

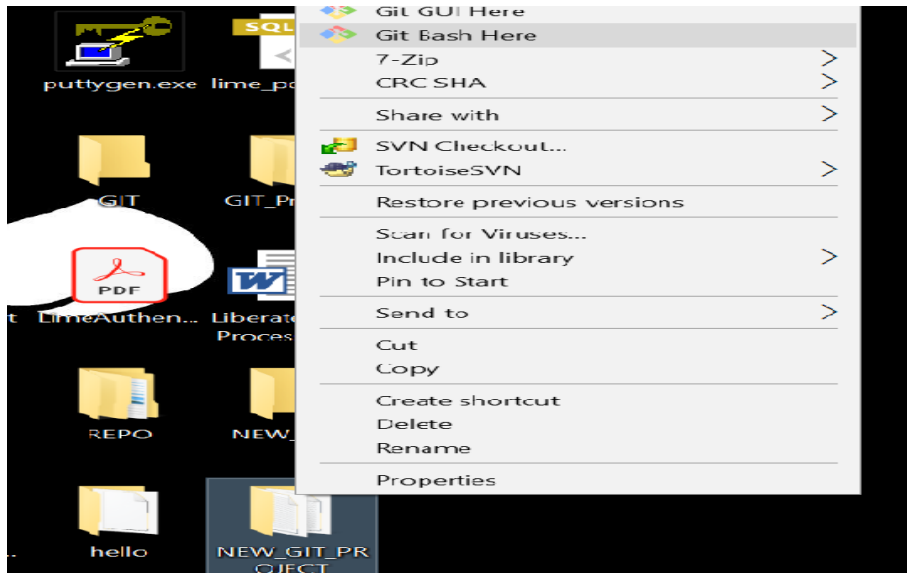
## GIT Documented by Praveen

### Index

1.	create git repository in your local desktop	2
2.	Using git init command	2
2.1	Create one file	3
2.2	git status command	3
2.3	git add	3
2.4	git commit command	4
2.6	git log command	4
3	git revert	6
3.1	git reset --soft command	6
3.2	git reset --hard command	7
4	create github account	7
5	local repository to github	8
6	clone the github	9
6.1	Copy url from github	9
6.2	Clone the repo into your local	9
6.3	How to create branch from master and checkout	10
7	git merge	10
7.1	git merge file	11
7.3	Check the merged branch changes are reflected on github	11
7.4	pull request	12
8	merge tool installation	13
8.1	The process of git merge tool	13
8.2	The result of merged file	15
8.3	commit modified files	15
9	git stash	16
9.1	Create a stash	16
9.2	git stash pop	16
9.3	delete the Stash	16
10	git Tag	17
10.1	create tags	17
10.2	tag commit and push	17
10.3	check tags in git hub	17
10.4	List all tags	18
10.5	Checkout tags	18
11	git pull and fetch	18
11.1	git pull	18
11.2	git fetch	18
12	git commands	19

## 1. How to create git repository in your local desktop:-

First download git package on your windows machine using: - <https://git-scm.com/downloads> and Create a folder (like:-NEW\_GIT\_PROJECT) on your desktop, open **git bash** and follow the below commands



## 2 . Using **git init** command it will create .git repository into your **NEW\_GIT\_PROJECT** folder.

MINGW64:/c/Users/vmatheba/Desktop/NEW\_GIT\_PROJECT

```
vmatheba@LIN70000272 MINGW64 ~/Desktop
$ mkdir NEW_GIT_PROJECT

vmatheba@LIN70000272 MINGW64 ~/Desktop
$ cd NEW_GIT_PROJECT/

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT
$ git init
Initialized empty Git repository in C:/Users/vmatheba/Desktop/NEW_GIT_PROJECT/.git/

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ \ls -la
total 36
drwxr-xr-x 1 vmatheba 1049089 0 Nov  8 19:44 .
drwxr-xr-x 1 vmatheba 1049089 0 Nov  8 19:44 ..
drwxr-xr-x 1 vmatheba 1049089 0 Nov  8 19:44 .git

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ |
```

2.1 Create one file1.txt file using vi editor into git project and commit to your local git repository

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ vi file1.txt
```

```
MINGW64:/c/Users/vmatheba/Desktop/NEW_GIT_PROJECT
This is my first commit|
```

2.2 Before add check the status of current file using **git status** command, if it is in red colour, so changes is in work space area as shown in below screen shot.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file1.txt

nothing added to commit but untracked files present (use "git add" to track)

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ |
```

2.3 Add file1.txt file using **git add** command and check the file status(if you want to add more files using **git add .** or **-A** or **\*** commands), if file status showing in green colour, then it is in staging area so ready to commit into local repository.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git add file1.txt
warning: LF will be replaced by CRLF in file1.txt.
The file will have its original line endings in your working directory.

