**GIT**

**GIT** is a **Distributed Version Control** **System** tool that supports distributed non-linear workflows by providing data assurance for developing quality software. It lets you and your team of developers work together on the same project from anywhere.

* Version Control System and GIT Repositories
* Staging area & Commits
* Branches
* Issues and Pull Requests

**Repository:** It is a **directory** or **storage space**  where your projects can live. It can be local to a folder on your computer or it can be a storage space on another online host (such as GITHUB). You can keep code files, text files, image files.

* To move the files from working directory to staging area (#git add <filename>)
* To move the files from staging area to local repo (#git commit -m “write any comment here”)
* Create a Remote Repo and add the path to git( #git remote add origin <link of remote repo>) after that (#git push --set-upstream origin master)
* #git push -m origin <branch Name/master>
* To move the files from local to remote repo (#git push)
* To jump from one branch to another branch( #git checkout -b <branch name>) (-b is used to create a new branch)
* To initialize a folder as repo (#git init)
* To check the status (#git status)
* To get the remote repo to local , copy the remote http address and open terminal (#git clone <http addresss>)

**Workflow of GIT**

|  |  |  |  |
| --- | --- | --- | --- |
| Work Space | Staging Area | Local Repository | Remote Repository |
| The untracked files are exists | To get files from work space to staging area  #git add <file name> | To move the files from staging are to local repo  #git commit -m “Write a comment about the commit” | To move the files from local repo to remote repo  #git push |



1. **What is Version Control System?**

Version Control is a system that record changes to a file or set of files over time so that you can recall specific versions later. These versions are recorded in a repository and can be recalled from the same.

We can easily roll back from current version to previous version.

You can reverse the faulty update and save time.

Analyze and compare version.

* **#git init** (Creates a new Git repository)
* **#git clone** (When you clone a repo, you create a copy of the original repo on your local machine)
* **#git fork** (When you fork a repo, you create a copy of the original repo on your GitHub account)
* **#git pull origin master** (Lets you copy all the files form the master branch of remote repo to your local repo)
* **#git push origin master** (Lets you push local changes into central repo)
* **#git log** (We can track all the commits have been done in my local repo)
* **#git show <filename/commit id>** (To show the complete details and info inside a file)
* **#git add .** (To add all the files from working directory to the staging area)
* **#git commit -m <message>** (To commit the files from staging area to local repo)
* **#git remote add origin <repo\_link>** (Lets you add a remote repository)
* **#git ls-files** (To list out the available files)
* **#git status** (Tells you which files are added to index and are ready to commit)
* **#git remote -v** (To print out the remote repo location)

**Process**

* Create a folder in local disk
* Initialize it as a Git repo by using the command ($git init)
* Open that folder in Vscode
* Create a file and save it in Vscode and check the status by using ($git status)
* Add that file from working directory to the staging area by using the command ($git add <filename>)
* Commit that file from staging area to the local repo by using the command ($git commit -m “comment”)
* Create a remote repo in Git hub
* Add remote repo to the working directory by using the command ($git remote add origin <SSH Address/HTTPS Address)
* Push that file from local repo to remote repo by using the command ($git push –set-upstream origin master)

1. Create a folder , go inside the folder
2. #git init (Initializing the folder as git repository)
3. Create a file in it.
4. #git status
5. There is an untracked file in it
6. #git add <file\_name>
7. There is an uncommitted file
8. #git commit -m “Comment about the commit”
9. To see what are the modifications done to the file (#git diff)
10. Now modify the file to the version 2
11. #git add <file\_name>
12. #git commit -m “This is the second version”
13. Now if you want to go back to the previous version
14. #git log
15. Copy the first commit ID and enter the following command
16. #git reset –hard <commit\_id>
17. Now check the file , it is modified to the version 1

**HTTPS: you can use Github password to download the code**

**SSH: You can use your public key to download the code**

**Git merge & git rebase & git cherry-pick**

**#git cherry-pick <commit-id>** (if we want to merge files from one branch to the main branch , without going into that branch we can directly merge those files from the main branch itself by the above command) (git cherry-pick is easy when there is 1 or 2 commits)

**git merge & git rebase**

both commands are used to merge the files from the other branches to the main branch , but when we use the git merge the commits goes to the top in the git log, while rebase will be not prioritized.

If we use git merge the commits are not in the linear way, while if we use the git rebase the commits are in linear way that everyone can understand and track easily.

**Both git rebase and git merge will do the same things whereas if you are using rebase you get a linear commit history whereas with merge you don’t get it.**

**What is the use of cherry-pick command in git?**

Cherry-picking in Git stands for applying some commit from one branch into another branch. In case you made a mistake and committed a change into the wrong branch, but do not want to merge the whole branch. You can revert the commit and apply it on another branch.