What is Git

Git is popular distributed version control system that is used to manage project.

Version Control System

Version Control System (VCS) is software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS −

- Allows developers to work simultaneously.
- Does not allow overwriting each other's changes.
- Maintains a history of every version.

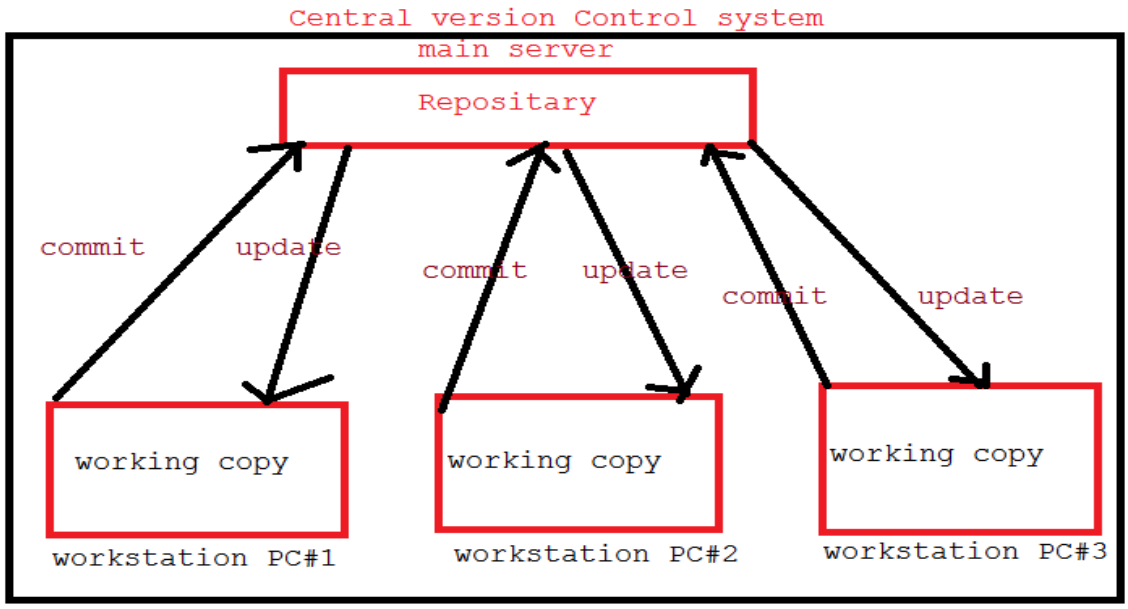
There are two types of VCS:

1) Centralized version control system(CVCS)

2) Distributed version control system(DVCS)

Centralized VCS:

In CVCS, the central server stores all the data. It enables team collaboration. It just contains a single repository, and each user gets their working copy. We need to commit, so the changes get reflected in the repository. Others can check our changes by updating their local copy.

