

1. Unstaging a Staged File:

**$ git reset HEAD <filename>**

**$ git restore --staged <filename>**

1. Unmodifying a Modified File:

**$ git checkout -- <filename>**

**$ git restore <filename>**

1. Showing Your Remotes:

**$ git remote**

1. Showing the URLs that git has stored for the shortname to be used when reading and writing to that remote:

**$ git remote -v**

1. Adding Remote Repositories:

**$ git remote add <shortname> <url>**

1. Fetching remote repository:

**$ git fetch <remote>**

1. Pulling remote repository:

**$ git pull <remote>**

1. Pushing remote repository:

**$ git push <remote> <branch>**

1. Inspecting a remote:

**$ git remote show <remote>**

1. Renaming a remote:

**$ git remote rename <old-remote> <new-remote>**

1. Removing a remote:

**$ git remote remove <remote>**

1. Listing all your tags:

**$ git tag**

1. Search for tags that match a particular pattern:

**$ git tag --list “<wildcard-pattern>”**

**$ git tag -l “<wildcard-pattern>”**

1. Creating an annotated tag:

**$ git tag -a <version> -m “<tagging-message>”**

1. See the tag data:

**$ git show <tag>**

1. Creating an lightweight tag:

**$ git tag <version>**

1. Tag a specific commit:

**$ git tag -a <version> <commit-checksum>**

1. Transferring tags to remotes:

**$ git push <remote> <tagname>**

1. Transferring all the tags:

**$ git push <remote> --tags**

1. Deleting tags:

**$ git tag -d <tagname>**

1. Deleting tags from remotes:

**$ git push <remote> :refs/tags/<tagname>**

**$ git push <remote> --delete <tagname>**

1. Checking out tags:

**$ git checkout <tagname>**

1. Git aliases:

**$ git config --global alias.<p> “<r>”**

**~ p: placeholder string**

**~ r: replacement string**

1. Git aliases remove:

**$ git config --global --unset alias.<p>**

**~ p: placeholder string**

1. Clear the terminal:

**$ clear**

1. Track remote branch:

**$ git branch --set-upstream-to=<rb> <lb>**

**~ rb: remote branch**

**~ lb: local branch**

**$ git branch -u <remote>/<branch>**

1. Create new branch:

**$ git branch <branch-name>**

1. Switch new branch:

**$ git checkout <branch-name>**

1. Show all branch and their commit:

**$ git log --all**

1. Divergent history:

**$ git log --oneline --graph --all**

1. Create and Switch to new branch:

**$ git checkout -b <branch-name>**

1. Merging branch:

**$ git merge <branch-name>**

1. Delete branch:

**$ git branch -d <branch-name>**

1. Remove untracked files:

**$ git clean -d -f**

1. List all branches:

**$ git branch**

1. See the last commit of each branch:

**$ git branch -v**

1. Show branched merged or not merged:

**$ git branch --merged**

**$ git branch --no-merged**

1. List of remote references and their info:

**$ git ls-remote <remote>**

**$ git remote show <remote>**

1. Push local branch to remote:

**$ git push <remote> <branch>**

1. Push local branch to remote with different name:

**$ git push <remote> <branch>:<diffbranch>**

1. Pull remote branch to local branch:

**$ git checkout -b <branch> <remote>/<branch>**

**$ git checkout –track <remote>/<branch>**

**$ git checkout <branch> ~ if not in local and only in remote**

1. Fetch everything:

**$ git fetch --all**

1. Delete remote branch:

**$ git push <remote> --delete <branch>**

**$ git push <remote> -d <branch>**

1. List branch with their remotes:

**$ git branch -vv**

1. Fetch all deleted branch from remote:

**$ git fetch -p**

1. command:

**$**

1. command:

**$**

1. command:

**$**

1. command:

**$**