Practical 1 -: To perform Version Control using GIT

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

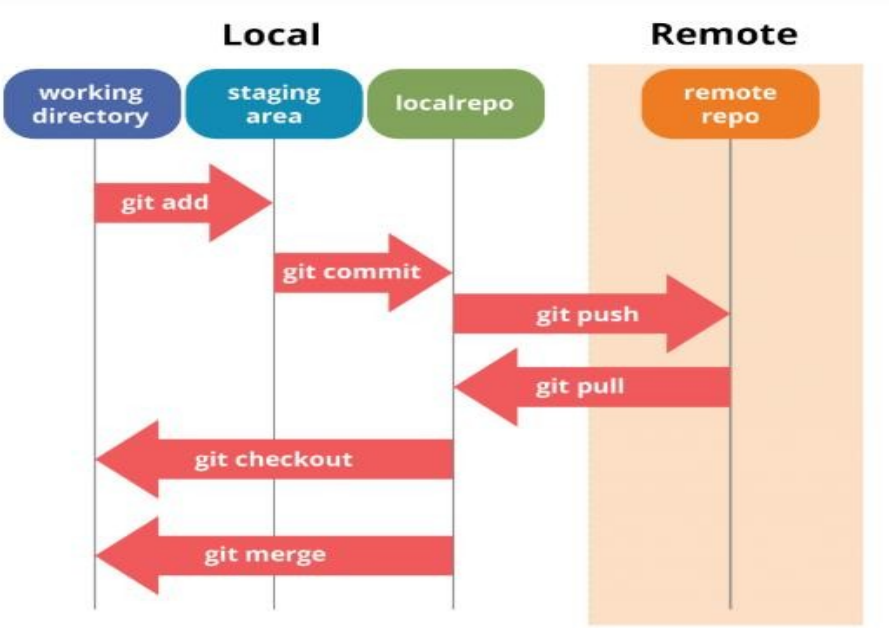
Some of the basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Some advanced Git operations are:

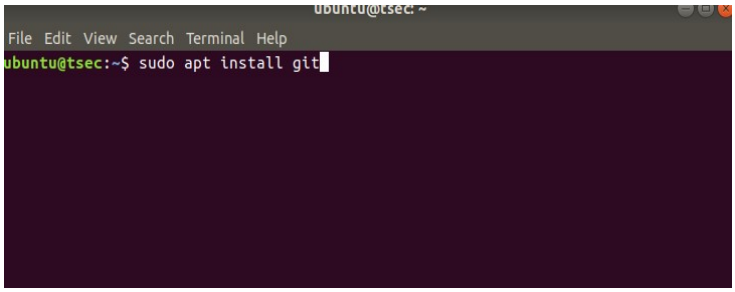
1. Branching
2. Merging
3. Rebasing

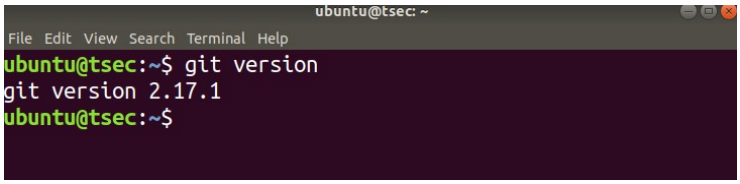
The following diagram depict the all supported operations in GIT



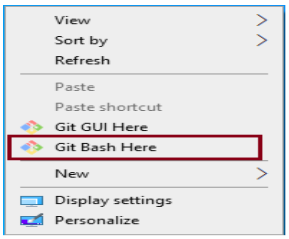
Installation of GIT

1) In windows, download GIT from https://git-scm.com/ and perform the straightforward installation.
2) In Ubuntu, install GIT using $sudo apt install git, Confirm the version after installation using command $git ¯version

