**Version Control Tools**

They are the software which track and manage the changes happening inside our files.

* Make save points, so that we can go back to any save point at any point of time.
* Because of this they provide total development freedom. For ex- we have developed version 1.0 ,all the functionalities are working in that version, so we make savepoint there with the help of git after that we developed 1.1 version but we saw all functionalities are not working there so now because of git we can go back to last savepoint where all the functionalities are working, because of this feature we have total development freedom.
* E.g- Git, subversion, perforce.

**Note** -When the programmer commit(save) the changes it becomes savepoint.

**GIT**

It is a version control tool.

**GITHUB**

It is a online-service, where we can host our repositories. (just like Youtube is a video hosting service), and we are doing this so that multiple people can see,access and contribute to our repository.(collaboration becomes easy).

**GIT REPOSITORY**

It is a folder where changes are by the git.

**DIRECTORY**

Means folder.

**GIT LIFECYCLE**

**Stages of a file in Git**

1. Working Directory(Untracked files)
2. Staging Area(Tracked files)
3. Git Directory
4. Working Directory

Working directory is the folder where files are stored and these files are monitored by git means git is aware of the files but tracking has not been started by git for these files yet.

So if we want to convert our directory into working directory we have to enable git for our directory, we use git init command, this will create an empty git repository in the folder. This means now our local folder/directory has converted into working directory.

1. Staging Area

If we want the changes of our files to be tracked we have to use git add filename command. This command will take snapshot of our current file and store it to staging area. And now the git will start to track that file.

* After staging a file, if you modify the file again, those new changes are considered modifications.
* These modification will be happening inside working directory not in staging area, git is aware of these modification(because that file is tracked by the git) as git will compare the snapshot of staging area with the file in working directory and it will get to know that modifications have happened.
* However, these modification won’t be included in the next commit as they are present in working directory unless you run git add again to stage them for commit.

// to add one file to staging area

git add <filename>

// to add all files of our folder/directory to the staging area

git add .

1. Git Directory

When we commit all the changes which are staged, the snapshot present in staging area will saved in git repository so it means all the changes present in staging area will be permanently saved in git repository so that we can come back to these changes later whenever we want to. It is like creating savepoints on which we can come back later.

git commit -m <Commit Message>

git repository = snapshot of changes that are commited + date + author + message

**What Does "Tracking" Mean in Git?s**

* + Once you use git add, Git starts **tracking** the file. A tracked file can exist in the following states:
    - **Unmodified**: The file is being tracked, and no changes have been made since the last commit.
    - **Modified**: The file has been changed in the working directory but has not been staged yet.
    - **Staged**: Changes to the file have been added to the staging area and are ready to be committed.

**Elaborate Explaination of Stages of a file in Git**

**1. Working Directory (Untracked or Modified State)**

* Definition: The working directory is where you have your actual project files. It's the state of the file on your local machine where you create, edit, or delete files.
* Key Characteristics:
  + Files in the working directory may be:
    - Untracked: Git is not tracking these files yet.
    - Modified: The file is already tracked but has been changed since the last staging or commit.
* Example:
  + After creating a new file:

touch new\_file.txt

git status

Output will show new\_file.txt as untracked.

**2. Staging Area (Staged State)**

* Definition: The staging area (or index) is a place where you prepare files to be included in the next commit. When you stage a file, Git takes a snapshot of the file’s current state and puts it in the staging area.
* Key Characteristics:
  + Files must be explicitly staged using git add.
  + Only files in the staging area will be included in the next commit.
* Example:
  + To stage a file:

git add new\_file.txt

git status

Output will show new\_file.txt as staged, ready to be committed.

**3. Repository (Committed State)**

* Definition: The repository contains the committed history of the project. When you commit files, their state is saved permanently in the local repository.
* Key Characteristics:
  + Once a file is committed, its state becomes part of the repository's history.
  + Tracked files remain part of the repository even after modification, but modifications need to be staged and committed again to update the repository.
* Example:
  + To commit a staged file:

git commit -m "Initial commit"

git status

Output will show the working directory as clean, meaning there are no uncommitted changes.

.gitignore file 🡪 we mention those file name inside the .gitignore file which we want Git should git should not track.