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master

No commits yet

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

        new file:   file1.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ |
```

2.4 If you want to commit staged files to local repository using **git commit** command, it will push the changes into your local git repository.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git commit -m "Commit file1.txt file in local repository"
[master (root-commit) 9f0a8f6] Commit file1.txt file in local repository
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```

2.5 So as per below screen shot your working tree is in clean state (i.e:- All files in your workspace committed into local repository).

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
nothing to commit, working tree clean
```

2.6 Once committed to local repository, then type **git log** command (because all files are saved in objects manner that is why git status is not working in this stage), it will show commit id. If you want to see which files committed under this commit id using **git show cid** command.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git log
commit 9f0a8f6630ae602df488429aacd46b582d037626 (HEAD -> master)
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 19:56:13 2018 +0530

    Commit file1.txt file in local repository
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git show 9f0a8f6630ae602df488429aacd46b582d037626
commit 9f0a8f6630ae602df488429aacd46b582d037626
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 19:56:13 2018 +0530

    Commit file1.txt file in local repository

diff --git a/file1.txt b/file1.txt
new file mode 100644
index 0000000..efbc80b
--- /dev/null
+++ b/file1.txt
@@ -0,0 +1 @@
+this is my first file
```

2.7 Now create another two files file2.txt and file3.txt using vi editor and follow the steps as per below screen shots(i.e: git add and git commit)

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ vi file2.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ vi file3.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file2.txt
        file3.txt

nothing added to commit but untracked files present (use "git add" to track)

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ |
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git add .
warning: LF will be replaced by CRLF in file2.txt.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in file3.txt.
The file will have its original line endings in your working directory.

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   file2.txt
        new file:   file3.txt
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git commit -m "commit file2 and file3 into local repository"
[master 12b227a] commit file2 and file3 into local repository
2 files changed, 2 insertions(+)
create mode 100644 file2.txt
create mode 100644 file3.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git log
commit 12b227aa9e6f7a0e519401a3e9f8681d30cbd409 (HEAD -> master)
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 20:02:28 2018 +0530
```

commit file2 and file3 into local repository

```
commit 9f0a8f6630ae602df488429aacd46b582d037626
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 19:56:13 2018 +0530
```

Commit file1.txt file in local repository

3. How to revert back the change from local repository to staging and staging to workspace(conversely)

3.1 Using **git reset --soft cid**, it will revert back your local repository files into staging area, as shown in below screen shot.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git log
commit 12b227aa9e6f7a0e519401a3e9f8681d30cbd409 (HEAD -> master)
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 20:02:28 2018 +0530
```

commit file2 and file3 into local repository

```
commit 9f0a8f6630ae602df488429aacd46b582d037626
Author: Praveen <raju.varahala@gmail.com>
Date: Thu Nov 8 19:56:13 2018 +0530
```

Commit file1.txt file in local repository

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git reset --soft 9f0a8f6630ae602df488429aacd46b582d037626
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
new file:   file2.txt
new file:   file3.txt
```

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ |
```

3.2 Using **git revert head file name**, it will revert back your staged files into workspace, as shown in below screen shot.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git reset head file2.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

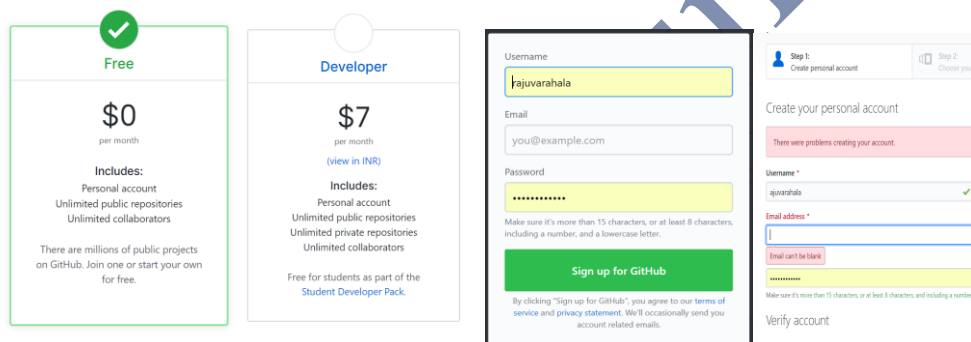
    new file:   file3.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    file2.txt
```

