**1.--->Git Is a Version Control System**

Advantages :

- Easily Recover Old Files

- Who Introduce An Issue When (It Mean Who Changes The Data Or Update Data)

- Rollback To Previous Version Of Code

**2.--->History Of Version Control System :**

-> Local Version Control System:

Firstly Developer Created Local Database And Store Their Versions Of Code In Local Database (Cons : That Local Version Control Can Damaged Or Maybe Can Affect And Loose Your Data) (Pros : It Can Track File & Rollback)

-> Centralized Version Control System :

Basically a Centralized Server In Which User Can Pull And Push Files And Works Same As Like Local Database But Its a Different Server Not In Our Computer (Cons : If The Server Damaged And Anything Happened Then The Data Can Be Loose)

-> Distributed Version Control System :

Same As Centralized Server The Change Is Every User Has Its Own Data's Backup In Thier System

And Git Is Distributed Version Control System

--Linus Torvals (Created Of Git)

Difference Between Git And Github :

Git Is Repository And Github Is Used To Host The Git Repository

--->Features Of Git :

-> It Captures The Snapshot (When You Create Git Repo It Create .Git File Than You Can Access Every Version Of Your Project Or Code)

-> Almost Every Operation Is Local(All Git Operation Performed In Your Computer Than It Pushes The Repo In Server)

-> Git Has Integrity (Git Internally Use Checksum To Verify The File) Checksum Is One Kind Of Verfication Method For Files

-> Git Generally Only Add Data (It Only Adds Data In .Git Repo When You Made Change In File)

--->Git Installed :

- We Got Git Command Line Interface & We Got Gitbash(Basically Commandline)

- Cli(You Have To Use Commands)

- Gui(It Provide Gui And Runs Command In Back And Thru Gui You Can Use Git)

For Setting User In Git You Can Use This Command

For Name : Git Config --Global User.Name "Akshay"

For Email : Git Config --Global User.Email "akshaykomade012345@gmail.com"

---> Git Three Stage Architecture :

To Save Any Version Of File We Call Them "Commit"

1.Working Directory : Currently Working Area

2.Stanging Area : Set Of Files Thats You Want To Upload

3.Git Directory : .Git Named Folder Where Our Files Saved In Compressed Version

Command :

**Git Status :** For Checking The Git Repo Is It Or Not

**Git Init :** To Make Repo Is Git

**Git Add --a :** Stage All Change

**Git Commit -m "This Is Modified" :** To Save Changed File

**Git Log:** To Check All The Commits

**Git Clone** "Link Of Any Github Repo" "Folder Name That You Want To Give To Repo": To Clone Any Github's Repo

**Rm -Rf .Git :** To Delete a Git Repo

**Ls :** List

**Pwd :** Present Working Directory

**Cd :** Change Directory

**--> File Status Life Cycle**

**Untracked** : When You Initialize The Git Repo That Stage

**Unmodified** : When Add Any File Using Git Add --a Or Maybe Solo File That Stage

**Modified** : When You Modify Any File That File After Adding Files

**Staged** : Finally Stage That Where All Unmodified And Modified Are Added Usiing Git Add Command

To Ignore Any Files During Commit Or Staging You Have To Create a (.Gitignore) And Write The Name Of File In It So Git Will Ignore That File While Adding Or Staging Repo

You Can Use Touch Command To Create Any Black File

Touch "File Name With Extension" Ex : Touch Index.Html

If You Want Remove Any Particulars Extension File Like If You Dont Want To See Any File With .Log Extension In Stage Or Unmodified Than You Have To Put "\*.{Extension}"

Ex : \*.Log

You Can Also Ignore Any Directory Or Folder While Performing Any Thing Just Put Directory/Folder Name In Gitignore File With "/"

Ex : Dir/

Git Diff : Its Command To Check Difference Between Working Stage And Staging Stage For Same File

Git Diff --Staged : It Compares Previous Commit To Its Current Staging Area And Show Its Result

To Skip Staging Area :

Git Commit -a -m “add commit comment”: It Skip The Stage Area And It Doesnt Commit Untracked File

How To Move Files Or Remove Files :

Git Rm "Filename" : It Removes The Files From Dirctory And Commit The Action

How To Rename Any File :

Git Mv "The Current Name" "Changing Name" : To Change Any Files Name

To Untrack Any File From Gitignore

Git Rm --Cached "File Name"

Git Log -p : To Show On Going Logs Done By Users With Their Differences What Removed And Added

Git Log -p -1 : We Can Also Specify The How Many Commits You Wants To So

Git Log --Stat : It Show Detailed Commit Of User What Have Done By User In Repo In Each File

Git Log --Pretty=Oneline : It Shows All Commit In Oone Line

Git Log --Pretty=Format:"%Ae == %h" : To Show Filtered Commit As According What You Want Ae Means Author Email An Means Author Name

Git Commit --Amend : To Edit Any Commit By Other User

How Unstage Staged Files :

Git Restore --Staged "Filename" : Alter Command In My Pc

Git Rm --Cached "Filename"

Git Checkout -- "File Name" : To Go Back To Previous Version Of Your File Previous Commit

Git Checkout -f : To Got Back To Your Last Commit File And Remove All The Current Channges

Working With Remote Repo'S :

Git Push -u Origin Master : To Push Your Repo To Github

Alias : Its Mean Write Short Command Instead Of Long Command

Git Config --Global Alias."Shortcut That You Want" "Original Command"

Ex : Git Config --Global Alias.St Status

Branching :

In Git, a Branch Is a New/Separate Version Of The Main Repository.

Branches Allow You To Work On Different Parts Of a Project Without Impacting The Main Branch.

When The Work Is Complete, a Branch Can Be Merged With The Main Project.

You Can Even Switch Between Branches And Work On Different Projects Without Them Interfering With Each Other.

To Craete New Branch

Git Checkout -b "Branchname"

Git Checkout Master : To Get Back To Master Branch'

To Check All Branch : Git Branch

To Merge Sub Branch To Master Command : Git Merge "Branch Name"

Merge Conflict :

Merge Conflicts Are The Conflicts That Occur When a Developer Is Editing a File In a Particular Branch And The Other Developer Is Also Editing That Same File Or When Developer A Edits Some Line Of Code And That Same Line Of Code Is Being Edited By Another Developer B That Leads To Conflicts While Merging.

To Solve Merge Conflit Vs Code Provides Simple Solution What You Need In Current Branch While Merging It

Branching Workflow :

There Are 2 Types Of Branches :

1->Long Running Branches

2->Topic Branches

Long Running Branches Inshort Used For To Resolve Real Time Error Or Bug

And Topic Branch Used For Idea Impletations

Remote Files :

**Git Remote -v :** To Check Remote Repos In Git Repo

**Git Remote Push Origin "Repo Url From Github" :** To Add Remote Origin

**Git Push -u Origin Master :** To Push Your Repo On Github

**Git Push -d Origin "Branch Name" :** To Delete Branch From Github

**Git Push Origin "Branch Name" :** To Push Branches In Same Repo