Md Sajid Ali : Day 1 : Learning Git and GitHub

Pdf : pro git second edition

Starting from : vcs -> version control system to track changes in code (like bank statement credit debit) tracks all the history .

Main task of git:

1. To track the history

2 . to collaborate

Github : help user to store their code using git

Git thinks of its data more like a series of snapshots of a filesystem.

Point: Most operations in Git need only local files and resources to operate — generally no information is needed from another computer on your network.

Point : Everything in Git is checksummed before it is stored and is then referred to by that checksum. This means it’s impossible to change the contents of any file or directory without Git knowing about it.

Point : The mechanism that Git uses for this checksumming is called a SHA-1 hash. This is a 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git. A SHA-1 hash looks something like this: 24b9da6552252987aa493b52f8696cd6d3b00373.

The Three States of git : Git has three main states that your files can reside in: modified, staged, and committed:

Git branching : Branching means you diverge from the main line of development and continue to do work without messing with that main line. In many VCS tools, this is a somewhat expensive process, often requiring you to create a new copy of your source code directory, which can take a long time for large projects.

Created git hub account with trueIgtech email.

Installation of git, git bash

Configuration : git config user.name “mohammad sajid ali”

Git config user.email [msajidali@trueigtech.com](mailto:msajidali@trueigtech.com)

Git config –list

Git commands : clone and status

Clone : to clone a repo on a local machine (pc)

Git clone <link>

Status: display the status of code

Git clone <https://github.com/sajidigtech/day1atigtech.git> (before this use cd to set local directory)

/d/GitDemo/folder1/day1atigtech (main)

git status

On branch main

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

nothing to commit, working tree clean

command : ls –(LS)

a : gives the list of all hidden files like .git

after cloning I changed readmefile and there was warning showed as modified changes in readme

when checked status.

Create new file in cloned repo : index.html -> git status -> untracked file found

Staged area : git add index.html

Again after this we can check status

We can also use “git add .” command

git commit -m "in this commit i have cloned repo , done with status check adding etc"

status :

Your branch is ahead of 'origin/main' by 1 commit.

Means all file as compared to prev version og github repo is ahead of 1 commit as we did in local environment after cloning

Till now I had issues as my gitbash was registered with old account instead of igtech account

git remote set-url origin <https://github.com/sajidigtech/day1atigtech.git> i used this command to set new authentication and removed old account from windows credentials.

And then I :

Git commit -m”commit msgs”

git push origin main: the github folder on server was updated

now talking about branch :

commad to create branch : git branch branch1 # create

git checkout feature-branch1 switch to that branch

git push origin branch1

now pushing the doc in blank folder

git init

git remote add origin https://github.com/sajidigtech/day1atigtech2.git

git add .

git commit -m "added Day1 LearningGit.docx"

git branch -M main

git pull --rebase origin main # very important

git push origin main

……………….again

Git init

Git remote add origin <link>

Git remote -v (to verify remote)

Git branch - > generally master

Change to main

Git branch -M main (to rename branch to main)

Git push origin main

Git push -u origin main.

Work flow :

Github repo -> clone c->changes- >add->->commit->push

Talking about branch: we cannot delete any branch while being on that branch

Eg: git branch ->branch1

Git branch -d branch1 : cannot delete

There fore:

Git checkout main then delete :

Git head is a pointer pointing towards current branch on which we re working

We can switch from one branch to other using git switch and git checkout

Git switch -c newbranch : create and switch to new branch

Git diff –staged

Git diff branch1 branch2

Git diff <commit hash> <commit hash2>

Stash : is a way to save your changes in a temporary location, it means we changed few content in one branch and without committing changes into that or adding anything, we can switch the branch to other branch

Git stash list

Git stash drop

Git stash pop

Git tags :

Sticky note similar to commits

Git tag <tagname>

Rebase : history ko re write krta hai isme,

Rebase in git : used to change the base of a branch,it effectively allows you to move a branch to a new starting point,usually a different commit by replaying the commits from the original base onto the new base.