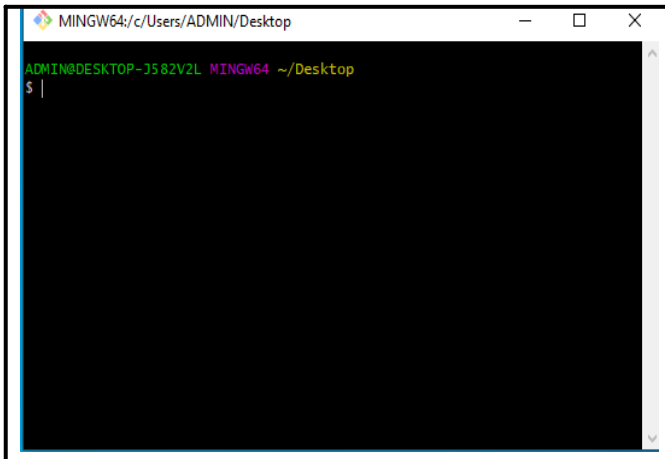




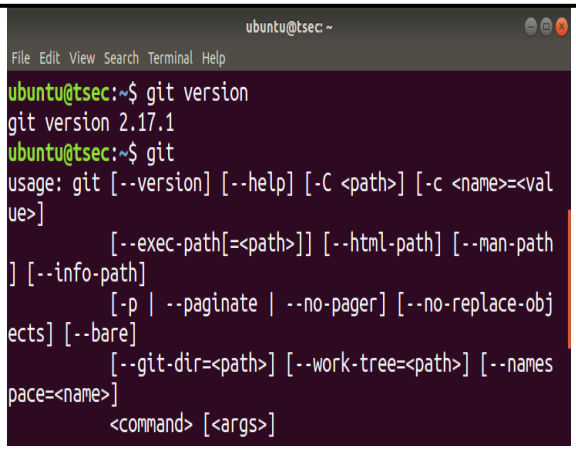
Once installation is done, open the terminal in Ubuntu and perform the following steps or in windows Right click and select Git bash here.



The output of GIT Bash in windows and GIT shell in Ubuntu is shown below



GIT Bash in Windows



GIT Shell in Ubuntu

To perform version control, let us create a directory dvcs (Distributed version control system) and change directory to dvcs.

$ mkdir git-dvcs

$ cd git-dvcs/

Now check the user information using

```
$ git config –global
```

As there are no users defined, let us define it using following two commands

```
$ git config --global user.name "bhushan"
$ git config --global user.email "bhushan,jadhav1@gmail.com "
```

Now, check the list of users

```
$ git config --global --list
user.name=bhushan
user.email=bhushan.jadhav1@gmail.com
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop
$ mkdir git-dvcs

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop
$ cd git-dvcs/

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs
$ git config --global --list
user.name=bhushan
user.email=bhushan.jadhav1@gmail.com

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs
$ cat ~/.gitconfig
[user]
        name = bhushan
        email = bhushan,jadhav1@gmail.com
```

Let us create a repository for version control named ”git-demo-project”

```
$ mkdir git-demo-project
$ cd git-demo-project/
```

Now, initialize the repository using following command

```
$ git init
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project
$ git init
Initialized empty Git repository in C:/Users/ADMIN/Desktop/git-dvcs/git-demo-project/.git/
```

The output of above command shown below which adds .git hidden directory in current repository.

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/
```

If you have existing repository, then simply delete .git file and reinitialize it.

$ rm -rf .git/

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project
$ ls -al
total 0
drwxr-xr-x 1 ADMIN 197121 0 Jan  1 18:19 ./
drwxr-xr-x 1 ADMIN 197121 0 Jan  1 18:17 ../
```

$ git init
Initialized empty Git repository in C:/Users/ADMIN/Desktop/git-dvcs/git-demo-project/.git/

Now, let us add some files inside our repository "git-demo-project"

To add files in the repository by create or copy some doc,html,image files inside current directory to see index and staging area. The add command is used along with dot (. Dot means current directory) for adding files in current repository i.e. making them in staging mode. They are untracked until we commit them .

$ git add .
Index and staging area
To check the status of repository, use
$ git status

Which will show you some untrack files, so untracks files can be tracked using commit command.

Now, let us commit the changes
$ git commit -m "First Commit" (# here -m for message)

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -m "First Commit"
[master (root-commit) 50148fb] First Commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Gitpracts.docx
 create mode 100644 Installation and Configuration of GIT.docx
```

Add index.html in our directory

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -m "First Commit"
On branch master
Untracked files:
        index.html

nothing added to commit but untracked files present
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add .

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -am "express Commit"
[master 97d0a76] express Commit
 1 file changed, 9 insertions(+)
 create mode 100644 index.html
```

$ git add .
$ git commit -am "express Commit" (# Here -a used for express commit)
$ nano index.html

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

$ touch teststatus

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

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

no changes added to commit (use "git add" and/or "git commit -a")
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git checkout -- index.html
```

Changes are Discarded by checkout

(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add index.html

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

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

$ git add index.html
$ git add teststatus

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

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


ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add teststatus

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
        new file:   teststatus
```

$ git commit -am "Express commit"

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -am "Express commit"
[master d3a6a76] Express commit
 2 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 teststatus

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Now let us see history of commits. The log command is used for seeing the commit history.

