GIT

* **Git** is a **distributed version control system (DVCS)** that helps track changes in files, typically used for source code management in software development.
* It allows multiple developers to work on the same project without conflicting with each other, enabling collaboration, tracking history, and handling different versions of a project.

And github is cloud based service for git

4 3 2 1

Working directory staging area local repository remote repository

## Features

Version control : tracks changes in file

Branching & merging : creates a new branch to work on new features or add new features. After completion of branching we can merge it into main project.

Distributed : git stores a full history of project as a copy in developers local repository.

Collaboration : Multiple users can access and clone the same project work on it locally

and share those changes by pushing or pulling from central repository.

Security:

Speed:

# Why do we use git

Tracking changes

Collaboration

Branching and Experimentation

Backup

Opensource

# Instead of git we can use

Dvcs:

Git, Fossil, Mercurial, Bazzar

Vcs:

Subversion ,Perforce

# Differences

|  |  |
| --- | --- |
| Distribute version control system (dvcs) | Version control system(vcs) |
| Each use has own repository | Centralized, one main repository |
| Work offline full repo locally | Can’t commit without server access |
| Can merge and branch | complicated |
| Combination of pull push fetch | Direct commit to central repo |
| Git, Fossil, Mercurial, Bazzar | Subversion ,Perforce |

Commands:

git init: to initialize git repository

git config: configure user info

git config --global user.name "Your Name"

git config --global user.email "mail @example.com"

git clone<path>

git status : to show current status

git add<file name>

git add .

git commit -m “message” : to save changes

git push

git remote add<name><url>

git push <name># remote repository

git remote add practice<url>

git push practice

git push --set-upstream practice master

git checkout -b “copy1” create branches

git push --set-upstream origin copy1 copying data from main to copy1

git log copy1 to see the committed history

after changing data in txt file

git checkout copy1

git pull origin main

git checkout copy2 // to switch to copy2 branch

git pull origin main //fetch and merge the last changes from remote main branch

git push origin copy1 // upload local commits to repository and updates copy1

git checkout main # Switch to main branch

git fetch origin # Fetch updates from the remote

git pull origin main # Pull the latest changes into main

* [Top 12 Git Commands for Every Developer](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#top-12-git-commands-for-every-developer)
  + [1. git config](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#1-git-config)

# sets up Git with your name

git config --global user.name "<srichandana26 >"

#sets up Git with your email

git config --global user.email "<srichandana837@gmail.com >"

* + [2. git init](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#2-git-init)

# to initialize to create a new git repository

* + [3. git clone](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#3-git-clone)

#Copying entire project from remote location (GitHub , GitLab, or local machine)

#it copies entire project including images, audio, video ,text

* + [4. git status](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#4-git-status)

#The state of repository

* + [5. git add](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#5-git-add)

#move files from working directory to staging area

#used to add files

#must use this command after every change in file

#we can add all files by git add .

* + [6. git commit](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#6-git-commit) -m “message”

#after adding files or anything it saves file in local repository

* + [7. git push](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#7-git-push)

# pushes the files from local repository to remote reository

* + [8. git branch](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#8-git-branch)

#to add new branch to main branch and remove all files from working directory and pull the files into new branch

* + [9. git checkout](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#9-git-checkout)

#switch to an existing branch

* + [10. git merge](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#10-git-merge)

#merge some specified branch with current branch

* + [11. git pull](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#11-git-pull)

# pull the information from remote repository to local repository

* + [12. git log](https://www.geeksforgeeks.org/top-12-most-used-git-commands-for-developers/?ref=lbp#12-git-log)

#shows all repository commits

# I have two branches copy1 and copy2 and in each branch there are two files f1,f2 . then i want to merge these two files and copy into another branch.

1. Created new folder

* git init

1. created text files f1.txt and f2.txt
2. created two branches copy1 and copy2

* git checkout -b copy1
* git add f1.txt
* git add f2.txt
* git commit -m “committed into copy1”
* git push -u origin copy1
* git checkout -b copy2
* git add f1.txt
* git add f2.txt
* git commit -m “committed to copy2”
* git push -u origin copy2
* git checkout master(switched to main branch)
* git push -u origin master
* git merge copy1
* git merge copy2
* git push -u origin master
* git log --oneline --graph
* git log --oneline --graph

\* 8894146 (**HEAD** -> **master**, **origin/master**) Resolved merge conflict (no conflict, just merging copy1 and copy2)

|\

| \* 379eee0 (**origin/copy2**, **copy2**) for merging it is not accepted so updated again

\* | dbaa6c2 (**origin/copy1**, **copy1**) commited to copy1 after update

|/

\* e340883 Initial commit with f1.txt and f2.txt

* git push origin master

Everything up-to-date

|  |
| --- |
| I have two branches copy1 and copy2 and in each branch there are two files f1,f2 . then i want to merge these two files and copy into another branch. |

|  |  |
| --- | --- |
| error: failed to push some refs to 'https://github.com/srichandana26/practice.git' | Changed the git checkout into master |
| Already up to date. | It is not taking same files immediately .it accepts after making changes |
| error: Committing is not possible because you have unmerged files. | Again I added f1 file by git add f1.txt  And make $ git push origin master  Then checked  git log --oneline --graph |
|  |  |

# To delete the branch

In master only because it can possible in master

* git checkout master
* git branch -d copy1 it shows deleted but in repository not deleted
* git push origin –delete copy1 then it deleted in repository and gives hashcode

# To fetch the deleted branch

* git fetch origin to check in remote repository
* git checkout -b copy1 origin/copy1

got message fatal: 'origin/copy1' is not a commit and a branch 'copy1' cannot be created from it

|  |  |
| --- | --- |
| git checkout -b copy1 origin/copy1  fatal: 'origin/copy1' is not a commit and a branch 'copy1' cannot be created from it | * git reflog to show recent actions and changes * git checkout -b copy1 dbaa6c2(hash code) it switches to copy1 * git push origin copy1 then it restored |
| git branch -d copy2  error: branch 'copy2' not found | git push origin --delete copy2  To https://github.com/srichandana26/practice.git  - [deleted] copy2 |

* git reflog to show recent actions and changes
* git checkout -b copy1 dbaa6c2(hash code) it switches to copy1
* git push origin copy1 then it restored

# if I wanted to change file f1 in master then make changes in master branch

git checkout master

git add f1.txt

git commit -m ”message”

git push origin master

# then I want to merge this master file to copy2

git checkout copy2

git merge master

git push origin copy2

# then we will get the same message in master and copy2 branches

## I want to delete single file

Git checkout copy2

git rm f1.txt

git commit -m “message”

git push origin copy2

## fetch that file

$ git log copy2 --f1.txt

But I did not fetch data because the f1 file not committed in copy2 so again I fetch the file from master

git checkout master -- f1.txt

git add f1.txt

git commit -m "Restored f1.txt in copy2"

git push origin copy2

## I want add only f1file to the copy3 but it taking both f1,f2 files because both are already committed in staged area . so trying to remove f2 from branch

git reset f2.txt this command will unstage f2

git status ( expected) the f1 is committed , f2 is unstaged but I found working tree clean , nothing to commit. So again I want to directly delete f2 file from copy3

git rm f2.txt to remove from both staged area and working directory

git rm --cached f2.txt to remove only from staged area

git commit -m “message”

git push origin copy3

then we get compare and pull request at github if we want to merge the copy3 with main then it replaced with copy3