

Git : Git is free and open source distributed version control system.

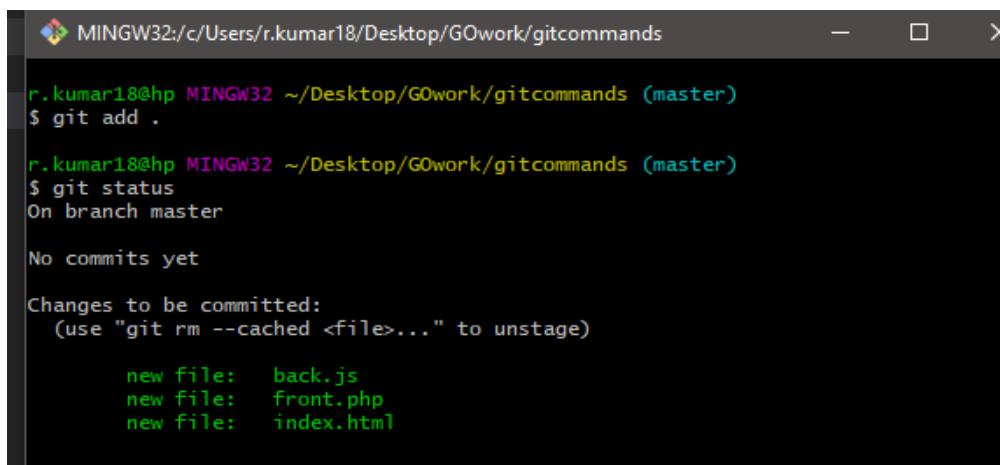
Git purpose is to keep track of projects and files as they change over time with manipulations happening from a different user.

Some commonly using commands in git are as following.

Commands	Description
\$git init	Initialize git in local folder
\$git add	Add a file to the staging area
\$git status	Check the status
\$git commit	Add the staged file
\$git log	lists the commits made in that repository.
\$git diff	What you change in file
\$git rm -f <file_name>	Remove files
\$git checkout --file	to discard last changes in current commit
\$git commit --amend -m "msg"	
\$ git reset (SAWID) head (file)	To move previous commit and delete all
\$git rm --cached file	To delete staged file
\$git commit -a "msg"	Add and commit at same time
\$git branch	To check current branch
\$git branch <branch_name>	To create new branch
\$git checkout <branch_name>	To checkout this branch
\$git checkout -b <branch_name>	To crate and checkout also at same time
\$git branch -m oldname new name	To rename branch name
\$git branch -d branchname	To delete branch
\$git merge branchname	To merge two branches
\$git stash save stash_name	To create a new stash
\$git stash apply stash@{number of stash}	To apply stash
\$git tag ticket_name saw1	To give name to saw1
\$git clone repository_id	To download project from github

Basic commands:

Git init: When you run git init in a new or existing directory, Git creates the .git directory, which is where almost everything that Git stores and manipulates is located.



```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git add .

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git status
On branch master

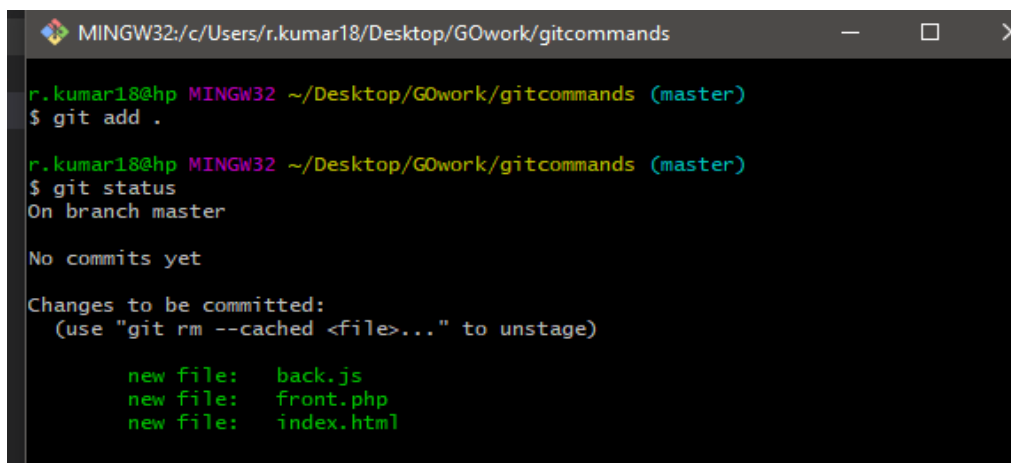
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

       new file:   back.js
       new file:   front.php
       new file:   index.html
```

Git add:- The git add command adds a change in the working directory to the staging area.