$ git log

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log
commit d3a6a763ff5a1fa33e16686d8d6c83ee8489843b (HEAD -> master)
Author: bhushan <bhushan,jadhav1@gmail.com>
Date:   Wed Jan 1 18:44:36 2020 +0530

    Express commit

commit 97d0a7681d218e1f45dd753c381254d2fa36141d
Author: bhushan <bhushan,jadhav1@gmail.com>
Date:   Wed Jan 1 18:32:44 2020 +0530

    express Commit

commit 50148fb629e12e29eaee04277be7a97afdbdd824
Author: bhushan <bhushan,jadhav1@gmail.com>
Date:   Wed Jan 1 18:26:02 2020 +0530

    First Commit
```

To see all the operation in oneline use the ⁻-oneline option in log command

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
d3a6a76 (HEAD -> master) Express commit
97d0a76 express Commit
50148fb First Commit
```

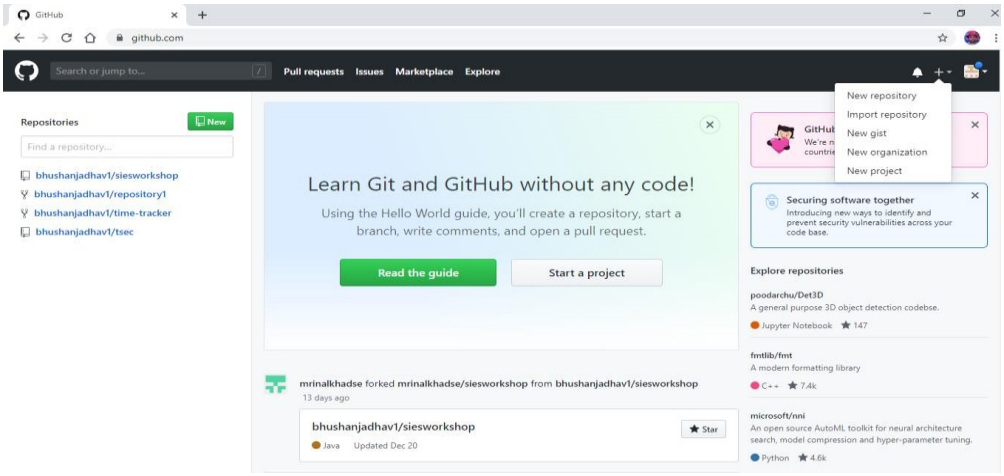
⁻-oneline option for particular file in log command

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline teststatus
d3a6a76 (HEAD -> master) Express commit

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
d3a6a76 (HEAD -> master) Express commit
97d0a76 express Commit
50148fb First Commit

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline 97d0a76..d3a6a76
d3a6a76 (HEAD -> master) Express commit

