**GIT AND GITHUB**

**About Git: -**

* Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people.
* It is a **Distributed Version Control System**. So Git does not necessarily rely on a central server to store all the versions of a project’s files. Instead every user “clones” a copy of a repository (a collection of files is called a repository) and has the full history of the project on their own hard drive.
* This clone has all of the **metadata** of the original while the original itself is stored on a self-hosted server or a third party hosting service like **GitHub**.

**Git Vs GitHub: -**

* **Git is version-control system** for managing our source code history, whereas **GitHub is a hosting service** for our Git repositories.
* Git was created by Linus Torvalds in 2005 for development of the Linux Kernel, with the kernel developers contributing to its initial development. And GitHub was created by four members: - Tom Preston-Werner, Chris Wanstrath, Scott Chacon, P.J. Hyett in February 8, 2008(as Logical Awesome LLC) in San Francisco, California, United States.

**Why we should use Git: -**

* Undo our mistakes we do in our projects or codes.
* Distributed Development
* Don’t miss things up
* Community support is very huge and very much supportive

New Merge Commit

Master Tip

Feature Tip

Common Base

**Basic Git Commands: -**

1. git init = Used to initialize a git repository for a new or existing project.
2. git add <file\_name> = Used to add one or more files to staging(index).

NOTE:-> If you have many files in a particular folder and to want to add them all at the time then use the command “git add .“, this will add all the files in that folder at one go.

1. git status = List the files we have changes and those we still need to add or commit.
2. git commit –m “commit message” = Commit changes to head.
3. git log = Used to see the history of our commits.
4. git rm –cached <file\_name> = Used to remove files from Staging area(Unstage).
5. git branch <branch\_name> = To create a new Branch with name as branch\_name.
6. git branch = Used to show the number of branches we currently have. The astringe (\*) will show you on which branch you currently are.
7. git checkout <branch\_name> = Used to switch from current branch to another.
8. git merge <branch\_name> = Used to merge a branch into current branch.
9. git checkout dev/master = Used to jump from master branch to dev branch and vice-versa.

NOTE:- In place of dev write branch name

1. git diff filename = Used to check what new things have added to our file after the last commit.
2. git clone “link” = Use to clone/upload a project to our computer.

**Initializing a Git Repository: -**

* The most effective way to run Git is through a command line prompt. That means we will be using Terminal in Mac/Linux or Command Prompt or PowerShell in Windows, or we can use Git Bash also.
* The first step is to initialize a repository(repo), a location for files and their revision history.
* If we want a new repo in a new folder, then we will need to create a new directory(folder) and then switch to that directory.

**Tracking Files: -**

* In general, files in a repo can have the following statuses:

1. Not tracked/ Untracked -> It means that the file is not being tracked by Git for changes. We should explicitly say which files it should follow/track.
2. Staged
3. Committed

* To find out the actual status of the files in a repo, the **git status** command is used.
* The **git add ‘filename’** command tells Git to track the file. This step is called **staging.**
* If we don’t want Git to track some specific files, we can “ignore” them.

**Git Commit: -**

* The git commit command saves the stage of our project by adding snapshots of staged files to the repository.
* The git commit can include the –m flag with a message describing what we have changed.
* Executing **git status** after a git commit verifies that tracked files are up to date.
* To commit modifications for every tracked file in the repo, use the git command –a before the –m command.

**Cloning and Pushing: -**

* After committing the changes, the next step is pushing the local repo to the Git server on a remote location (GitHub or BitBucket).
* If you have initialize a local repository, we can connect to the remote one using the following command -> $ git remote add origin <https://www.github.com/user/project_name.git> (copied link)
* After making our local changes and commits, now it’s time to push the changes to the remote repository. The push command tells Git where to put our commits -> $ git push –u origin master
* The –u tells Git to remember the parameters, so that next time we can simply run git push and Git will know what to do. Also remember that it is not necessary to push our project to a remote location, we are free to work on our project alone without pushing it anywhere, if we need to.

**Git Workflow: -**

* A git workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner. Git workflows encourage uses to leverage Git effectively and consistently. Git offers a lot of flexibility in how users manage changes.
* Here is a Git workflow diagram given below for a better understanding.

Staging

Area

Remote Repo

(MASTER)

Local Repo

(HEAD)

Working Directory

Git Pull

Git Fetch

Git Merge

Git Push

Git Commit

Git Add

**Example with code: -**

* Let us imagine, we have created a folder LearnGitRepo in which it has two files of java program -> HelloWorld.java and ForLoops.java.
* Now to push this folder code on GitHub we will do the following coding on our command prompt.
* The lines written with **blue color is the code written by the user** and the lines with **black color is the output given by the computer** after processing our codes: ->

C:\Users\DELL\Desktop>cd LearnGitRepo

C:\Users\DELL\Desktop\LearnGitRepo>git init

Initialized empty Git repository in C:/Users/DELL/Desktop/LearnGitRepo/.git/

C:\Users\DELL\Desktop\LearnGitRepo>git add HelloWorld.java

C:\Users\DELL\Desktop\LearnGitRepo>git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: HelloWorld.java

Untracked files:

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

ForLoops.java

C:\Users\DELL\Desktop\LearnGitRepo>git add ForLoops.java

C:\Users\DELL\Desktop\LearnGitRepo>git commit -m "for loops understood"

[master 54f8a1d] for loops understood

1 file changed, 7 insertions(+)

create mode 100644 ForLoops.java

C:\Users\DELL\Desktop\LearnGitRepo>git status

On branch master

nothing to commit, working tree clean

C:\Users\DELL\Desktop\LearnGitRepo>git remote add origin https://github.com/Saurabh-Shivam/GitLearningRepo.git

C:\Users\DELL\Desktop\LearnGitRepo>git remote -v

origin https://github.com/Saurabh-Shivam/GitLearningRepo.git (fetch)

origin https://github.com/Saurabh-Shivam/GitLearningRepo.git (push)

C:\Users\DELL\Desktop\LearnGitRepo>git push -u origin master

Enumerating objects: 12, done.

Counting objects: 100% (12/12), done.

Delta compression using up to 8 threads

Compressing objects: 100% (11/11), done.

Writing objects: 100% (12/12), 1.29 KiB | 330.00 KiB/s, done.

Total 12 (delta 2), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (2/2), done.

To https://github.com/Saurabh-Shivam/GitLearningRepo.git

\* [new branch] master -> master

Branch 'master' set up to track remote branch 'master' from 'origin'.

C:\Users\DELL\Desktop\LearnGitRepo>git pull

remote: Enumerating objects: 4, done.

remote: Counting objects: 100% (4/4), done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 771 bytes | 48.00 KiB/s, done.

From https://github.com/Saurabh-Shivam/GitLearningRepo

e4a9ab0..4ce7f8a master -> origin/master

There is no tracking information for the current branch.

Please specify which branch you want to merge with.

See git-pull(1) for details.

git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream-to=origin/<branch> dev

C:\Users\DELL\Desktop\LearnGitRepo>git add .gitignore

C:\Users\DELL\Desktop\LearnGitRepo>git status

On branch master

Your branch is up to date with 'origin/master'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: .gitignore

C:\Users\DELL\Desktop\LearnGitRepo>git commit -m "just added git ignore"

[master 3be49e5] just added git ignore

1 file changed, 1 insertion(+)

create mode 100644 .gitignore

C:\Users\DELL\Desktop\LearnGitRepo>git push

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 8 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/Saurabh-Shivam/GitLearningRepo.git

4ce7f8a..3be49e5 master -> master