3.3 How to revert back local repository changes into workspace:- **git revert --mixed cid**.

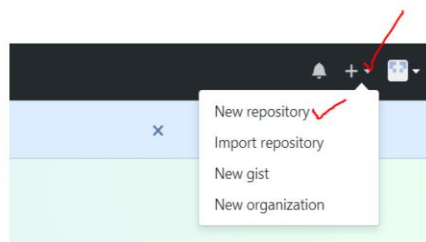
#### 4. How to create git-hub account:-



Create new repository on above signup git hub account



OR



#### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: devopsmethodologies / Repository name: new-project

Description (optional):

☒ Public  
Anyone can see this repository. You choose who can commit.

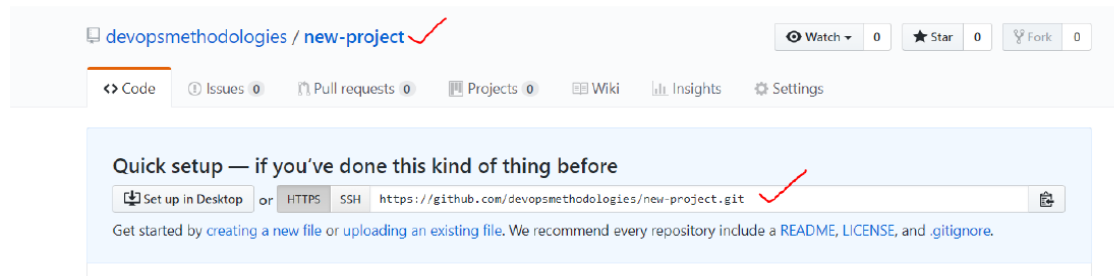
☐ Private  
You choose who can see and commit to this repository.

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None

Create repository

## Repository created as new-project



### 5. How to add your local repository to github:

5.1 Using **git remote add origin url** command as shown in below, it will automatically added created repository into github.

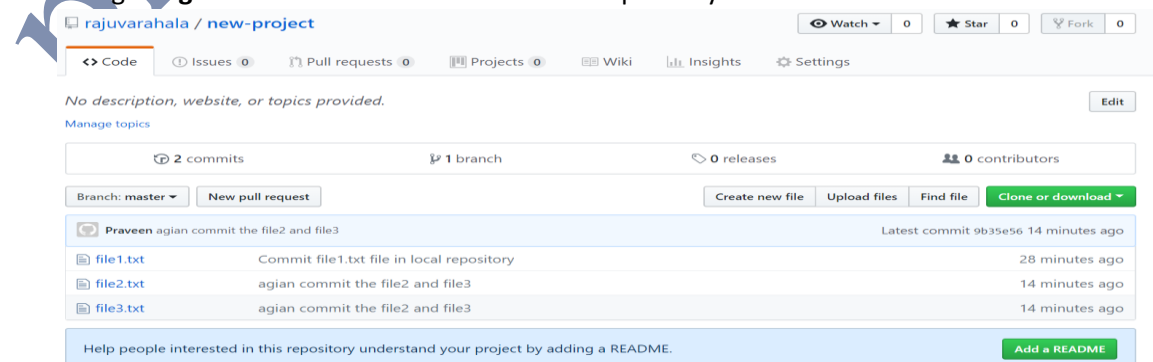
Note: - We have already created git local repository in (refer:-point 1) using **git init** command and same has been pushed into central git hub as shown in below screen shot. But first we need to create one empty repository on git hub and add your local repository with same name [In above point we have already created empty git repository on git hub (i.e:- new-project)].