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline -n 2
d3a6a76 (HEAD -> master) Express commit
97d0a76 express Commit
```

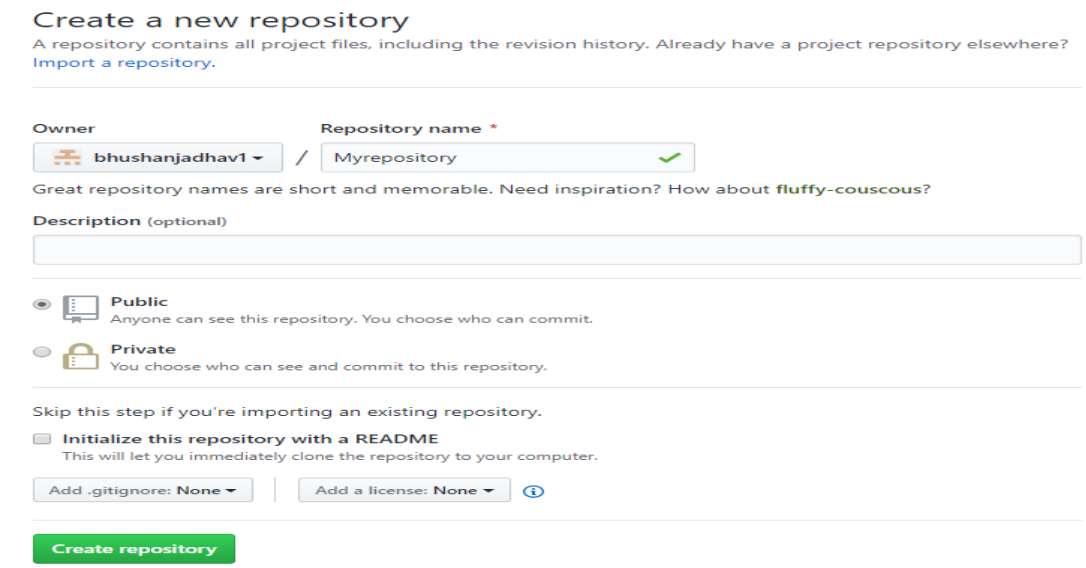
Example 2: Performing Version control in GITHUB with Pull and Push commands

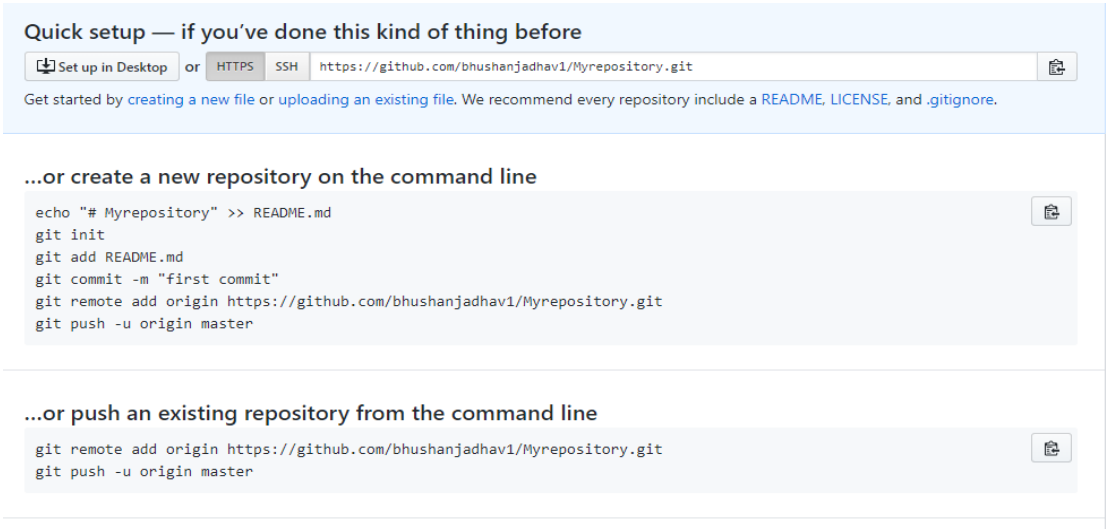
First open Github.com and create a new account.After verifying account through E-mail, create a Repository on github.com.

Open github.com→ create an account→After login Select New repository from the menu.



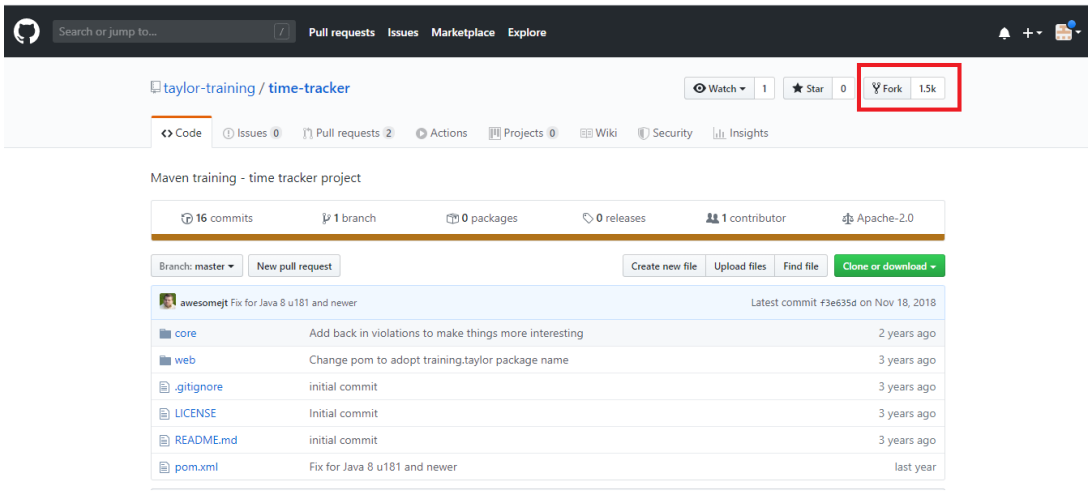
Now Specify a Name to repository and select public option followed by create repository



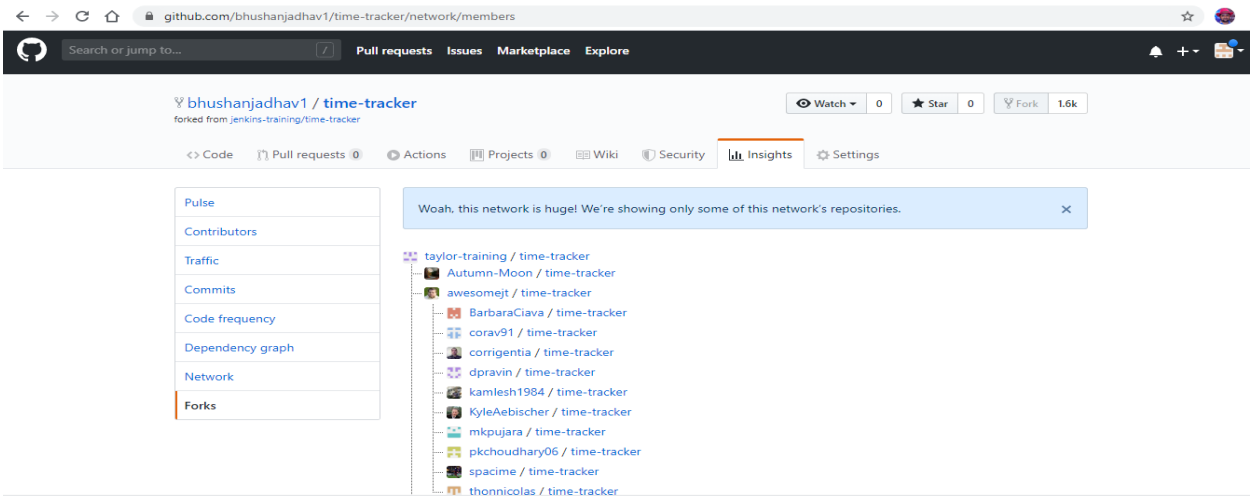