Git status is mostly used command in git ,gives status of tracked and untracked changes in your local repository.



```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git add .

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

       new file:   back.js
       new file:   front.php
       new file:   index.html
```

Git add.

- Git add -A :- To add all.
- \$git add . To add new and modified but not deleted.
- \$git add -u To add modified and deleted but not new files/directories

```
r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git log --oneline
7ed4da7 (HEAD -> master) add model again in master
2309d32 models add
0a34bec c file add
a5eaff2 index is added

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git checkout header
```

Git log `--oneline` Prints log of past commits in one line.

Git Log Prints log of past commits

- Git log contains
 - i. Git hash(SHA)
 - ii. Git author meta data
 - i. Date and time of commit
 - ii. Message

```
MINGW32:/e/couresra/notes/Golang/gito
r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (header)
$
r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (header)
$ git checkout master
Switched to branch 'master'

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git log
commit 2309d3213d3a5aee6c52284cf414891e9ab6d327 (HEAD -> master)
Author: r.kumar18 <rkrko3@gmail.com>
Date: Sun Mar 29 21:38:39 2020 +0530

    models add

commit 0a34bec37be64b8f93d92929d3f69524bbb9e5a3
Author: r.kumar18 <rkrko3@gmail.com>
Date: Sun Mar 29 21:38:11 2020 +0530

    c file add

commit a5eaff234611418faf86a41e7b0b2e9caa106ef6
Author: r.kumar18 <rkrko3@gmail.com>
Date: Sun Mar 29 21:37:32 2020 +0530

    index is added
```

Branching & Merging

```
MINGW32:/e/couresra/notes/Golang/gito
r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git commit -m "models add"
[master 2309d32] models add
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 gito/model.php

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git branch
* master

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git branch header

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git checkout header
Switched to branch 'header'

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (header)
$ git branch
* header
  master
```

Git Branch List branches (the asterisk denotes the current branch)

Git branch branch_name Create a new branch

```
r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git log --oneline
7ed4da7 (HEAD -> master) add model again in master
2309d32 models add
0a34bec c file add
a5eaff2 index is added

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (master)
$ git checkout header
Switched to branch 'header'

r.kumar18@hp MINGW32 /e/couresra/notes/Golang/gito (header)
$ git log --oneline
b3fa51a (HEAD -> header) new models in header
2309d32 models add
0a34bec c file add
a5eaff2 index is added
```

Git checkout branch_name Switch to a branch.

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
no changes added to commit (use "git add" and/or "git commit -a")
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git add back.js

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git commit -m "new js file changes"
[master a4c3b10] new js file changes
1 file changed, 2 insertions(+), 1 deletion(-)

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git status
On branch master
nothing to commit, working tree clean

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git merge header
Auto-merging gitcommands/back.js
CONFLICT (content): Merge conflict in gitcommands/back.js
Automatic merge failed; fix conflicts and then commit the result.

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master|MERGING)
$
```

Git add filename Add a file to the staging area

Git commit -m "message" Add a file to the staging area

Git merge branchname Merge a branch into the active branch

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master|MERGING)
$ git add back.js

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master|MERGING)
$ git commit -m "merge files "
[master 203b0e6] merge files

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git diff
diff --git a/gitcommands/back.js b/gitcommands/back.js
index 7477f81..f694687 100644
--- a/gitcommands/back.js
+++ b/gitcommands/back.js
@@ -1,6 +1,2 @@
 back
-<<<<<<< HEAD
-//new back
-=====
 header //changes back
->>>>>>> header
```

Git diff shows the changes you have done in your existing file (compare different versions of your files)

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git rm --cached -r back.js
error: the following file has staged content different from both the
file and the HEAD:
    gitcommands/back.js
(use -f to force removal)

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git rm --cached -f back.js
rm 'gitcommands/back.js'