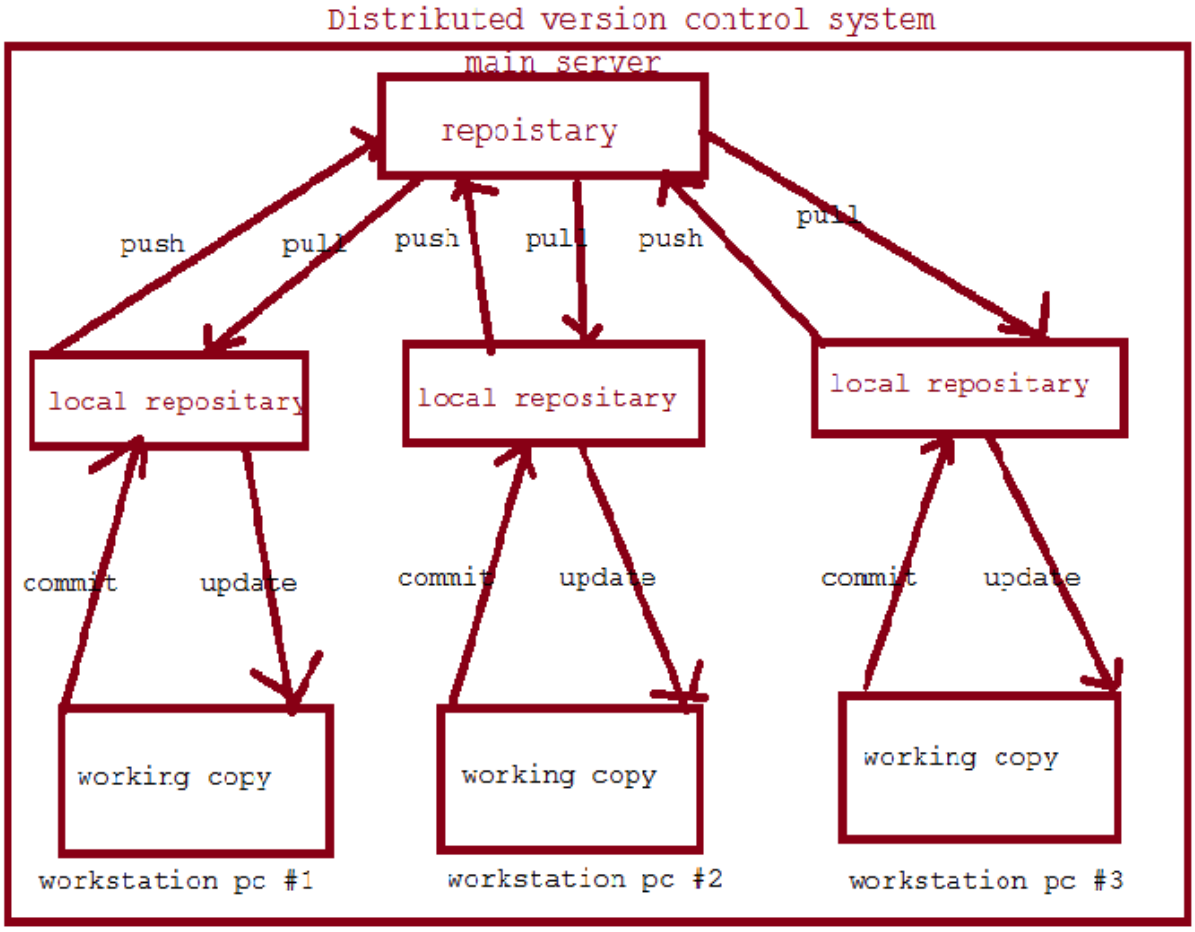


- It is not locally available, which means we must connect to the network to perform operations.
- During the operations, if the central server gets crashed, there is a high chance of losing the data.
- For every command, CVCS connects the central server which impacts speed of operation

The Distributed Version Control System is developed to overcome all these issues.

Distributed Version Control System (DVCS) :

In DVCS, there is no need to store the entire data on our local repository. Instead, we can have a clone of the remote repository to the local. We can also have a full snapshot of the project history.



The User needs to update for the changes to be reflected in the local repository. Then the user can push the changes to the central repository. If other users want to

check the changes, they will pull the updated central repository to their local repository, and then they update in their local copy.

Repository is a storage where all the developer code will be stored
Repositary is the storage to integrate the code.
Once development completed the every developer should push the code to repositary.
Repositary will track all the changes happening to files

What is difference between git and github

	Git	GitHub
1.	Git is a software.	GitHub is a service.
2.	Git is a command-line tool	GitHub is a graphical user interface
3.	Git is installed locally on the system	GitHub is hosted on the web
4.	Git is maintained by linux.	GitHub is maintained by Microsoft.
5.	Git is focused on version control and code sharing.	GitHub is focused on centralized source code hosting.
6.	Git is a version control system to manage source code history.	GitHub is a hosting service for Git repositories.
7.	Git was first released in 2005.	GitHub was launched in 2008.

Environment setup:

Step1)Create Account in github site

www.github.com

step2) download and install git client software

it is used to perform git commond operation with git repo

git bash we will execute git command

go to any folder ->right click on mouse->select git bash here ->open

https://git-scm.com/downloads

Git for Windows stand-alone installer

Download the latest **Git** for Windows installer.

When you've successfully started the installer, you should see the Git Setup wizard screen. Follow

the Next and Finish prompts to complete the installation. The default options are pretty sensible for most users.

After installing git client,go to any folder and right click on mouse then it should display Git GUI here and Git BASH here

Click on GITBASH here

Run the following commands to configure your Git username and email using the following commands, replacing Emma's name with your own. These details will be associated with any commits that you create:

$ git config --global user.name " abc"

$ git config --global user.email abc@gmail.com

Command :

git init:-

→ git command are used for initializing git folder

git remote add origin 'repo-url':

git remote command is used to configure git central repo url.

Ex:

git pull origin master :-

pull command is used to taking the latest changes from central repo to local repo.

git status :-

current branch and changed files and updated files and new files

git add:-

git add command are used for adding files to staging area.