By default, we can create public repository in Github. So we can copy the entire public repository of any other users in to own account using "FORK" Operation. Now fork the repository (Sharing with other users who wants to contribute).

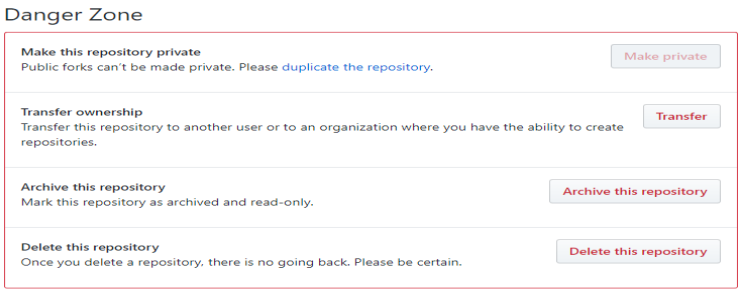
Login with another account →Copy and Paste URL of repository →then just click on fork to clone to others account. Suppose we want to fork public repository "timetracker". So search for "timetracker" github repository on google and once its opened clicked on "Fork button" from the top of the github web page as shown below.

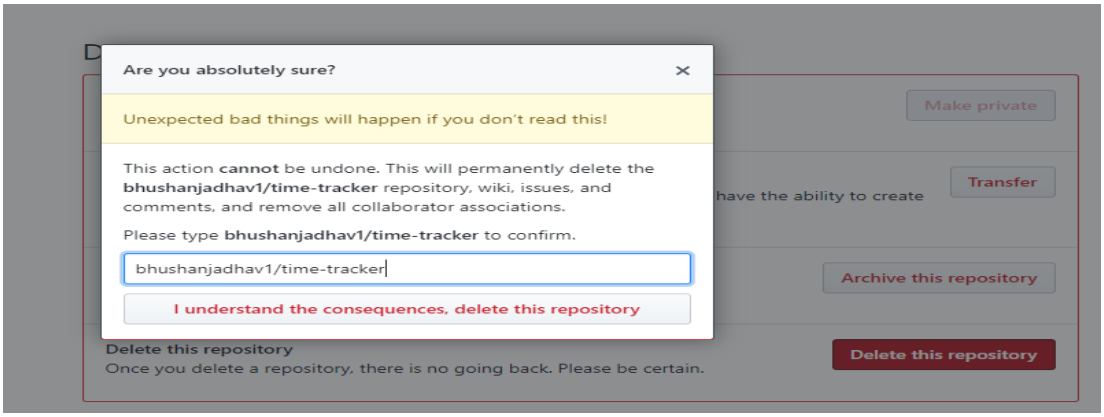


After fork it will be added in your local repository.

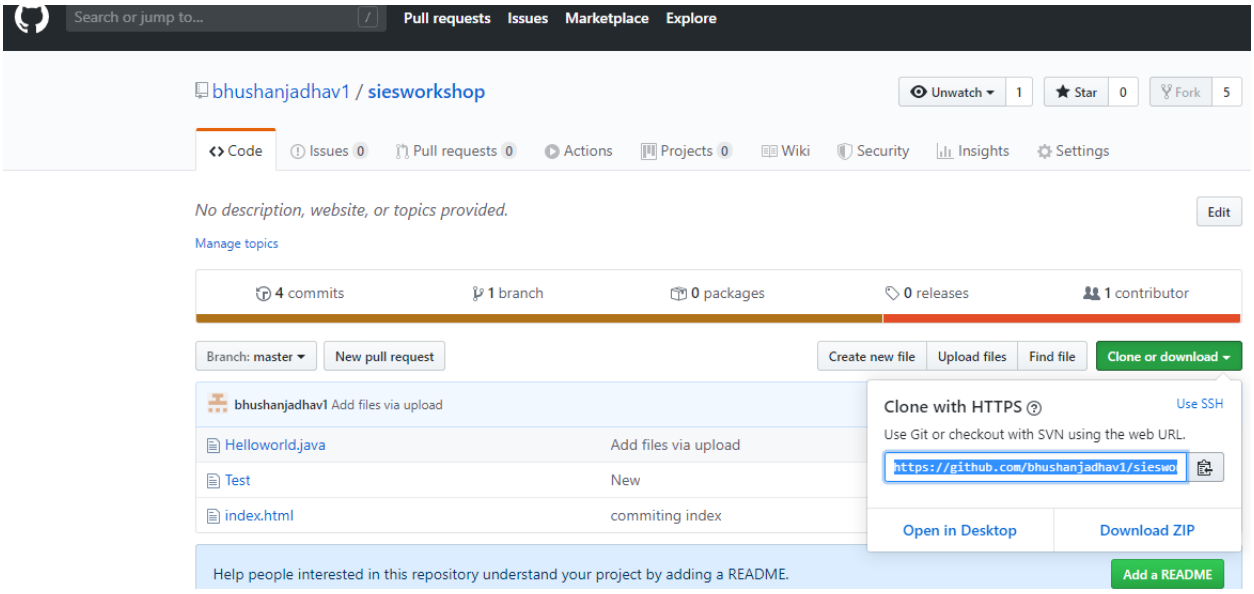


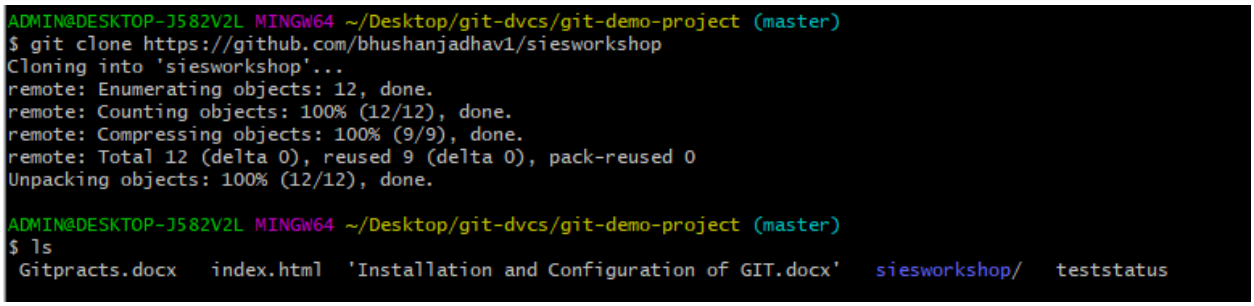