Iska main kam hai base ko re align krna

Git pull, git fetch , git clone

Git clone :

>git clone <link>

>cd <repo name which is cloned>

> makes repo git initiated without mentioning “git init”

> I implemented it and also after making changes pushed the codes and checked on github

**Now imp**

**Git pull vs git fetch**

**Git pull :**

Remote se changes le aata hai aur apply bhi kar deta hai.

Code update ho jayega agar github pe already badme ksi ne changes kiye toh !

**Git fetch:**

Remote se sirf info (commits, changes) laata hai — changes ko apply nahi karta,

Code bilkul waise ka waisa hi rhega local repo me.

As I did not practice these both on project level, my thinking is to create github repo with readme file, clone that into local vs code repo -> make changes and push to the github repo.

Now create two more local vscode repo to implement **git pull and git fetch.**

**Total 3 repositories (local github)**

**One for pushing code into the github repo**

**One for pulling and one for fetch**

**Elaboration:**

Daytwopractice folder -> cloned -> main branch empty and formbranch ->form creation html ->push

Daytwopractice2 folder -> git init

* git remote add origin <https://github.com/sajidigtech/DayTwoExample.git>
* git fetch origin
* gave dtails like :

remote: Enumerating objects: 9, done.

remote: Counting objects: 100% (9/9), done.

remote: Compressing objects: 100% (7/7), done.

remote: Total 9 (delta 1), reused 3 (delta 0), pack-reused 0 (from 0)

Unpacking objects: 100% (9/9), 2.27 KiB | 105.00 KiB/s, done.

From https://github.com/sajidigtech/DayTwoExample

\* [new branch] formbranch -> origin/formbranch

\* [new branch] main -> origin/main

-> git checkout -b formbranch origin/formbranch

formbranch naam ki local branch banayi Usko link kar diya origin/formbranch (jo GitHub pe hai)

indirectly with get fetch

->git checkout -b main origin/main

Switch to form branch

Create another form and push to github repo

* git add .
* and then git commit -m”form2 entered through folder2”
* git push origin formbranch

now doing pull via third folder

**Git reset : soft**, **mixed**, aur **hard**.

HEAD : latest commit pointer of current branch

git reset --soft HEAD~1 ek commit peeche hona

HEAD~2 = Last 2 commits will be removed

git reset --soft HEAD~1

commit hatega, staging and code will remain as it is

git reset --mixed HEAD~1

Commit + staging dono hata deta hai

git reset --hard HEAD~1

Commit bhi gaya

Staging bhi gayi

Code bhi gaya / ya fir written content jo bhi tha sb htega

git reset --hard <commit-hash> the best way

**git log vs git reflog**

Kisne commit kiya

Kab kiya

Kya message diya

Aur uska SHA (unique ID)

git log –oneline

**git reflog**

Tumne **HEAD (current branch pointer)** ko kab-kab kahan move kiya

Sha -> hash id

**Pull, merge and rebase !**

**Problem Statement:**

If I have ek local branch (feature-branch), aur tu uspe kaam kar raha hai.  
Par usi beech me main branch (remote) pe kuch naye commits aagaye hain.

Ab tu main ke naye changes apni branch me lana chahta hai.

**git pull origin main**

***(by default ye merge karta hai)***

**git pull --rebase origin main**

***(rebase karta hai)***

**git pull (Default = Merge):**

Ye kya karega?

* Remote main se naye commits lega
* Unhe merge karega current branch ke sath
* Result: ek extra merge commit Banega

Isse history thodi "messy" ho sakti hai

Remote main: A -- B -- C1 -- C2 -- C3 ← (latest main HEAD)

\

Feature branch: F1 -- F2 ← (your feature work)

**Scene**

Tere paas ek **main** branch hai.