Once added to origin and then need to push the same repository into origin master branch

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git remote add origin https://github.com/rajuvarahala/new-project.git

vmatheba@LIN70000272 MINGW64 ~/Desktop/NEW_GIT_PROJECT (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 585 bytes | 146.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/rajuvarahala/new-project/pull/new/master
remote:
To https://github.com/rajuvarahala/new-project.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

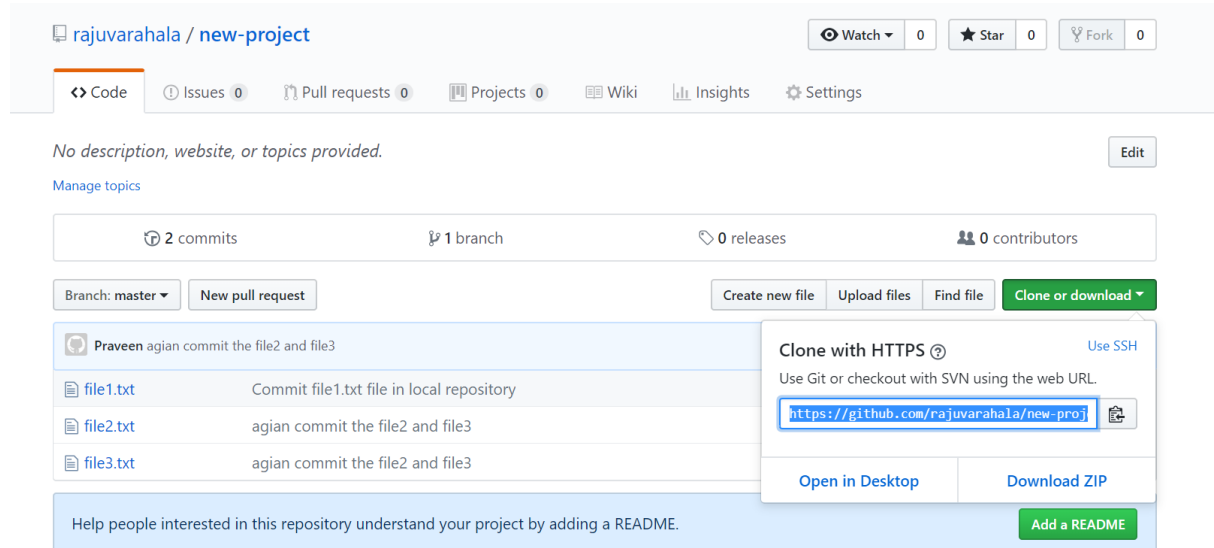
### 5.2 And go to **github** and *check* the above added repository are exist or not





## 6. How to clone the new created repository on Github and clone the same repo into your local

### 6.1 :-Copy url from github



rajuvarahala / new-project

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

2 commits 1 branch 0 releases 0 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Praveen agian commit the file2 and file3

File	Commit Message
file1.txt	Commit file1.txt file in local repository
file2.txt	agian commit the file2 and file3
file3.txt	agian commit the file2 and file3

Help people interested in this repository understand your project by adding a README. [Add a README](#)

Clone with HTTPS  
Use Git or checkout with SVN using the web URL.  
<https://github.com/rajuvarahala/new-project.git>  
[Open in Desktop](#) [Download ZIP](#)

### 6.2 :- Clone the repo into your local

```
vmatheba@LIN70000272 MINGW64 ~/Desktop
$ mkdir Project_for_cloning

vmatheba@LIN70000272 MINGW64 ~/Desktop
$ cd project_for_cloning

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning
$ git init
Initialized empty Git repository in C:/Users/vmatheba/Desktop/project_for_cloning/.git/

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning (master)
$ git clone https://github.com/rajuvarahala/new-project.git
Cloning into 'new-project'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning (master)
$ \ls -lrt
total 0
drwxr-xr-x 1 vmatheba 1049089 0 Nov 8 20:27 new-project

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning (master)
$ cd new-project/

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ \ls -lrt
total 3
-rw-r--r-- 1 vmatheba 1049089 25 Nov 8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov 8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov 8 20:27 file1.txt
```

### 6.3 cthe same branch

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ git branch GIT_BRANCH

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ \ls -lrt
total 3
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ clear

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ \ls -lrt
total 3
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ git branch
GIT_BRANCH
* master

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ git checkout GIT_BRANCH
Switched to branch 'GIT_BRANCH'

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ \ls -lrt
total 3
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt
```

## 7. Git Merge:

When two developers tried to commit the same files with their changes then git Conflicts arise. In order to resolve this one has to use **merge** tool and pull compare the changes and then push the modified file.

Checkout done, so now you are in GIT\_BRANCH then try to create one file and **merge** the same file into master branch.