To delete the repository, open the desired repository you want to delete and go to the settings option. There you will see delete repository button to delete it.

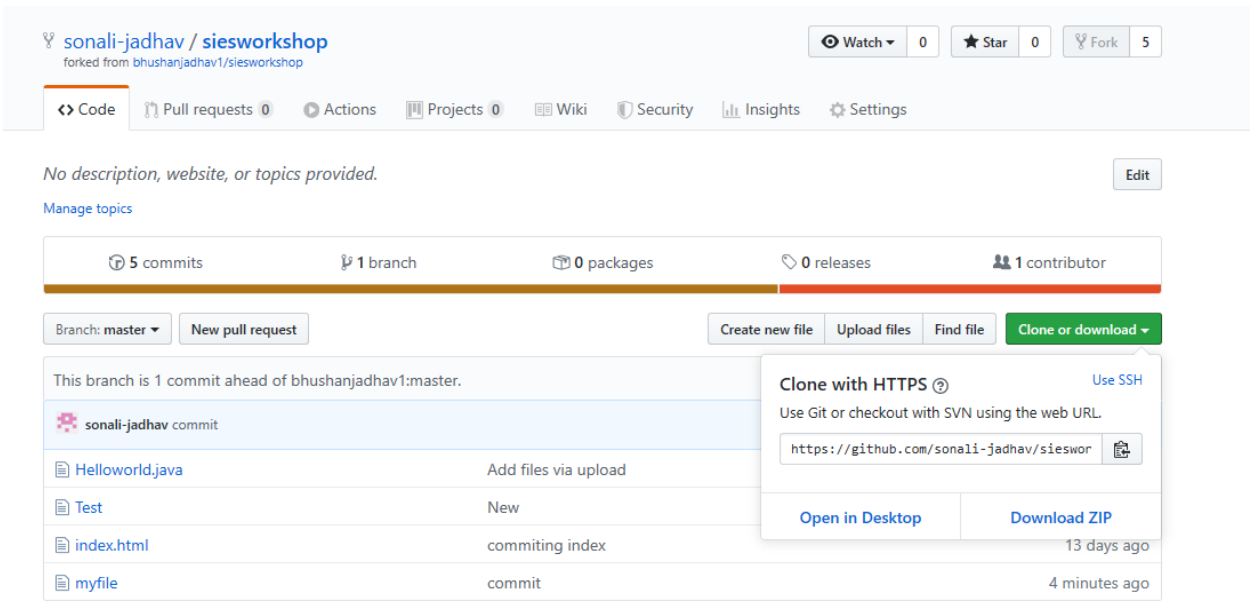




Now, if you want to download a repository in local machine, then git clone command is used followed by path to repository. In GitHub the path of repository can be known through clone or download button and it can be downloaded using git clone command as shown below.







Pull and Push Processes

The pull command used to fetch the repository from github to local while push is used to commit from local repository to Github.

Push→ Push changes to Web repository

Pull→ Pull changes to Local repository

The following commands are used for pull and push repositories

A) Push command