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    back.js
```

Git rm --cached -f filename Remove a file (or folder)

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/git...

$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    back.js

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ ^C

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    back.js

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git log --oneline
203b0e6 (HEAD -> master) merge files
a4c3b10 new js file changes
a75e5e4 git ignore file
6d00796 (header) header changes back files
95acb82 all files added

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git reset --soft HEAD~2

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git log --oneline
a75e5e4 (HEAD -> master) git ignore file
95acb82 all files added
```

Git reset

```
git reset --soft <commit id>
```

```
git reset --mixed <commit id>
```

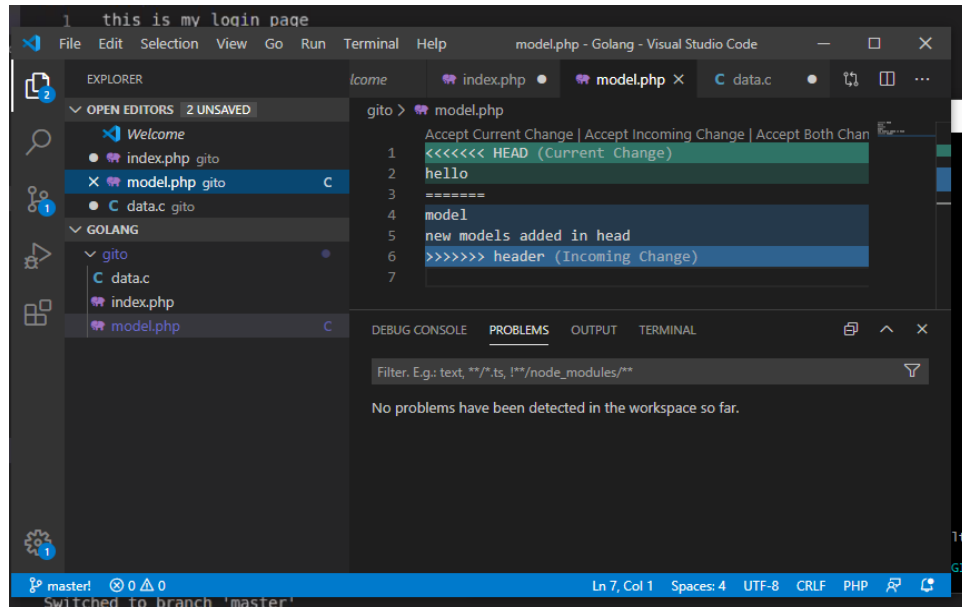
- The difference between `--mixed` and `--soft` is whether or not your index is also modified. So, if we're on branch master with this series of commits: - A - B - C (master) • HEAD points to C and the index matches C. • When we run `git reset --soft B`, master (and thus HEAD) now points to B, but the index still has the changes from C; `git status` will show them as staged. So if we run `git commit` at this point, we'll get a new commit with the same changes as C. •

- Okay, so starting from here again: • - A - B - C (master) • Now let's do `git reset --mixed B`. (Note: `--mixed` is the default option). Once again, master and HEAD point to B, but this time the index is also modified to match B. If we run `git commit` at this point, nothing will happen since the index matches HEAD. We still have the changes in the working directory, but since they're not in the index, `git status` shows them as unstaged. To commit them, you would `git add` and then commit as usual. • _____ • And finally, `--hard` is the same as `--mixed` (it changes your HEAD and index), except that `--hard` also modifies your working directory. If we're at C and run `git reset --hard B`, then the changes added in C, as well as any uncommitted changes you have, will be removed, and the files in your working copy will match commit B. Since you can permanently lose changes this way, you should always run `git status` before doing a hard reset to make sure your working directory is clean or that you're okay with losing your uncommitted changes.

Reset • `git reset --hard <commit id>` resets the HEAD to any commit mentioned in command. And throws away all your changes which is made after the commit corresponding to the commit id.

NOTE:-

- **Git doesn't upload a empty file or folder.**
- **.gitignore** file is a text file that tells **Git** which files or folders to ignore in a project. Here you can upload empty folder also .
- **Conflict:** A **conflict** arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other.



This is the example of Conflict. Mainly this error only solved by human intelligence.

The commonly using tools for handling the conflict are as following.

1. Vimdiff
2. Tortoise
3. Meld

Git "fetch" Downloads commits, objects and refs from another repository.

Fetch downloads only new data from a remote repository.

Stash : in git stash is like storage area , which hold our new changes in file .Using this we can move in other branch .