```
vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git status
On branch GIT_BRANCH
nothing to commit, working tree clean

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ vi file4.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git status
On branch GIT_BRANCH
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    file4.txt

nothing added to commit but untracked files present (use "git add" to track)

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git add file4.txt
warning: LF will be replaced by CRLF in file4.txt.
The file will have its original line endings in your working directory.

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git status
On branch GIT_BRANCH
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   file4.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git commit -m "commit file4 into local repository" file4.txt
warning: LF will be replaced by CRLF in file4.txt.
The file will have its original line endings in your working directory.
[GIT_BRANCH aala36a] commit file4 into local repository
1 file changed, 1 insertion(+)
create mode 100644 file4.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git status
On branch GIT_BRANCH
nothing to commit, working tree clean
```

7.1 GIT\_BRANCH has 4 files, but in master has only 3 files see the difference in below screenshot. But after perform the **git merge** file4.txt merged into master branch.

```

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ \ls -lrt
total 4
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt
-rw-r--r-- 1 vmatheba 1049089 24 Nov  8 20:34 file4.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (GIT_BRANCH)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ \ls -lrt
total 3
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt

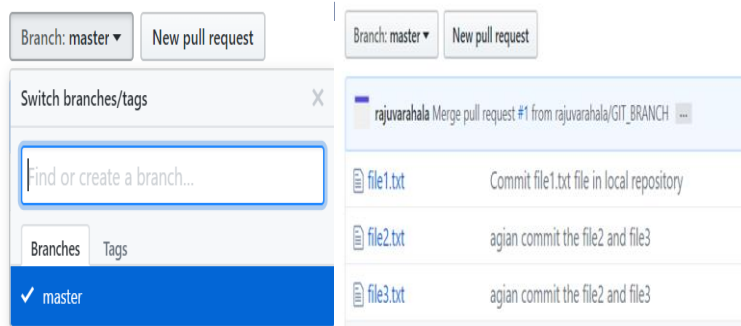
vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ git merge GIT_BRANCH
Updating 9b35e56..a1a36a
Fast-forward
 file4.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt

vmatheba@LIN70000272 MINGW64 ~/Desktop/project_for_cloning/new-project (master)
$ \ls -lrt
total 4
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:27 file3.txt
-rw-r--r-- 1 vmatheba 1049089 26 Nov  8 20:27 file2.txt
-rw-r--r-- 1 vmatheba 1049089 23 Nov  8 20:27 file1.txt
-rw-r--r-- 1 vmatheba 1049089 25 Nov  8 20:38 file4.txt

```

7.2 So all changes are merged in to your local repo and need to push the merged local master upto date changes into remote master using **git push origin master** command, then it will automatically pushed the merged changes into remote master branch.

7.3 Check the merged branch changes are reflected on **github** master branch? As per the below screen not reflected on git hub. We have only 3 files on master branch.



## 7.4 pull request

Branch: master ▾    New pull request

rajuvarahala Merge pull request #1 from rajuvarahala/GIT\_BRANCH ...

file1.txt	Commit file1.txt file in local repository
file2.txt	agian commit the file2 and file3
file3.txt	agian commit the file2 and file3
file4.txt	commit file4 into local repository

praveenkundarapu

## 8. Merge tool installation:-

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd "p4merge.exe \"%BASE\" \"%LOCAL\" \"%REMOTE\" \"%MERGED\""

git config --global diff.tool p4merge

git config --global difftool.p4merge.cmd "p4merge.exe \"%LOCAL\" \"%REMOTE\""

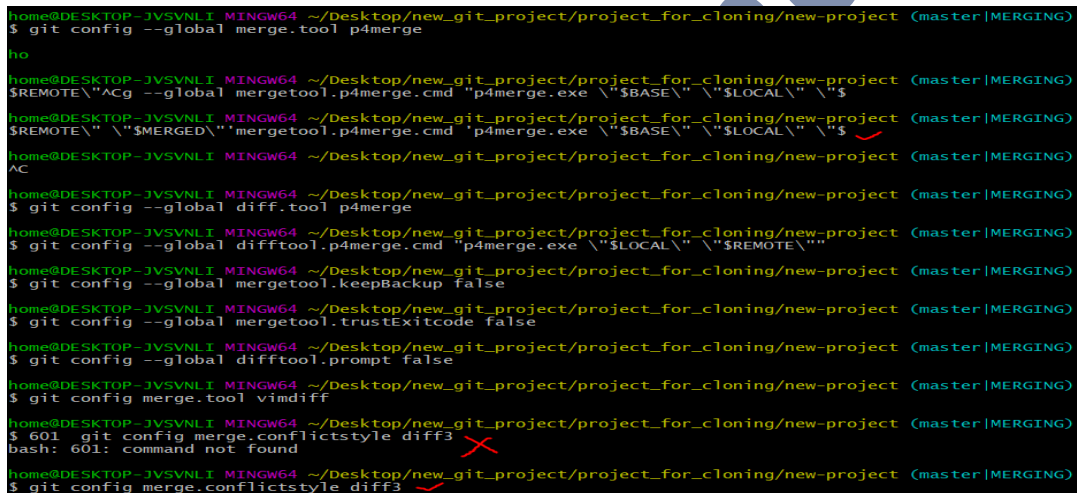
git config --global mergetool.keepBackup false

git config --global mergetool.trustExitcode false

git config --global difftool.prompt false

git config merge.tool vimdiff

git config merge.conflictstyle diff3