```
$ git remote add origin https://github.com/bhushanjadhav1/siesworkshop.git
$ git remote show origin
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/bhushanjadhav1/siesworkshop.git
  Push  URL: https://github.com/bhushanjadhav1/siesworkshop.git
  HEAD branch: master
  Remote branch:
    master new (next fetch will store in remotes/origin)
  Local ref configured for 'git push':
    master pushes to master (local out of date)
```

If you add remote again then will show you fatal error.

$ git remote add origin https://github.com/bhushanjadhav1/Myrepository.git

fatal: remote origin already exists.

So, to delete origin rm origin command is used

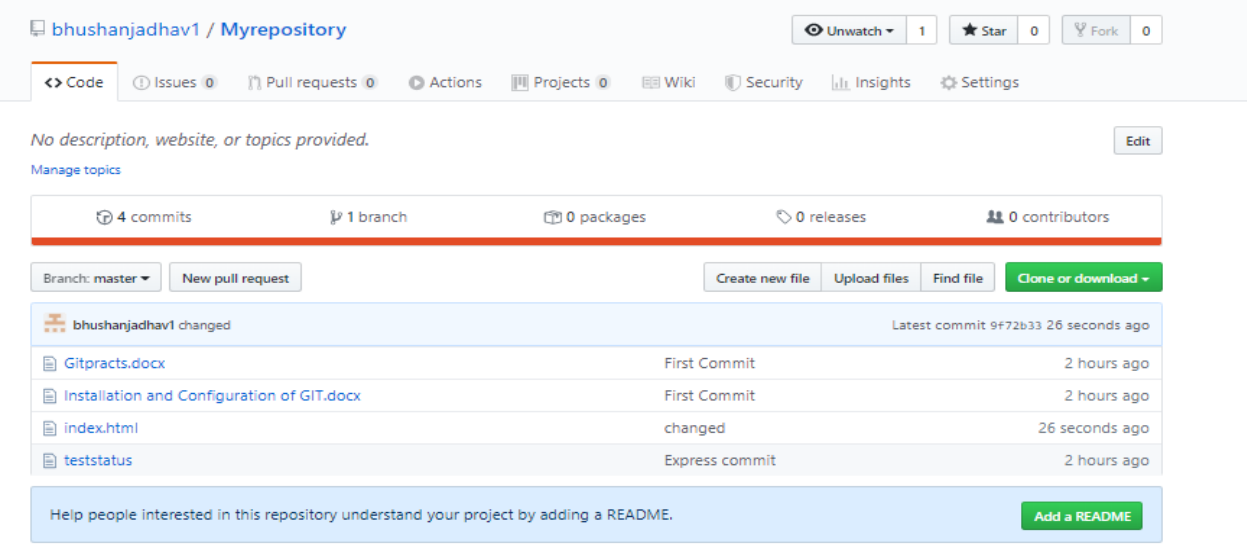
$ git remote rm origin

Now, to push the local repository to remote github following command is used

$ git push -u origin master

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git push -u origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 770.93 KiB | 10.56 MiB/s, done.
Total 11 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/bhushanjadhav1/Myrepository.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Now you can check the github for updated contents.



B) Pull Changes

Pull command is used to download the remote updated repository into local one. The command for download is

$ git pull

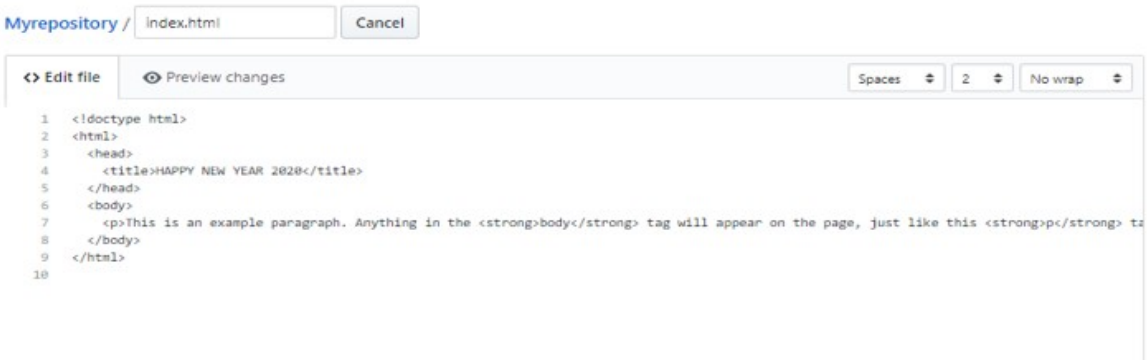
```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/bhushanjadhav1/Myrepository
   d3a6a76..9f72b33  master     -> origin/master
Updating d3a6a76..9f72b33
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Now you can see the changes in local repository using git log.