**Commands for Git Bash**

1. pwd (print working directory) used to see in which folder we are
2. cd pathOfFolder (change directory) to move from one folder to

another. For ex- cd D:\Probation\Git\NewFolder

1. ls to list all the file and folders you are in.

ls -a to see hidden folders

1. git diff it show file name which is modified and also show the

data which is not modified (both versions of that data before

modifying and after modifying).

1. ls -al it shows three folder.

. currently in which folder we are present

.. parent folder of current folder

.git this is the folder where meta data is present means which file are modified, which file is added to staging area etc

which will be used by git..

1. q to exist the command
2. cls to clear the terminal screen
3. git init to add git to that folder. (it will convert our folder into git

repository)

1. git add filename.ext - it will move the changes done in working directory to staging area. For example git add APP.java
2. git add . all the changes done in working directory will move to staging area.
3. git status name of the files modified, staged, and untracked.
4. git commit -m “write your message” it will save our changes permanently in our repository.
5. git checkout hashvalue move back to that particular commit changes.
6. git branch give you names of all the branch present in git repository
7. git checkout branchName move back to the latest commit

**Two ways to make git repository**

1. Make a folder in your system and add git to that folder…convert that folder into git repository..

**How to do this**

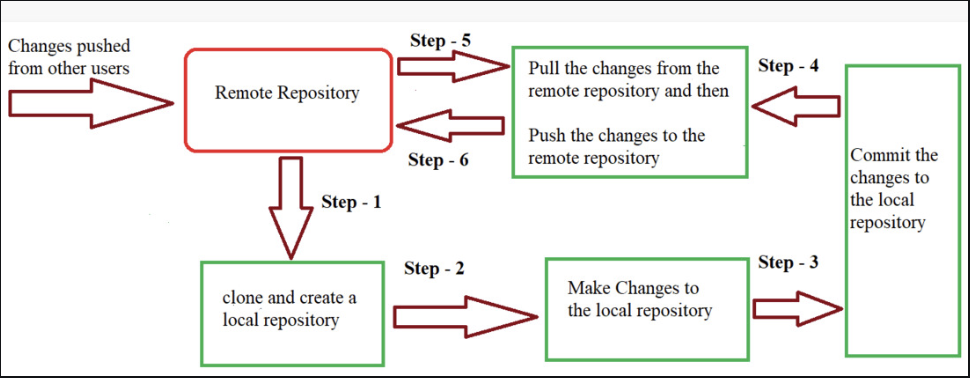
Make a folder into your system and go inside that folder open git bash, use “git init” command to initialise that folder with git

1. Directly clone repository from github

Clone – means bringing git repository into your local system.

**How to do this**

Make a folder and go inside that folder and use “git clone” command. For ex git clone /path/to/your/repository

****

• In Step – 1 We first clone any of the code residing in the remote repository or make our own local repository.

• In Step-2 we edit the files in our local repository and make the necessary changes in it.

• In Step-3 we commit our changes by first adding them to our staging area and committing them with a commit message.

• In Step – 4 and Step-5 we first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.

If there are no changes we directly proceed with Step – 6 in which we push our changes to the remote repository and we are done with our work.

**REVIEW A REPO HISTORY**

If you want to see all the commit history use

git log - it will show all the commits with commit hashvalue, name of author(who did the

commit), date of commit and message given at the time of commit and hashvalue of

the commit.

git log -3 (only latest three commit will be shown)

Number of commit you want to see, you can enter any

number

If you want to see the which files and what content is modified use

git log -p it will show git log + git diff

If you want to find any commit fastly and you remember its message use

git log --oneline it will show commit id + commit message

If you want to see only which files are modified use

git log --stat it will show git log + name of file modified,

number of modification done in that file

If you want to see what changes are done in a particular commit use

git show commitId all the changes done in that particular

commit will be listed.

How to push your code in github

Create a repository in github go to you repository then click on new …. Give name to you

Repo and then click on create

Now open git bash inside the folder which you want to upload and then enable git using git init and then send all changes to staging area by git add .

And then git commit – m “message” and then paste three command coming in your github repository one by one in git bash.