1. to add only one file provide specific path

2. to add all changes for commit provide (.)

git add abc.txt (for adding a single file)

git add . (to add all changes for commit)

git commit –m 'msg' :-

Commit is used for committing files to local repository.

git push -u origin :-

push command is used to push files from local repo to central repo.

git push origin (branchname) for push the branch and changes

Git clone:

Git clone means taking whole project from central repo to local repo.

git checkout .

git checkout -b (branchname) to create new branch

git checkout (branchname) to change branch

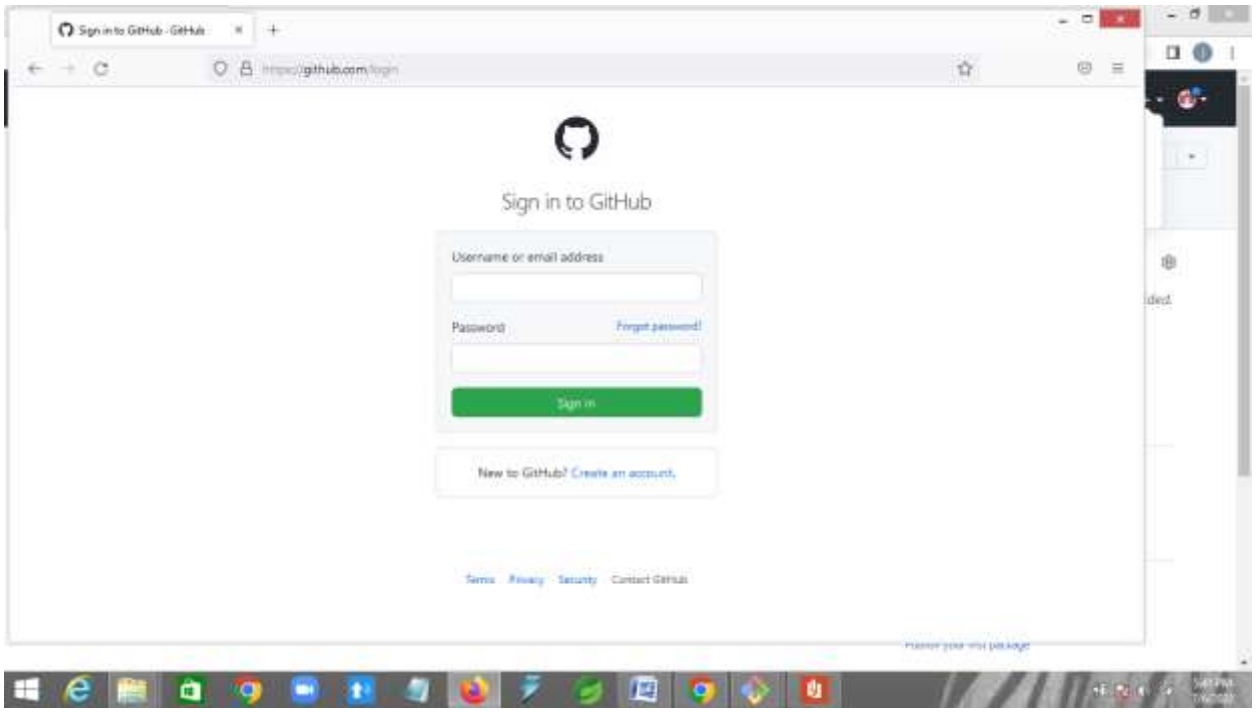
git merge <barnchName> :

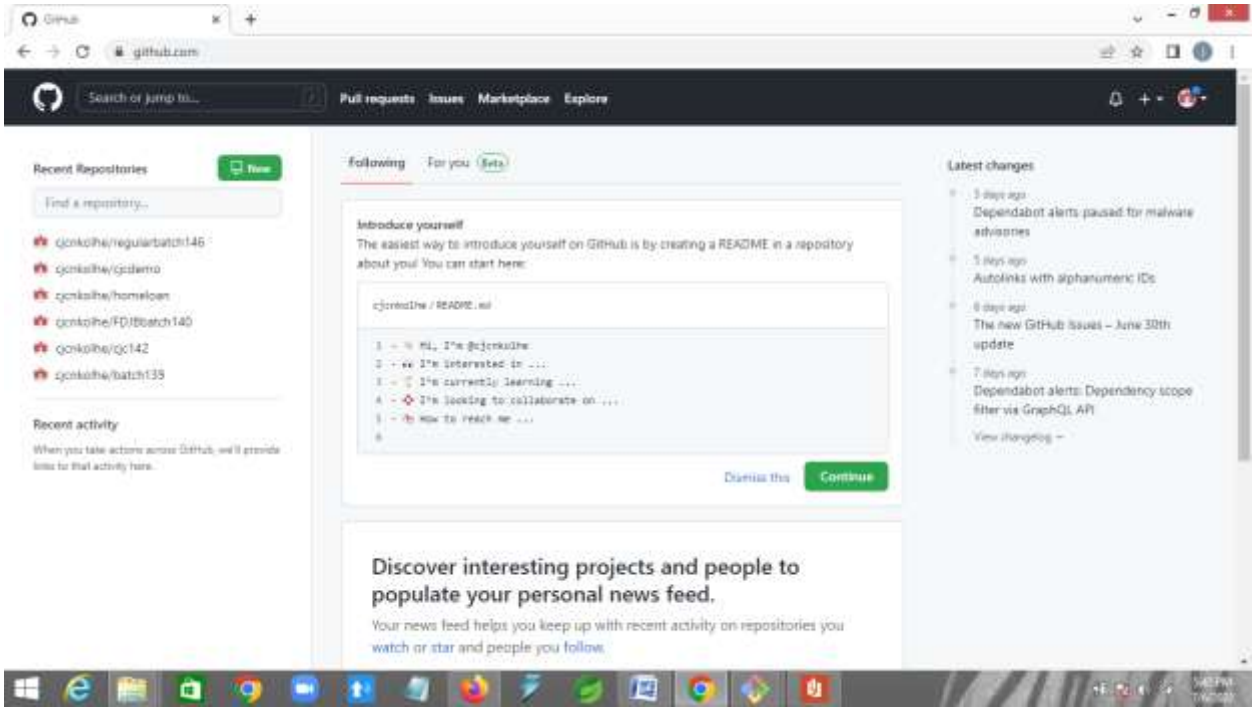
The git merge command is used to merge the branches.

