# Git Hands-on Commands

| Commands | Locations | Comments |
|---|---|---|
| **--system** | /etc/gitconfig | One of the git configuration locations |
| **--global** | ~/.gitconfig or ~/.config/git/config | One of the git configuration locations |
| **--local** | .git/config | One of the git configuration locations |
| **Git config –list –show-origin** | | In one short, you will be able to find the location of each config files |
| **git config --global user.name "Sameer Zulfi"** | | Configure the User Name before start of any project |
| **git config --global user.email sameer@example.com** | | Configure the User Email before start of any project |
| **git config --global core.editor vi** | | Configure the Vi editor |
| **Git config --list** | | To find out the list of env., variables configured |
| **Git config –h** | | This is the help file for config |
| **Git add –h** | | Help file for add. You can add any commands in between to find out the help file |
| **git log** | | Will display all the commits |
| **Git log -p** | | P means patched output. It will display what was the content changed with the file name in detail manner |
| **git log –2** | | Show the last recent 2 commits |
| **Git log –since=”2008-01-15”** | | This pull the commits which happened since this date |
| **Git log –since=2.weeks** | | This pulls the commits which happened since last two weeks, you can add days or hours etc. |
| **Git log --stat** | | This command just give you which file changed and how many lines modified |
| **Git log <branch name>** | | This will pull the logs for that particular branch |
| **git log --pretty=oneline** | | This will give you the logs in pretty oneline format |
| **Git log --oneline** | | This will reduce the hash characters and provide you |

| | | |
|---|---|---|
| | | the commits in oneline as well |
| **git log --pretty=format:"%h - %an, %ar : %s"** | | This provides you output with commithash, author name, author committed date in relative format and subject of the commit |
| **git log --name-only** | | It will just give you only the filename which was modified |
| **git log –name-status** | | Will tell you the status of the files like (git status –s) |
| **Git log –relative-date** | | Date will be displayed in relative timing. Like 20 hours ago or 2 weeks ago etc., |
| **Git log –author=”Sameer”** | | Will fetch all the commits done by Sameer |
| **Git log –grep=”something”** | | This will search the word something in all the commit messages and fetch the result which is matching |
| **git log --pretty=format:"%H"** | | Without any spaces after format: This will display all the hash characters in long length |
| **git log --pretty=format:"%h"** | | This will display hash commits in short format |
| **git log --pretty=format:"%ad"** | | This will show the committed date in normal format |
| **The oneline and format options are particularly useful with another log option called –graph. This option adds a nice little ASCII graph showing your branch and merge history** | | |
| **git log --pretty=format:"%h %s" --graph** | | This produces a nice ASCII graph if you have lot of branches |
| **Git log --graph** | | Will show the detail ASCII graph. This graph will be helpful if you have multiple branches |
| **git log --since="2 years 1 day 3 minutes ago"** | | This will provide you exactly what you want |
| **git log --after="2019-08-15"** | | This provides you after this data, all the commits will be displayed |

| | | |
|---|---|---|
| **git log --before="2019-09-15"** | | Provide all the commits before this date |
| **Git log –pretty=”%h - %s” –author=”Sameer” –since=”2008-10-01” –before=”2008-11-01”** | | This provides you the commits in between these date |
| **Git log –branches=*** | | This will list out all the logs from all the branches |
| **Git commit --amend** | | You can amend your recent commit messages |
| **Git checkout -- filename** | | Unmodifying the modified file |
| **Git reset HEAD filename** | | Unstaging a staged file |
| **git config --global alias.co checkout** | | Git co instead of git checkout |
| **git config --global alias.br branch** | | Git br instead of git branch |
| **git config --global alias.ci commit** | | Git ci instead of git commit |
| **git config --global alias.st status** | | Git st instead of git status |
| **git config --global alias.last 'log -1 HEAD'** | | Type “git last” and analyse what’s going on |
| **.gitignore file** | Usually you need to create this file in the root repository directory | Blank lines or lines starting with # are ignored. |
| | *.a | Ignore all .a files |
| | /TODO | Only ignore the TODO file in the current directory, not subdir/TODO |
| | Build/ | Ignore all files in any directory named build |
| | Doc/*.txt | Ignore doc/notes.txt, but not doc/server/arch.txt |
| | Doc/**/*.pdf | Ignore all .pdf files in the doc/ directory and any of it’s subdirectories |
| | | https://github.com/github/gitignore this link has been updated by many developers and you will have an idea further |
| **Git diff** | | Differences between working directory and staging area |
| **Git diff –stated or git diff --cached** | | Differences between staging area and last committed snapshot |
| **Git difftool** | | Try this and analyze |
| **Git commit** | | This will open up the default editor |
| **Git commit -am** | | Skipping the staging area |
| **Git rm index.html** | | Removes the files |

