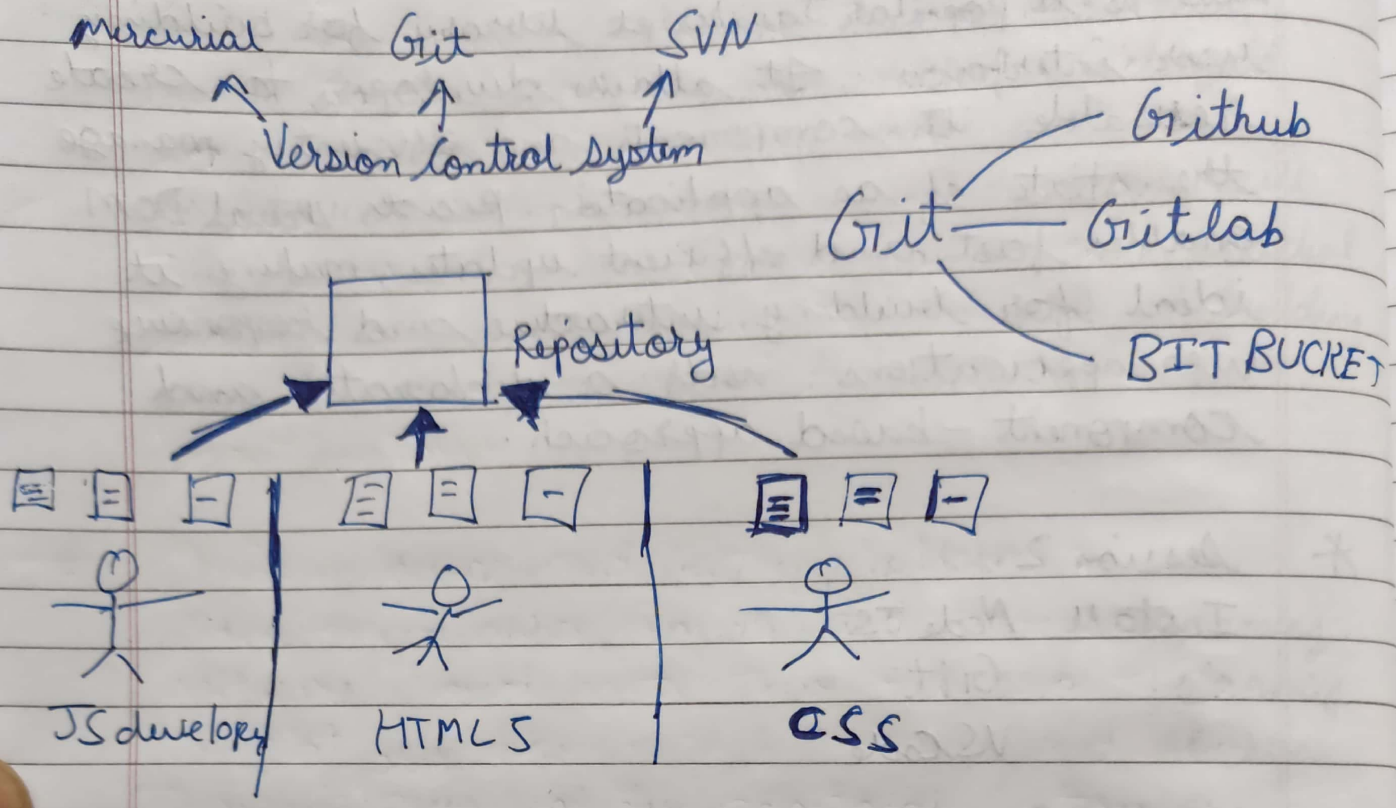
* Git



→ Git is a distributed version control system (VCS) designed to help software developers track changes in their source code and collaborate on projects. It was created by Linus Torvalds in 2005 and has since become one of the most widely used version control system in the software development industry.

Here's how Git works and is used for
1. Version Control
2. Distributed System
3. Branching and Merging
4. Collaboration
5. History and Accountability
6. Open Source and Hosting Platforms.

→ ① Version Control :
Git allows developers to track changes to their codebase over time. Every time a change is made, Git records a snapshot of the entire project. This makes it easy to compare different versions of the code, revert to previous state and understand how the code has evolved.

→ ② Distributed System
Unlike centralized version control systems, Git is distributed. This means that every developer working on a project has a full copy of the repository on their local machine. This local copy enables developers to work offline, make changes, commit them, and later sync those changes with the central repository or other developers repository.

→ ③ Branching and Merging
One of Git's powerful features is its ability to create branches. A branch is essential a seperate line of development. This allows developers to work on new features or bug fixes in isolation without affecting the main codebase. Once a feature or fix is complete, the branch can be merged back into the main codebase.

→ ④ Collaboration → Git facilitates collaboration among multiple developers. Each developer can work on their own branch, and when ready, they can merge their changes into shared branch, such as the "master" branch. Git's branching and merging capabilities make it possible for teams to work on different features concurrently and integrate those features smoothly.

→ ⑤ History and Accountability :
Git maintains a detailed history of all changes made to the codebase. This includes who made each change, when it was made, and the purpose of the change (commit messages). This historical record is valuable for tracking down bugs, understanding the evolution of the project, and reviewing the work of contributors.