git log:

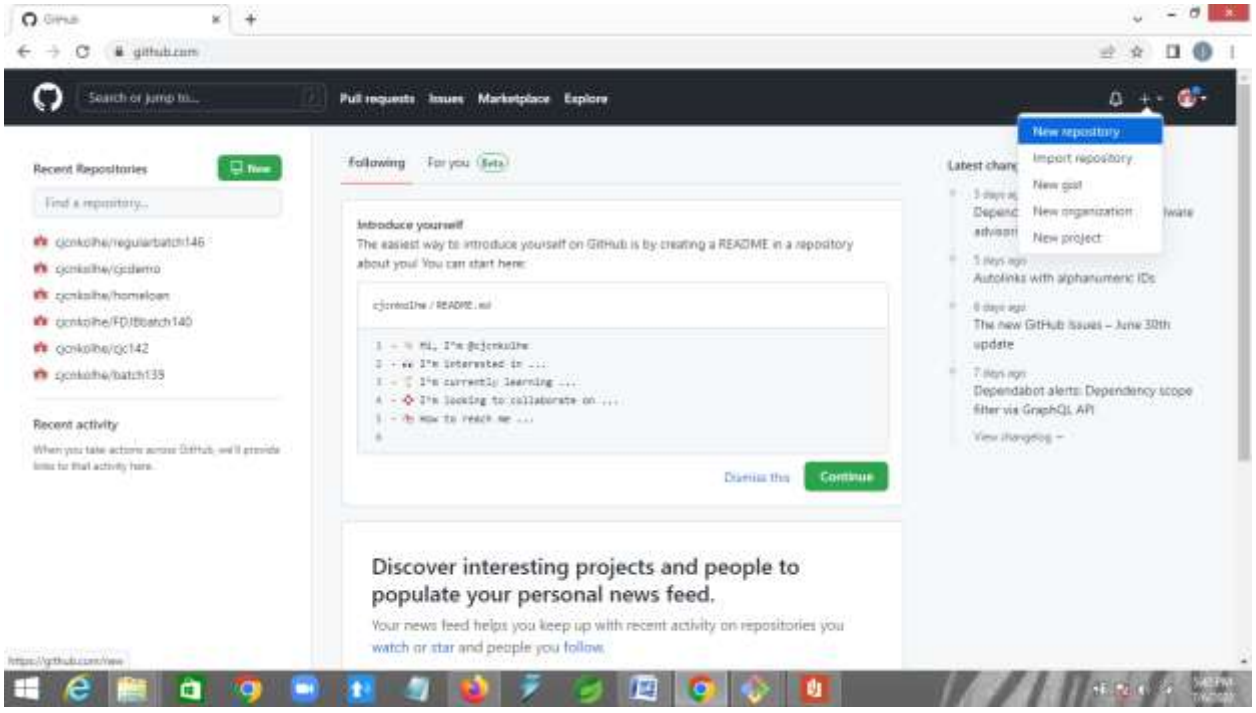
log command used to display commit history.