* Tu ek **new feature branch** banata hai feature, jab main ka HEAD C1 pe hai.
* Tu feature branch me kaam karta hai, aur 2 naye commits karta hai: F1 & F2
* Meanwhile, kisi aur dev ne main branch me aur commits kar diye: C2 & C3
* Tu kaam kar raha tha C1 ke baad, tabhi kisi ne main me aur kaam kar diya (C2, C3)
* **git merge main (while on feature branch)**
* Ye kya karega?
*  Ye main ke current HEAD (C3) ko lega
*  Aur feature ke commits (F1, F2) ke saath merge karega
*  Ek **merge commit** banega: M

C2 -- C3

/ \

A -- B -- C1 M (merge commit)

\ /

F1 -- F2

git rebase main (while on feature branch)

Ye kya karega?

* Pehle F1 & F2 ko **temporarily hata dega**
* Phir main ke naye commits C2, C3 leke aayega
* Phir F1, F2 ko **C3 ke baad apply** karega

A -- B -- C1 -- C2 -- C3 -- F1' -- F2'

**Tag** ek pointer hota hai kisi particular commit ke upar, jise usually software ka **release version** mark karne ke liye use kiya jata hai.

Types of Tags in Git:

1. Lightweight Tag

Ek simple pointer hota hai commit ke upar.

Metadata store nahi karta (jaise author, date).

git tag v1.0

1. Annotated Tag

Zyada information store karta hai (jaise message, author, timestamp).

git tag -a v1.0 -m "Release version 1.0"

**git –amend**

Git me --amend ek **powerful command** hai jo **last commit ko edit karne** ka option deta hai — bina naye commit ke. Ye mainly tab kaam aata hai jab:

* Tum **galat message** likh dete ho
* Tumne kuch **files commit karne bhool gaye**
* Tum chahte ho **latest commit ko overwrite** karna

git commit –amend

eg.

git commit -m "Finshed login page"

git commit --amend -m "Finished login page"

if missed file in commit

git add style.css

git commit –amend

Remove File from Commit

git reset HEAD unwantedFile.js

git commit –amend

**cherry picking**

In Git, cherry-picking is a way to apply a specific commit from one branch onto another branch, without merging the whole branch.

Syntax : git cherry-pick <commit-hash>

Suppose you're on the main branch:

git checkout main

And you want to bring a specific commit from feature-branch:

git log feature-branch

Find the commit hash, e.g., a1b2c3d.

Then cherry-pick it:

git cherry-pick a1b2c3d

goal :

 Create a Git repo

 Keep main branch empty

 Create sidebranch

 Add form1.html → commit it

 Add form2.html → commit it

 Switch to main

 Cherry-pick only the first commit (form1.html)

Step 1 Create a new folder and initialize Git

mkdir git-cherry-pick-demo

cd git-cherry-pick-demo

git init

step 2 Create an empty main branch

git checkout -b main

(Now main is empty and active.)

Step 3: Create a new branch sidebranch

git checkout -b sidebranch

**Step 4: Add form1.html (branch:sidebranch)**

git add form1.html

git commit -m "Added form1.html"

**Step 5: Add form2.html (branch : sidebranch)**

git add form2.html

git commit -m "Added form2.html"

**Step 6: Switch back to main**

git checkout main

Step 7: Cherry-pick the commit of form1.html

First, get the commit hash of the first commit:

git log sidebranch

You’ll see two commits:

commit a1b2c3d4 (← use this hash for form1.html)

Author: ...

Date: ...

Added form1.html

commit e5f6g7h8

Author: ...

Added form2.html

Now cherry-pick only the first one:

git cherry-pick a1b2c3d4

**Cherry-picking multiple commits**

git cherry-pick <commit1> <commit2> ...

Or a range:

git cherry-pick commit1^..commit3

**abort cherry picking:**

**git cherry-pick –abort**

git cherry-pick --abort sirf us situation me kaam karta hai jab:

Cherry-pick process abhi complete nahi hua ho — kyunki conflict aa gaya ho aur Git ne cherry-pick ko pause kar diya ho.

Till now almost 90% of topics are covered, few implementations are required to understand in more depth