

git show git diff

git diff --staged HEAD

git diff HEAD

git diff --color git diff --staged

git tag

git push my-remote my-branch v1.0 --delete my-remote v1.0 -a v1.0 -m msg -delete v1.0 git show v1.0 hsnd git tag git tag git

| git fetch ---tags

git pull my-remote my-branch

| git clean -f | git clean -f -d/git clean -fd git clean -f -X/git clean -fX

git clean -f -x/git clean -fx

git clean -d --dry-run

# shows one or more objects (blobs, trees, tags and commits).

# show changes between commits, commit and working tree #show changes between working directory vs last commit

#show changes between stage area vs last commit

# show colored diff

# Shows changes staged for commit

# shows all the tags

# creates an annotated tag

# shows the description of version-1.0 tag

# deletes the tag in local directory

# deletes the tag in my-remote (be carefore to not delete a branch)

# push v1.0 tag to my-remote in my-branch

# pulls the tags from remote

# pulls and tries to merge my-branch from my-remote to the current branch git pull = git fetch 86 get merge

# clean untracked files permanently

# To remove directories permanently

# To remove ignored files permanently

# To remove ignored and non-ignored files permanently

shows what would be deleted



	5	3	,
	+		,
_		_	

no-pager log git

-oneline git log

decorate graph oneline git log

--since=<time> git log -p <file\_name> git log

^<Branch2> <Branch1> log git

git log

--oneline git log

'<string/regex>' --heading --line-number grep git

---grep='<string/regex> log git

reflog git

ls-files git

-m msg git commit

-m "description" -m "title" commit git

-- amend git commit --no-edit --amend commit git

--author='Author Name git commit --amend

git push my-remote my-branch

git revert <commit-id>

shows

the log of # shows command shows the log of single line

# shows the log of commits, in a single line with graph

# shows the log of given time

change over time for a specific file

# lists commit(s) in branch1 that are not in branch2

# lists the last x commits, each commit in single line

# Find lines matching the pattern in tracked files

Search Commit log

# record when the tips of branches and other references were updated in the local repository.

# show information about files in the index and the working tree

msg commit changes with a # commit changes with a title and description

# combine staged changes with the previous commit, or edit the previous commit message without changing its snapshot

# amends a commit without changing its commit message

# Amend the author of a commit

# pushes the commits to the my-remote in my-branch (does not push the tags)

# Undo a



| git remote add origin https://github.com/ repo\_name.git init git

git clone <address>

<path/ -b <br/>branch\_name> git clone <address> to/directory> single -b <br/>branch\_name> <address> clone branch git

git add <file\_name>

add git

file txt reset git

-hard reset git

<commit\_id> soft reset git

<commit\_id> --mixed reset git

<commit\_id> -hard reset git

file txt ᇤ git --cached file txt r. git

status git branch git git branch

..< commit\_id\_B> <commit\_id\_A>^ cherry-pick git

<commit\_id>

cherry-pick

git

add remote reposiory

# creates a git repo from given address (get the address from your git-server) | git clone <address> -b <branch\_name> <path/ to/directory>

# clones a git repo from the address into the given directory and checkout's the given branch

adds(stages) file.txt to the git

# adds(stages) all new modifications, deletions, creations to the git

# Throws away all your uncommitted changes, hard reset files to HEAD

moves the head pointer

then copies is now # moves the head pointer and the files from the commit it pointing to the staging area,

# the default when no argument is provided # moves the head pointer and then copies the files from the commit it is now pointing to the staging area # and working directory thus, throw away all uncommitted changes

removes file.txt both from git and file system # only removes file.txt both from git index

the modifications and staged yet # shows

# shows all the branches is shown with a star)

44 of commits where is for including # pick the entire range is older than B ( the as well )



| git config --global --list | git config --global --edit

| git config ---global alias.<handle><command>

git config --global core.editor <editor\_name>

| git archive <branch\_name> ---format=zip outpute=./<archive\_name>.zip

git stash git stash -u git stash save "msg"

git stash list

git stash pop

| git stash pop stash@{2}

git stash show

git stash apply

| git stash branch my-branch stash@{**a**}

git stash drop stasha{1}

git stash clear

git rebase -i <commit\_id>

git rebase ---abort

git rebase --- abort

# lists the git configuration for all repos

# opens an editor to edit the git config file # add git aliases to speed up workflow ,

# config default editor

# create an archive of files from a named tree # stashes the staged and unstaged changes (git status will be clean after it) # stash everything including new untracked files (but not .gitignore)

# stash with a msg

# list all stashe

# delete the recent stash and applies it

# delete the {2} stash and applies it

# shows the description of stash

# keep the stash and applies it to the gi

# creates a branch from your stash

# deletes the {1} stash

# clears all the stash

# Rebase commits from a commit ID

# Abort a running rebase

# Continue rebasing after fixing all conflicts