Create Account in github.com

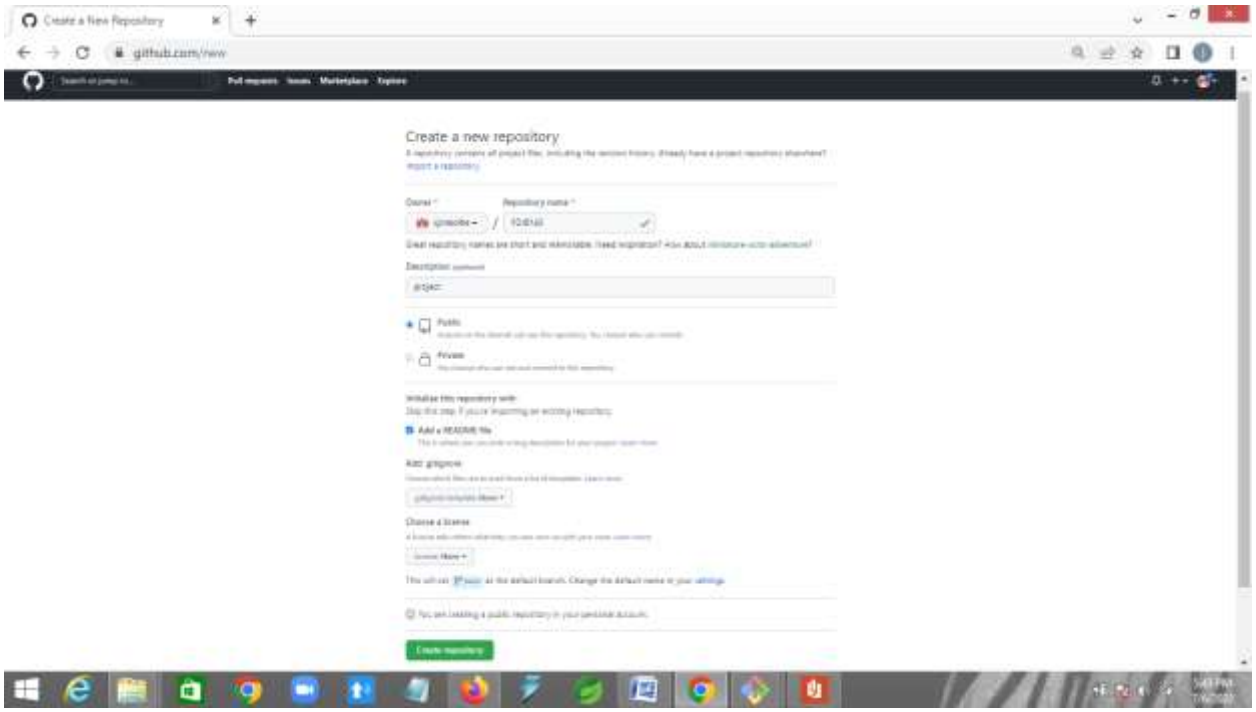


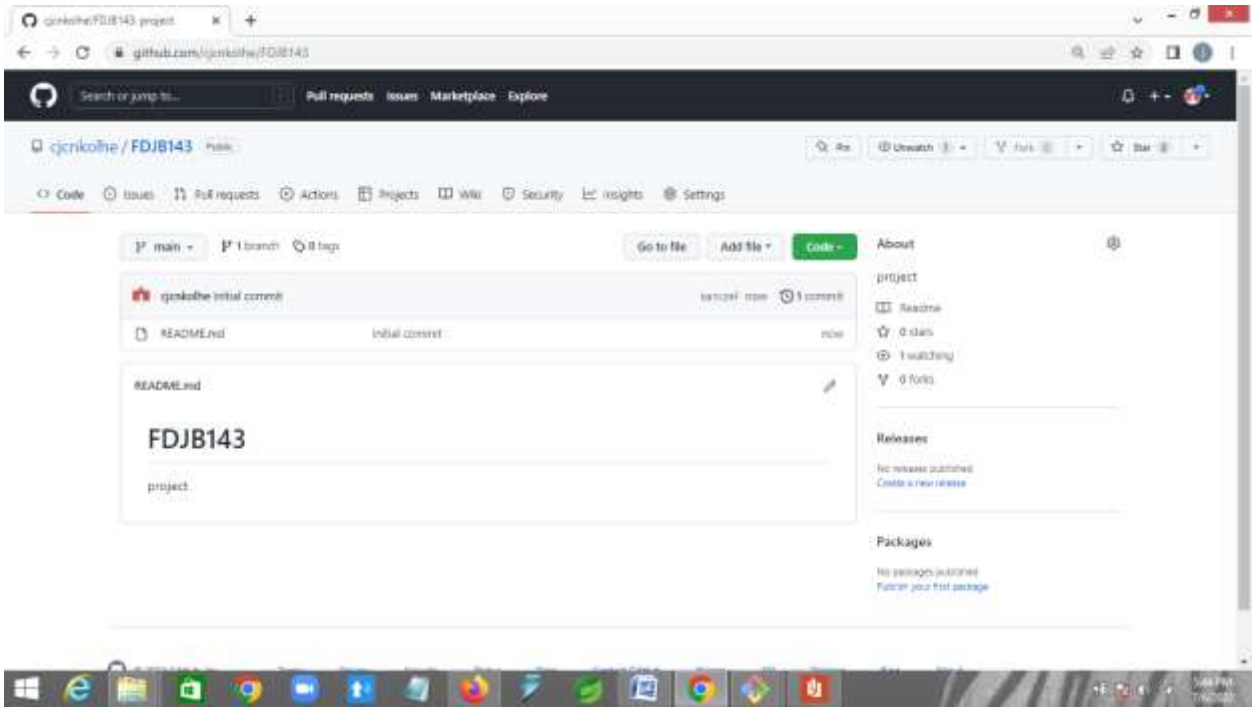


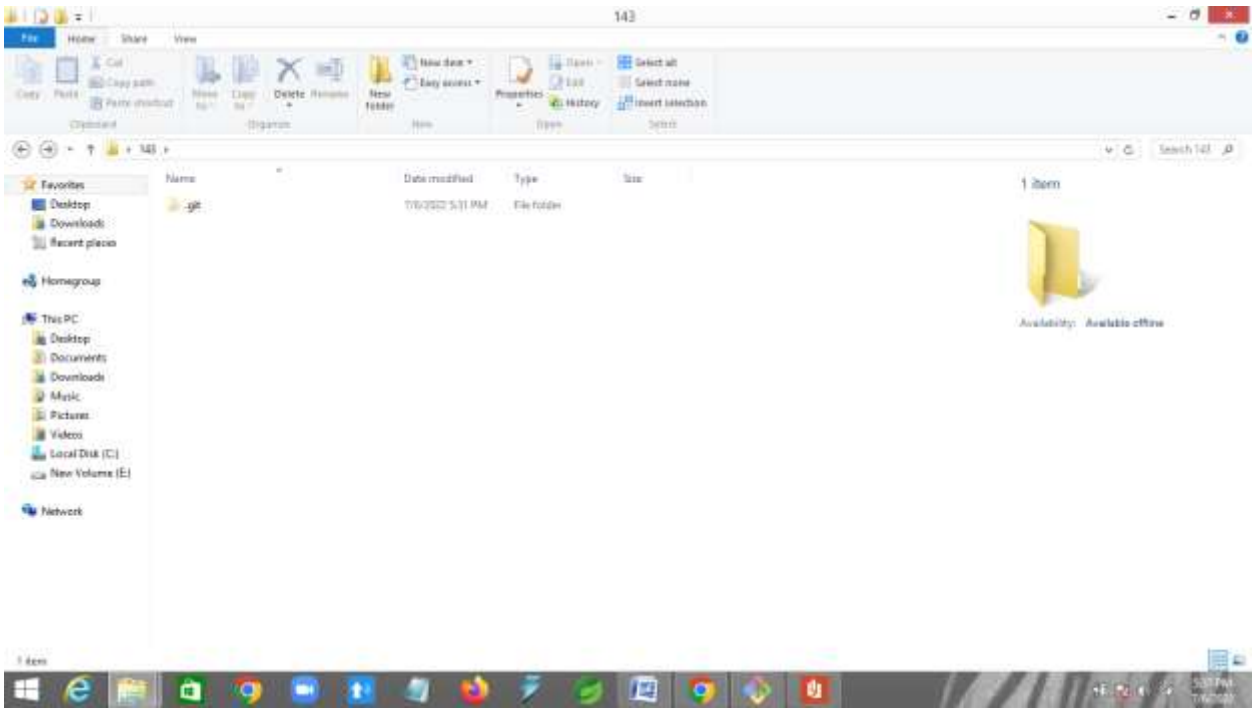
Step 1: create git repositary

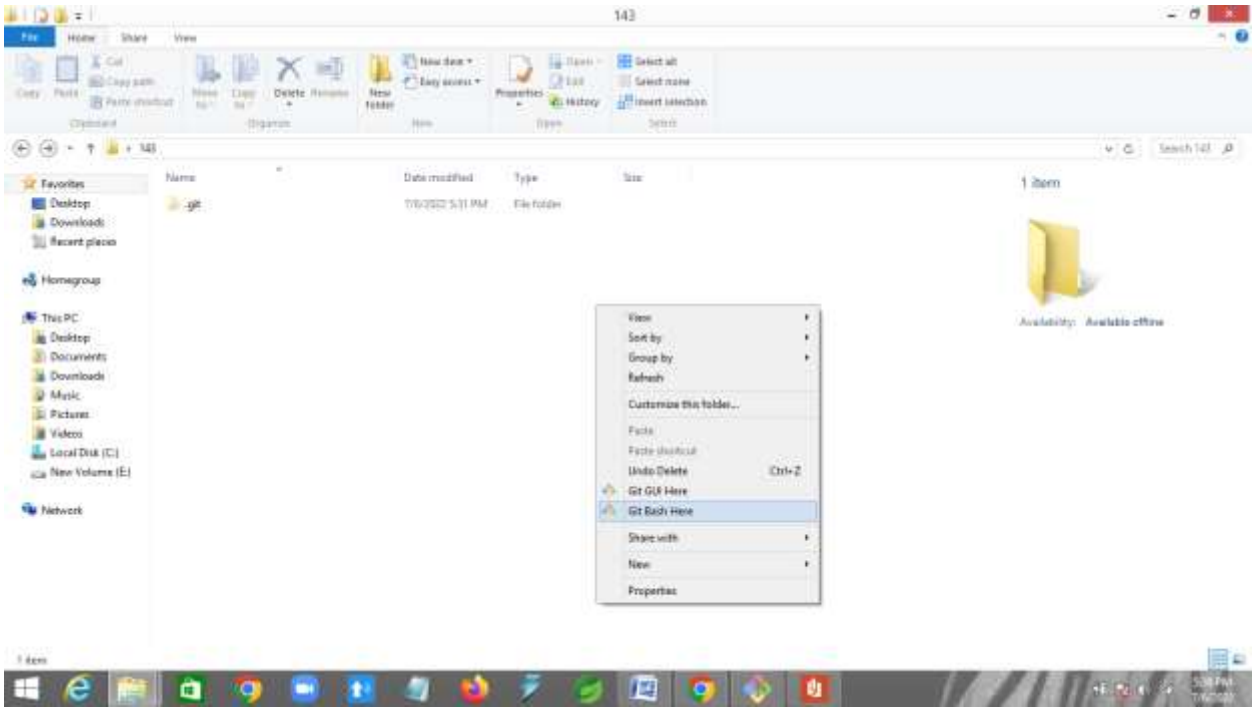


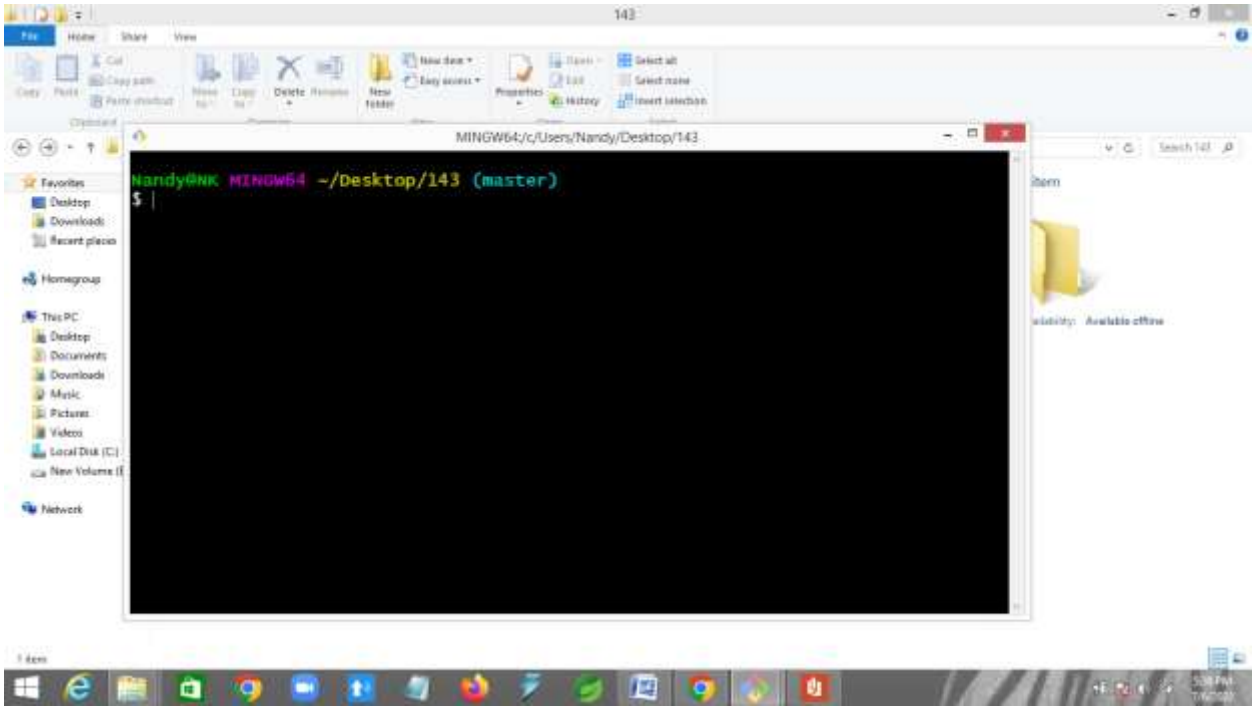
Step 2: write repositary name




cjcnkolhe / FDJB143
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
Go to file
Add file
Code
About
project
README.md
FDJB143
project
Releases
Packages