| | | |
|---|---|---|
| **Git tag** | | Will list out all the tags |
| **Git tag –a v1.0 –m “version 1.0”** | | This adds the tag to the recent commit |
| **Git show v1.0** | | This will list out the tag connected to the particular commit details as well |
| **Git tag –a v1.2 <commithashid>** | | This will tag to the one particular hash id |
| **Git push origin v1.5** | | This will push the tag to your remote repo |
| **Git push origin --tags** | | This will push all your tags at once to your remote repo |
| **Git tag –d <tagname>** | | this will delete the tag from your locate repo |
| **Git push origin –delete <tagname>** | | This will delete the tag from your remote repo |
| **Git init** | | Initialize the git repo |
| **Git init directoryname** | | Will create the directory and initialize the git |
| **Git status -s** | | Will give you the status in very short format i.e, the first column is staging and second column is working directory |
| **git rm --cached --dry-run firstfile secondfile** | | Check what this command does |
| **git branch testing** | | This will create a new branch name “testing” |
| **git log --oneline --decorate** | | To check where the HEAD is pointing to. Infact we can just use git log command also for this |
| **Git checkout testing** | | This will switch the HEAD from master to testing. i.e, switching the branch from master to testing |
| **git log --oneline --decorate --graph --all** | | In ASCII graph with detail history of your branches |
| **Git checkout –b testing** | | This will immediately create a branch and switch it to that branch . this is equivalent two commands ( git branch testing & git checkout testing) |
| **Git merge branchname** | | This will merge the branches. For example if you want to merge hotfix with your main master branch. You need to type “git merge hotfix” . |

| | | |
|---|---|---|
| | | Condition is you need to be on master branch |
| **Git branch –d branchname** | | This will delete the particular branch |
| **Git branch** | | This will list out the # of branches |
| **Git branch -v** | | This will list out the recent commits done on each branches |
| **Git branch --merged** | | This will list out how many branches are merged with the current branch |
| **Git branch –no-merged** | | This will list out how many branches are not yet merged with the current branch |
| **Git branch –no-merged master** | | This will tell you what are branches not merged with the master branch. |
| **Git remote show remoteservername** | | This will provide full list of remote branches |
| | Origin/master | Means on the origin remote server / master branch |
| | Origin/hotfixbranch | Means on the origin remote server/hotfix branch |
| **Git clone https://github.com/beginners-sameer/GitBranchingRemotebesant.git** | | This clones your remote repo |
| **Git fetch remoteservername** | For ex : git fetch origin | It fetches any data from the remote server that you don't yet have and it updates your local database |
| **Git remote add shortname remoteservername** | Git remote add projalpha https://github.com/beginners-sameer/GitBranchingRemotebesant.git | This is alternative for git clone. |
| **Git push remoteservername branchname** | Git push origin serverfix or git push origin serverfix:serverfix or git push origin serverfix:awesomebranch | This will push the changes to the remote server called "origin" with the branch named "serverfix |
| **Git checkout –b localbranchname remoteservername/branchname** | Git checkout –b serverfix origin/serverfix | This creates a local branch called "serverfix" and switched it to that then it tracks with remote server called origin with remote branch "serverfix" |
| **Git checkout –track origin/serverfix** | | Branch serverfix set up to track remote branch serverfix from origin. |

| | | |
|---|---|---|
| | | Switched to a new branch 'serverfix' |
| **git checkout -b sf origin/serverfix** | | Branch sf set up to track remote branch serverfix from origin.<br>Switched to a new branch 'sf'. Now, your local branch sf will automatically pull from origin/serverfix |
| **git branch -u origin/serverfix** | | Branch serverfix set up to track remote branch serverfix from origin. |
| **git branch -vv** | | This will list out what tracking branches you have setup |
| **git fetch --all** | | This will fetch all the remote repos branches info |
| **Git push remoteservername –delete remotebranchname** | git push origin --delete serverfix | This will delete the remote branch name |
| **git remote rename remoteservername newremoteservername** | Git remote rename origin main | The origin will be renamed to main. If you type “git remote” now, it will give you only main. |
| **Git remote remove remoteservername** | Git remote remove main | It will remove the remote server from the local machine |
| **How to avoid typing https password every time when you push and pull** | Step1 : git config credential.helper store Step2: After you hit enter the previous command, type this command “git push https://entire repo name” enter. That’s all | Next time when you push or pull , it will not ask you to type the credentials |
| **git push --set-upstream remoteservername remotebranchname** | git push --set-upstream origin serverfix | This will setup the remote server and branch name with respect to the current branch you are on |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |