**GIT and GITHUB**

* version control 🡪 used for tracking changes in source code

maintain a history of all modifications made to the codebase, can go back to previous version

* collaboration
* distributed version control

**version control system**

1. Local version control

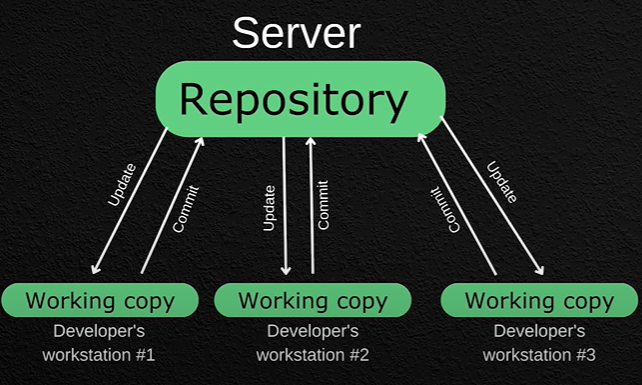
entire version history of the project is stored on the local disk

operates on a single machine

1. Centralized Version Control System(CVCS)

Central repository, collaboration, access control, have backup, commit history

Remote repositories - github, bitbucket



// to verify git is downloaded or not

git –version

// to view config properties

git config

// to view name and mail details

git config –global --list

// need to config username and mail

git config –global user.name “dinesh”

git config –global user.email “[dinesh@gmail.com](mailto:dinesh@gmail.com)”

git bash- to interact with the git

// to view status

git status

//to initialize git(.git🡪 staging area, commit history present)

git init

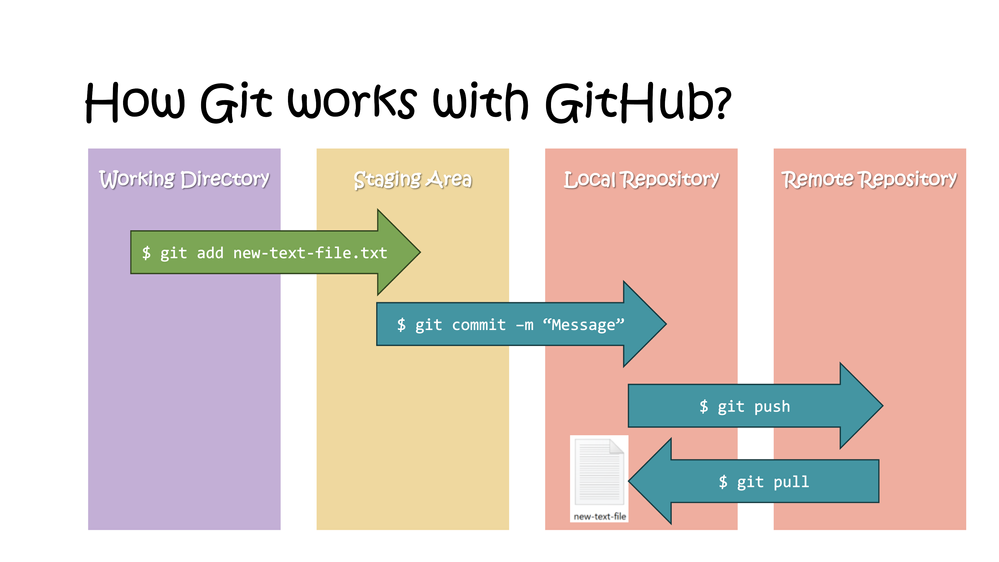
note: default master branch is created so..create as main for more inclusive and respectful language

// with main branch

git init -b main

**flowchart**

working area /directory🡪add/update/delete ( staging area )🡪commit (git local rep)🡪 push (github- remote rep)



// to track all files in staging area and track untracked files

git add index.html

git add .

// to see history of commits

git log

// commit- to get tracked by git

git commit -m ”commit\_message”

// skipping the staging area

git commit -a -m ”commit\_message”

//to view changes made

git diff

//to view changes made in staging area

git diff –staged

// remove file from git repo

Git rm –cached sample.txt

**Git hub-remote repository**

* **Web based git repository hosting service**

// to clone a project

git clone link(the copied link from project)

//create repository

//create folder

mkdir git-course

// to view list of files in the folder

ls

// create a file ,md-mark down

echo “# git-course” >> README.md

//initialize git,add,commit

git init

git add README.md

git commit -m “first commit”

git branch -M main

git push origin main

// to get the changes from rep to local rep

git pull

//get pull changes from main

git pull origin main

// to view all the branches

git branch

git branch -a

//change branch

git checkout gitbranch1

git checkout main

//git check diff from main branch

git diff gitbranch1

//git merge

git merge gitbranch1

//then push it to main

// create new branch

git branch newbranch2

//add a file, commit changes, push to newbranch2

//go to main branch

// can add pull request

// main can view the pull request notification and merge pull request and add them to main branch

**Conflicts**

Conflict arises when Git cannot automatically merge changes from different branches due to overlapping modifications to the same part of a file. This situation commonly occurs when:

* Developer A modifies a file in a branch.
* Developer B also modifies the same file in a different branch.
* When Git attempts to merge these branches, it identifies conflicting changes on specific lines of the file.

<<<<<<< HEAD

This is the content modified in the current branch (e.g., main).

=======

This is the conflicting content from another branch being merged.

>>>>>>> branch-name