And can also use that stash in other branch file.

\$git stash save "stash name" command is use to create a stash.

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (header)
$ git stash save "stash1"
Saved working directory and index state On header: stash1

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (header)
$ git checkout header
Already on 'header'

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (header)
$ git stash list
stash@{0}: On header: stash1
```

\$git stash list command is use to check list of stash .

```
r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (header)
$ git stash show -p stash@{0}
diff --git a/gitcommands/front.php b/gitcommands/front.php
index f6029ad..edde338 100644
--- a/gitcommands/front.php
+++ b/gitcommands/front.php
@@ -1,2 @@
 view one
+/sss/ss/
\ No newline at end of file
```

\$git stash show stash@{number of stash} command is use to check where is stash.

\$git stash show -p stash@{number of stash} command is use to what contains stash .

```
MINGW32:/c/Users/r.kumar18/Desktop/GOwork/gitcommands
$ git branch
* header
  master

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (header)
$ git checkout master
Switched to branch 'master'

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git stash pop stash@{0}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   front.php

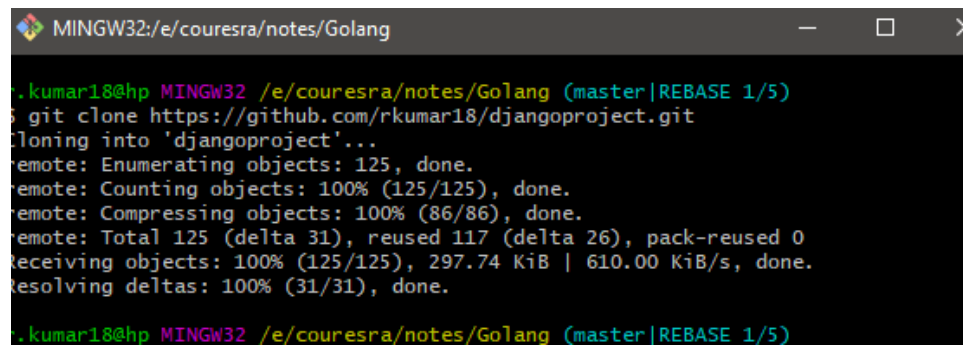
no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{0} (0b97cdedbf7cdaa9dff0e74975819766dde1d2e1)

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
$ git stash list

r.kumar18@hp MINGW32 ~/Desktop/GOwork/gitcommands (master)
```

\$git stash apply stash@{number of stash} command is use to **copy and paste** the stash in the file .

\$git stash pop stash@{number of stash} command is use to **cut and paste** the stash in file .Here **stash is deleted automatically.**

A terminal window titled 'MINGW32:/e/couresra/notes/Golang' showing the execution of a git clone command. The output shows the cloning process from a GitHub repository, including object enumeration, counting, and compression statistics.

```
.kumar18@hp MINGW32 /e/couresra/notes/Golang (master|REBASE 1/5)
$ git clone https://github.com/rkumar18/djangoproject.git
Cloning into 'djangoproject'...
remote: Enumerating objects: 125, done.
remote: Counting objects: 100% (125/125), done.
remote: Compressing objects: 100% (86/86), done.
remote: Total 125 (delta 31), reused 117 (delta 26), pack-reused 0
Receiving objects: 100% (125/125), 297.74 KiB | 610.00 KiB/s, done.
Resolving deltas: 100% (31/31), done.
.kumar18@hp MINGW32 /e/couresra/notes/Golang (master|REBASE 1/5)
```

Download your project from the github .

\$Git clone projectlink

Github Commands :

- ✓ Git init
- ✓ Git remote add origin <URL>
- ✓ Git push -u origin master
- ✓ Git pull origin master (will fetch all the changes from the remote's master and merge it into local .

Some alternatives to Github.

- Bitbucket.
- SourceForge.
- GitLab.
- Kiln.
- Codeplex.
- Beanstalk.

FAQ:

- How delete branch in git ?

`$git branch -d <branch_name>`

- How rename branch ?

`$git branch -m oldname newname`

- What is SHA ?

SHA is a 40 character checksum data generated by SHA (Secure Hash Algorithm) algorithm. It is unique for all commits.

References:

- ✓ <https://git-scm.com/book/en/v2/Git-Branching-Branched-in-a-Nutshell>
- ✓ <https://www.javatpoint.com/git-log>