```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global merge.tool p4merge
ho
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$REMOTE%"ACg --global mergetool.p4merge.cmd "p4merge.exe \"%BASE\" \"%LOCAL\" \"%
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$REMOTE\" \"%MERGED\"'mergetool.p4merge.cmd 'p4merge.exe \"%BASE\" \"%LOCAL\" \"%$
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
AC
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global diff.tool p4merge
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global difftool.p4merge.cmd "p4merge.exe \"%LOCAL\" \"%REMOTE\"
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global mergetool.keepBackup false
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global mergetool.trustExitcode false
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config --global difftool.prompt false
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config merge.tool vimdiff
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ 601 git config merge.conflictstyle diff3
bash: 601: command not found
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git config merge.conflictstyle diff3
```

8.1 The below screenshots tells that the process of git merge tool. Openup merge window.

Its contain 4 text box

1. local
2. Base
3. Remote
4. Merge display

The image consists of four terminal screenshots showing Git operations:

- Top Screenshot:** Shows the output of `git log` for three branches: `praveen`, `praveenkumar`, and `praveen`. The `praveen` branch has a commit with message "hi this is praveen". The `praveenkumar` branch has a commit with message "hi this is praveenkumar".
- Second Screenshot:** Shows the output of `git log` for the `praveen` branch. The commit message is "hi this is praveen". The terminal also shows the output of `git log` for the `praveenkumar` branch, which has a commit with message "hi this is praveenkumar". Handwritten red text "Local" is visible on the left, and "Remote" is visible on the right.
- Third Screenshot:** Shows the output of `git log` for the `praveen` branch. The commit message is "hi this is praveen". The terminal also shows the output of `git log` for the `praveenkumar` branch, which has a commit with message "hi this is praveenkumar". A red arrow points to the local commit.
- Bottom Screenshot:** Shows the output of `git log` for the `praveen` branch. The commit message is "hi this is praveen". The terminal also shows the output of `git log` for the `praveenkumar` branch, which has a commit with message "hi this is praveenkumar". A red arrow points to the local commit, and a red checkmark is visible on the remote commit.

```

./file2_LOCAL_19916.txt [dos] (08:33 10/11/2018)
<<<<<<< HEAD
hi this is kpk
||||||| merged common ancestors
=====
hi this is kpk praveen
>>>>>>> git_branch

```

```

./file2_LOCAL_19916.txt [dos] (08:33 10/11/2018)
<<<<<<< HEAD
hi this is kpk
||||||| merged common ancestors
hi this is kpk praveen
>>>>>>> git_branch

File2.txt [dos] (08:33 10/11/2018)
:diffg RE

```

8.2 After successful exit from merge tool window, merged modified files displayed. The below screenshot is the result of file2.txt and file3.txt.

```

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git mergetool
Merging:
file2.txt
file3.txt

Normal merge conflict for 'file2.txt':
{local}: modified file
{remote}: modified file
4 files to edit

Normal merge conflict for 'file3.txt':
{local}: modified file
{remote}: modified file
4 files to edit

```

8.3 Then commit modified files.