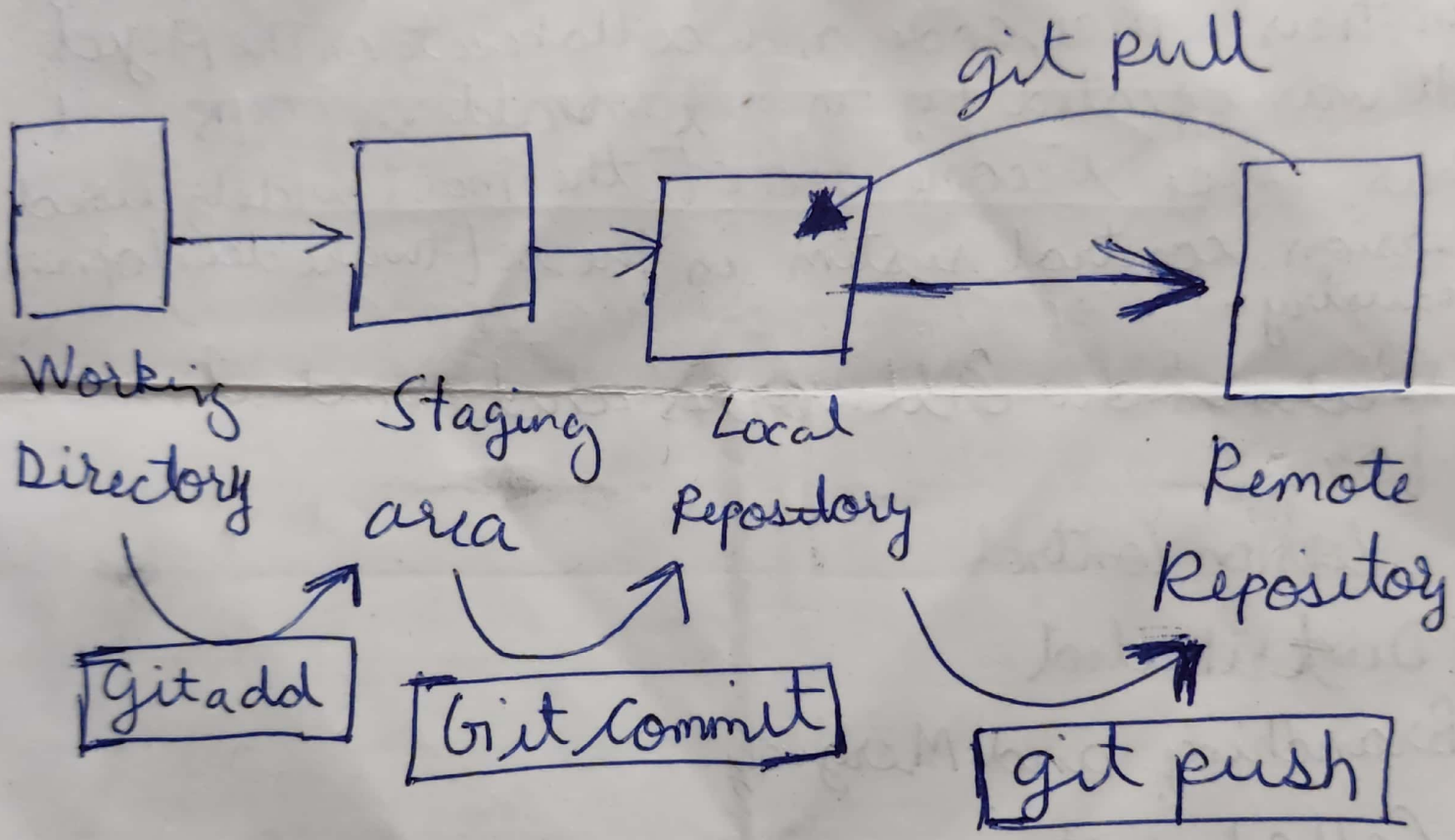
→ ⑥ Open Source and Hosting Platforms :
Many open-source projects and commercial software projects hosts their Git repositories on platforms like GitHub, GitLab and Bitbucket. These platforms provides additional features such as issue tracking, code review, continuous integration, and more, which enhance the development process.

Operation (1)

Cloning a repository =

git clone <url>

* .git → Local repository



+ Stage commit and enter msg ... dropdown ↓ Push.

* What is Git?

Free & Open Source Version Control System

↓

tools that help to track changes in code

helps in

① track history

② help to collaborate

* What is Github

→ Website where we host repositories online

* Imp points while creating a Repo

→ Private / public [Select acc to Requirement]

→ Select Add Readme file

README.md

↓ project → mark down

* Using Git

Can be used through

→ Command Line

→ IDE / Code Editors

→ Graphical User Interface (like GitKraken)

To check cmd

↓

git --version

* Configuring Git

① git config --global user.name "My Name"

② git config --global user.email "--email--"

Check this by
git config --list

* Clone → Cloning a repository on our local machine.
status → displays the ~~status~~ state of the code.

git clone <- some link >
git status

* add → adds new or changed files in your working directory to the Git staging area

| git add <- file name -> |

git add . means adds all

* commit → it is the record of change

| git commit -m "some message" |

* push — upload local repo content to remote repo

| git push origin main |

* Git Init
  - .git → local repository

* Pushing Local Repository

→ init → used to create a new git repo
  git init
→ mkdir <Folder Name>
→ git init
→ git remote -v (to verify remote)
→ git remote add origin <- link ->
→ git remote -v
→ git branch (to check branch)
→ git branch -M main (to rename branch)
→ git push origin main

* Branch Commands

→ git branch

→ git branch -M main (to rename branch)

→ git checkout <- branch name -> to navigate

→ git checkout -b <- new branch name -> to create new branch

→ git branch -d <- branch name -> to delete branch

* Merging code

git diff <- branch name ->

git merge <- branch name ->

Or

Pull request