143
File
Home
Share
View
Favorites
Desktop
Downloads
Recent places
Homegroup
This PC
Documents
Music
Pictures
Videos
Network
.git
1 item


143
Git GUI Here
Git Bash Here


143
MINGW64:/c/Users/Nandy/Desktop/143
Nandy@NK MINGW64 ~/Desktop/143 (master)
$

Commands:

```
Nandy@NK MINGW64 ~/Desktop/143
$ git init
Initialized empty Git repository in
C:/Users/Nandy/Desktop/143/.git/

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git remote add origin
"https://github.com/cjcnkolhe/regularbatch146.git"

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git pull origin master
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 6 (delta 0), pack-
reused 0
Unpacking objects: 100% (9/9), done.
From https://github.com/cjcnkolhe/regularbatch146
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> origin/master

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git add c.txt

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git commit -m "first commit"
[master 6a89d40] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 c.txt

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 261 bytes | 87.00 KiB/s,
done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1
local object.
To https://github.com/cjcnkolhe/regularbatch146.git
   e37c0d8..6a89d40  master -> master

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git status
```

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be
committed)
        d.txt

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

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git add d.txt

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   d.txt


Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git commit -m "second commit"
[master b0cdb90] second commit
 1 file changed, 1 insertion(+)
 create mode 100644 d.txt

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git status
On branch master
nothing to commit, working tree clean

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 265 bytes | 132.00 KiB/s,
done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1
local object.
To https://github.com/cjcnkolhe/regularbatch146.git
   6a89d40..b0cdb90  master -> master

Nandy@NK MINGW64 ~/Desktop/143 (master)
```

```
$ git checkout -b team1
Switched to a new branch 'team1'

Nandy@NK MINGW64 ~/Desktop/143 (team1)
$ git add re

Nandy@NK MINGW64 ~/Desktop/143 (team1)
$ git commit -m "second branch commit"
[team1 6b93948] second branch commit
 1 file changed, 1 insertion(+)
 create mode 100644 re/c.txt

Nandy@NK MINGW64 ~/Desktop/143 (team1)
$ git push origin team1
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 97.00 KiB/s,
done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1
local object.
remote:
remote: Create a pull request for 'team1' on GitHub by
visiting:
remote:
https://github.com/cjcnkolhe/regularbatch146/pull/new/t
eam1
remote:
To https://github.com/cjcnkolhe/regularbatch146.git
 * [new branch]      team1 -> team1

Nandy@NK MINGW64 ~/Desktop/143 (team1)
$ git checkout master
Switched to branch 'master'


Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git checkout  team1
Switched to branch 'team1'

Nandy@NK MINGW64 ~/Desktop/143 (team1)
$ git checkout master
Switched to branch 'master'
```

```
Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git checkout -b team2
Switched to a new branch 'team2'

Nandy@NK MINGW64 ~/Desktop/143 (team2)
$ git add oe

Nandy@NK MINGW64 ~/Desktop/143 (team2)
$ git commit -m "3rd brach commit"
[team2 1f8c1fd] 3rd brach commit
 1 file changed, 1 insertion(+)
 create mode 100644 oe/d.txt

Nandy@NK MINGW64 ~/Desktop/143 (team2)
$ git push origin team2
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 97.00 KiB/s,
done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1
local object.
remote:
remote: Create a pull request for 'team2' on GitHub by
visiting:
remote:
https://github.com/cjcnkolhe/regularbatch146/pull/new/t
eam2
remote:
To https://github.com/cjcnkolhe/regularbatch146.git
 * [new branch]      team2 -> team2

Nandy@NK MINGW64 ~/Desktop/143 (team2)
$ git checkout master
Switched to branch 'master'

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git merge team2 -m "merge team2"
Updating b0cdb90..1f8c1fd
Fast-forward (no commit created; -m option ignored)
 oe/d.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 oe/d.txt
```

```
Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/cjcnkolhe/regularbatch146.git
   b0cdb90..1f8c1fd  master -> master

Nandy@NK MINGW64 ~/Desktop/143 (master)
$ git clone
"https://github.com/cjcnkolhe/regularbatch146.git"
Cloning into 'regularbatch146'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 21 (delta 5), reused 15 (delta 2), pack-
reused 0
Unpacking objects: 100% (21/21), done.
```