```

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master|MERGING)
$ git commit -m "commit for merging"
[master d0b96f4] commit for merging

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 9 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ cat file2.txt
hi this is kpk praveen

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ cat file3.txt
<<<<<<< HEAD
<<<<<<< HEAD
hi this is praveen
=====
hi this is praveenkumar
>>>>>>> git_branch
||||||| merged common ancestors
=====
hi this is praveenkumar
>>>>>>> git_branch

```

*merging for base*

## 9. Git stash

Stash stores the unstaged(`git add`) changes into a packages and stores in memory, whenever you want to apply the unchanged changes you can do this.

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git checkout -b git_stash
Switched to a new branch 'git_stash'

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ ls
file1.txt file2.txt file3.txt file4.txt
```

### 9.1 Create a stash with command using `git stash`

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ vi file1.txt

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git stash
Saved working directory and index state WIP on git_stash: d0b96f4 commit for merging

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ |
```

9.2. Check `git status`, it displays your `git_stash` branch has been clean and there is no files on your local repo to commit( it means git stash creates separate virtual space for your changes and placed into stash repository). If you want to revert back the previous changes into `git_stash` branch use below command `git stash pop`.

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git status
On branch git_stash
nothing to commit, working tree clean

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git stash pop ✓
On branch git_stash
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:   file1.txt ✓

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (8b36ff5da2b2322270068a530a1ed15337174692)
```

9.3 How to see the list of stash files using `git stash list` command, it will show the all stashed files with refs id's and if you want to delete particular file using below `git stash drop refs id`.

How to delete the Stash: - `git stash drop stash@{0}` command.



## 10. Git tags

In git tags are read only, Snapshots of particular version number having a name and meant for read only.

10.1 Create a lightweight tag then check the version with git show tag command and list all tag with git tag command.

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git tag tag_1.0

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git tag
tag_1.0

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git show tag_1.0
commit d0b96f45c7d97a245d4e13928eee982b5bc50401 (HEAD -> git_stash, tag: tag_1.0, master)
Merge: e79bf08 53a7123
Author: praveen <kpk424@gmail.com>
Date: Sat Nov 10 08:46:04 2018 +0530

    commit for merging

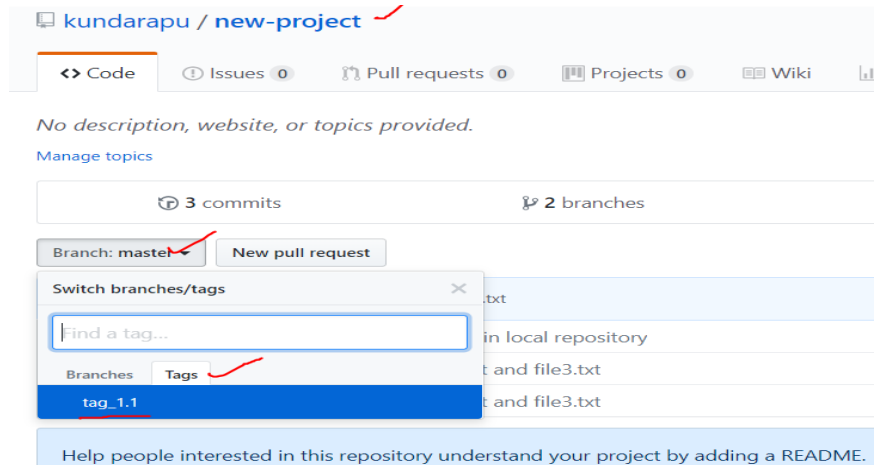
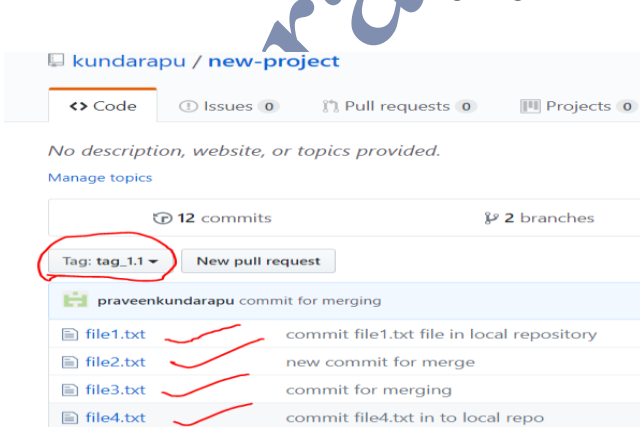
diff --cc file3.txt
index aee8665..07b04ca..7aa0d06
--- a/file3.txt
+++ b/file3.txt
@@@ -1,5 -1,1 +1,10 @@@
+<<<<<< HEAD
+<<<<<< HEAD
+hi this is praveen
+=====
+hi this is praveenkumar
+>>>>>> git_branch
+||||||| merged common ancestors
+=====
+hi this is praveenkumar
+>>>>>> git_branch
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_stash)
$ git tag
tag_1.0
```

10.2 Create an annotated tag with some messages. And push to origin (github).

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git tag -a tag_1.1 -m "tag commit"

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git push origin tag_1.1
Enumerating objects: 27, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (20/20), done.
Writing objects: 100% (24/24), 2.15 KiB | 157.00 KiB/s, done.
Total 24 (delta 7), reused 0 (delta 0)
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/kundarapu/new-project.git
* [new tag]         tag_1.1 -> tag_1.1
```

