Git Documentation

1. [Create a new repository](https://help.github.com/articles/creating-a-new-repository) on GitHub. To avoid errors, do not initialize the new repository with *README*, license, or gitignore files. You can add these files after your project has been pushed to GitHub.
2. Open Git Bash.
3. Change the current working directory to your local project.
4. Initialize the local directory as a Git repository.
5. git init
6. Add the files in your new local repository. This stages them for the first commit.
7. git add .
8. # Adds the files in the local repository and stages them for commit. To unstage a file, use 'git reset HEAD *YOUR-FILE*'.
9. Commit the files that you've staged in your local repository.
10. git commit -m "First commit"
11. # Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.

At the top of your GitHub repository's Quick Setup page, click to copy the remote repository URL.

1. In the Command prompt, [add the URL for the remote repository](https://help.github.com/articles/adding-a-remote) where your local repository will be pushed.
2. git remote add origin *remote repository URL*
3. # Sets the new remote
4. git remote -v
5. # Verifies the new remote URL
6. [Push the changes](https://help.github.com/articles/pushing-to-a-remote) in your local repository to GitHub.
7. git push origin master

# Pushes the changes in your local repository up to the remote repository you specified as the origin

when you get any error like below

Another git process seems to be running in this repository, e.g.

an editor opened by 'git commit'. Please make sure all processes

are terminated then try again. If it still fails, a git process

may have crashed in this repository earlier:

remove the file manually to continue.

try the below command

$ rm -f ./.git/index.lock

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

reset Reset current HEAD to the specified state

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

checkout Switch branches or restore working tree files

commit Record changes to the repository

diff Show changes between commits, commit and working tree, etc

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

git help -a

available git commands in 'C:\Program Files\Git\mingw64/libexec/git-core'

add gc receive-pack

add--interactive get-tar-commit-id reflog

am grep remote

annotate gui remote-ext

apply gui--askpass remote-fd

archimport gui--askyesno remote-ftp

archive gui.tcl remote-ftps

askpass hash-object remote-http

bisect help remote-https

bisect--helper http-backend repack

blame http-fetch replace

branch http-push request-pull

bundle imap-send rerere

cat-file index-pack reset

check-attr init rev-list

check-ignore init-db rev-parse

check-mailmap instaweb revert

check-ref-format interpret-trailers rm

checkout log send-email

checkout-index ls-files send-pack

cherry ls-remote sh-i18n--envsubst

cherry-pick ls-tree shortlog

citool mailinfo show

clean mailsplit show-branch

clone merge show-index

column merge-base show-ref

commit merge-file stage

commit-tree merge-index stash

config merge-octopus status

count-objects merge-one-file stripspace

credential merge-ours submodule

credential-manager merge-recursive submodule--helper

credential-store merge-resolve subtree

credential-wincred merge-subtree svn

cvsexportcommit merge-tree symbolic-ref

cvsimport mergetool tag

daemon mktag unpack-file

describe mktree unpack-objects

diff mv update

diff-files name-rev update-git-for-windows

diff-index notes update-index

diff-tree p4 update-ref

difftool pack-objects update-server-info

difftool--helper pack-redundant upload-archive

fast-export pack-refs upload-pack

fast-import patch-id var

fetch prune verify-commit

fetch-pack prune-packed verify-pack

filter-branch pull verify-tag

fmt-merge-msg push web--browse

for-each-ref quiltimport whatchanged

format-patch read-tree worktree

fsck rebase write-tree

fsck-objects rebase--helper

git commands available from elsewhere on your $PATH

flow lfs

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

usage: git rm [<options>] [--] <file>...

-n, --dry-run dry run

-q, --quiet do not list removed files

--cached only remove from the index

-f, --force override the up-to-date check

-r allow recursive removal

--ignore-unmatch exit with a zero status even if nothing matched