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline origin/master
9f72b33 (origin/master) changed
d3a6a76 Express commit
97d0a76 express Commit
50148fb First Commit
```

C) Fetch

Suppose you have a file in github and you have changes that.



ORIGINAL FILE

Changed File

Now we use fetch command to fetch the changes, which will show you both the files like original and changed in local repository.

$ git fetch

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/bhushanjadhav1/Myrepository
   9f72b33..21e9ada  master     -> origin/master
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline origin/master
21e9ada (origin/master) Fetch
9f72b33 changed
d3a6a76 Express commit
97d0a76 express Commit
50148fb First Commit
```

Here fetch will not show you like updated changes file as like push. So use merge command to merge the changes so use following command for merge.

$ git merge origin/master

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ cat index.html
<!doctype html>
<html>
  <head>
    <title>FETCH....HAPPY NEW YEAR 2020</title>
  </head>
  <body>
    <p>This is an example paragraph. Anything in the <strong>body</strong> tag will appear on the page, just like this <strong>p</strong> tag and its contents.</p>
  </body>
</html>

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git merge
Merge made by the 'recursive' strategy.
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ cat index.html
<!doctype html>
<html>
  <head>
    <title>HAPPY NEW YEAR 2020</title>
  </head>
  <body>
    <p>This is an example paragraph. Anything in the <strong>body</strong> tag will appear on the page, just like this <strong>p</strong> tag and its contents.</p>
  </body>
</html>
```