10.3 Then check tags in git hub to follow below screenshots



## 10.4 List all tags

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git tag
tag_1.0
tag_1.1
```

## 10.5 Checkout tags

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project ((tag_1.1))
$ git checkout tag_1.0
HEAD is now at d0b96f4 commit for merging
M       file1.txt

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project ((tag_1.1))
$ ls
file1.txt file2.txt file3.txt file4.txt

home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project ((tag_1.1))
$ |
```

## 11. Git pull/fetch

**11.1 Git pull:** Fetches files from remote repository and integrates with local repository

Note: git pull fetches and merges the remote repository to your local repository, whereas git fetch only stages the files into local repository index.

Enter git pull command as shown below.

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (master)
$ git pull origin git_branch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/kundarapu/new-project
* branch                git_branch -> FETCH_HEAD
  f39506c..700e735  git_branch -> origin/git_branch
Merge made by the 'recursive' strategy.
 file4.txt | 1 +
 1 file changed, 1 insertion(+)
```

Below merge window contains details of commits that need to merge. Save and quit

```
Merge branch 'git_branch' of https://github.com/kundarapu/new-project

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
```

**11.2 git fetch:** This stages the files of remote repository in the local repository.

Enter the below command and check the result then compare with git pull result.

```
home@DESKTOP-JVSVNLI MINGW64 ~/Desktop/new_git_project/project_for_cloning/new-project (git_branch)
$ git fetch origin git_branch
From https://github.com/kundarapu/new-project
* branch                git_branch -> FETCH_HEAD
```

## 12. Git commands:

- Git config –list(all env variable of git)
- Git config –global user.name Praveen(set the user name as praveen)
- Git config –global user.email Praveen@gmail.com(set the user email)
- Git config –global push.default simple
- Git clone
- Git add
- Git commit –m “message”
- Git commit –a –m “”(staging and commit)
- Git push(send the commit to remote repo)
- Git status(status of working tree)
- Git branch(on which branch you are in)
- Git pull(get the changes from remote repo)
- Git mergetool(this will open your merge tool)
  - git config --global merge.tool p4merge(mergetool setup first command)
  - REMOTE\ " \"\$MERGED\ "mergetool.p4merge.cmd 'p4merge.exe \" \$BASE\ "  
\" \$LOCAL\ " \"\$( mergetool setup second command)
- Git rm(delete the file in a working tree and to reflect the same in server remote repo you need to commit and push)
- Git log (history of your repo)
- Git log file name
- Git log –author=name
- Git shortlog
- Git log –online
- Git checkout (change set)
- Git diff (to check the difference between staging area and git)
- Git show (show changes of patchset)
- Git clean –n
- Git clean –f
- Git blame file1(information about each line modification done by author)
- Git fetch/pull (track remote branching into local working tree)
- Git pull(performs a git fetch and git merge)
- Git merge(git branches and integrate them into a single branch.)
- Git reset (used to remove commits from the current branch.)
- Git branch(list all branches)
- Git stash(Stash stores the unstaged(git add) changes)
- Git tag(Snapshots of particular version number)