Git Tutorial

# Git Commands

#### Practice remote repository: <https://github.com/sonu-dev/practice-git>

### Basic Commands

1. Git help <Command> [Description of specified command]
2. Git init [Create empty git repo]
3. Start . [Open the root folder of working directory]
4. Git status [List which files are staged, unstaged and untracked]
5. Git show <object> [show the detail of specified object]
6. Git gui | citool [graphical alternative to git commit/push]

### Getting new stuff from Remote

1. git fetch
2. git in
3. git pull
4. git pull --prune

### Pending changes and staging area

1. Git add . [add all pending changes to the stage level]
2. Git add <dir> [add all pending changes of specified directory]
3. Git add <filename> [add all pending changes of specified file]
4. Git rm <filename>[remove specified file]
5. Git diff [show unstaged changes b/w your index and working directory]
6. Git diff HEAD [show differences b/w working directory and last commit]
7. Git diff –cached [show differences b/w staged and last commit]

Or

1. Git diff –staged [show differences b/w staged and last commit]

### Commit changes

1. Git commit –m [staged pending changes with comment]
2. Git commit –a [automatically staged files that are modified and deleted, but new files you have not told Git to add are not affected ]

### Undo/Revert changes

1. Git revert <commit> [undo specified commit, then apply it to the current branch]
2. Git reset <filename> [remove file from the staging area, but leave the working directory unchanged]
3. Git reset [reset staging area to match most recent commit, but leave the working directory unchanged]
4. Git reset <commit> [move the current branch tip backward to specified commit, reset the staging area to match, but leave the working directory unchanged]
5. Git reset –hard [reset staging area and working directory to match most recent commit and **overwrites all changes in the working directory**]
6. Git reset –hard <commit>
7. Git reset –hard HEAD~<n> [reset staging area and working directory up to specified commit no from the top, also overwrites all changes in the working directory]
8. Git reset –soft [doesn’t reset staging area and working directory at all, just reset the HEAD ]

### Reverting already pushed commits

1. git revert [SHA1]
2. gitk/gitex

### Log

1. Git log -<limit> [show latest no. Of limit commits]
2. Git log –oneline Or git root [condense each commit to a single line]
3. Git log –p [show all commits with all changes]
4. Git log -- <file> [only display commits that have the specified file]
5. Git log ..<remote branch-name> [show all incoming commits to the remote branch]
6. Git log <remote branch-name>.. [show all outgoing commits to the remote branch]
7. Git log –graph
8. git log --graph --decorate –oneline
9. git log --graph --decorate –oneline --all

### List branches

1. Git branch [list all local branches]
2. Git branch –a [list all local + remote branches]
3. Git branch –r [list all remote branches]

### Checkout/Create and push branches to remote

1. Git checkout <branch-Name>
2. Git checkout –b <New branch-name> [create & checkout a new branch from the current branch]
3. Git push [if it fails at first, but gives you the command to use:]
4. Git push --set-upstream origin my-new-feature [push a new branch]
5. Git push [now it works as the upstream branch is set]

Else

1. Git push –u origin <current branch-name>

### Examining outgoing changes

1. git out [list out all out going commits to current branch]
2. git out --pretty=oneline
3. git outdiff [list out all out going commits with details to current branch]

### Getting new stuff from remote server

1. git fetch
2. git in [list out all in coming commits to current branch]
3. git pull
4. git pull --prune

### Examining diffs between branches

1. git log origin/dev..origin/feature
2. git log origin/dev..feature
3. git log origin/dev..
4. git log origin/feature..origin/dev
5. git log feature..origin/dev

### Delete branches

1. git branch -d feature [Delete a branch. The branch must be fully merged in its upstream branch]
2. git branch -D feature [Delete a branch irrespective of its merged status]
3. git push origin <a space>:feature (delete branch from remote server)
4. git pull –prune
5. git pull -p

### Stash Changes

1. Git stash [stash current pending changes]
2. Git stash list [list all stashes]
3. Git stash show <stash> eg. Git stash show stash@{0} [show the specified stash]
4. Git stash drop [drop top stash]
5. Git stash drop <stash>
6. Git stash apply [apply to top stash] – git stash pop
7. Git stash apply <stash>

### Merge and tools

1. Git merge <branch> [specified branch will be merged to current one]
2. Git mergetool [in case of conflicts, open default merge tool]
3. Git mergetool –t <mergetool> [open specified merge tool]
4. Git config --global merge.tool <merge-tool> [specified a default merge tool, it might be kdiff3, p4merge and winmerge etc.]
5. Git mergetool –tool-help [list all available merge tools and also list all possible merge tools]

### Config

1. Git config user.name <username> [Define author name to be used for all commits in current repo, commonly use –global flag to set config options for current user]
2. Git config --global --edit [open config file in edit mode]
3. Git config --global alias.<alias-name> <git-command> [create shortcut for a git command]

### Tags

1. git tag –a [ –m “msg”] <tag-name> [create new tag from the current branch]
2. git tag –list
3. git tag –d <tag-name> [delete the specified tag]
4. git checkout -b [branchname] [tagname]

### Cherry-commit

1. git cherry-pick <commit>
2. if there are conflicts, resolve those first then run git cherry-pick –continue
3. or if you want to cancel the cherry-pick operation run git cherry-pick –abort

### More...

1. Gitk [graphical Tcl/Tk based interface to local git repository]
2. Gitk <file |folder | branch> [show the history of specified file]
3. Git gc [run garbage collector for repository, optimize repository, should be run occasionally ]
4. Git grep <word/phrase> [show all occurrences into the whole tree, it is a case sensitive search]

### Config settings suggested by Jaakko:

git config --global push.**default** simple

git config --global core.autocrlf **false**

git config --global diff.tool kdiff3

git config --global credential.helper "!'C:/Program Files (x86)/Git/libexec/git-core/git-credential-store.exe'"

git config --global core.preloadindex **true**

git config --global core.fscache **true**

git config --global merge.renamelimit 6816

git config --global http.postbuffer 524288000

git config --global alias.in "log ..@{upstream}"

git config --global alias.out "log @{upstream}.."

git config --global alias.indiff "difftool -d ..@{upstream}"

git config --global alias.outdiff "difftool -d @{upstream}.."

<https://confluence.basware.com/display/APPKB/Git+training+-+command+reference>