Setup

* Git -- It checks installed or not
* Git config --global user.name “Susmitha789257”
* Git config --global user.name --> to check username
* git config --global --unset user.name --> delete user.name
* Git config --global user.email [“atigaddasusmitha789@gmail.com”](mailto:\“atigaddasusmitha789@gmail.com\”)
* Git config --global user.email --> to check email
* git config --global --unset user.email --> delete user.email
* Git config --list --> to see all config globals
* git config --list --show-origin
* git config --unset user.name
* git config --unset user.email
* git config --system --unset user.name
* git config --system --unset user.email
* Git config --global --add safe.directory E:/
* git config --global --unset-all safe.directory

Inside New folder

* Git clone <https://github.com/Susmitha789257/timesheet.git>
* We cant modify github file not available in local pull it after delete in local push after deleted in git.
* Incase deleted in local not pushed but I need again that test file but here pull again not works we need use restore.
* Git restore test.txt
* Git log (to come out “q" for long logs / github - commits)
* git log --oneline
* git blame test.txt — we can see when, and what line are modified--> after many commits we use blame.
* Git pull.
* Cd foldername.

New file creation test2

* Git status
* Git add test2.txt
* Git commit -m “test2.txt added”
* Git push origin main

Delete test3.txt

* Git status
* Git add test3.txt
* Git commit -m “deleted”
* Git push origin main

Github to local

* Git pull

Local to git

* Cd ..
* Cd newfoldername
* Git init
* Git add .
* Git status
* Git commit -m “created index.html and style.css”
* Git remote add origin https://github.com/Susmitha789257/JobPortal.git
* Git branch(\*master)
* Git push origin master
* Git branch -m main (modify branch name)
* Git branch
* Git push origin main

Branch

Git branch -a --> all local and remote branches

* Git checkout branchname --> switch one branch to another branch
* Git checkout main
* Git diff branch1
* Git merge branchname
* Git push origin main

Pull request

* Clear (in vscode)
* Git branch newbranch (local creation branch)
* Q for typeing if stuck
* Git checkout newbranch
* Git push origin newbranch
* Inside github create pull request

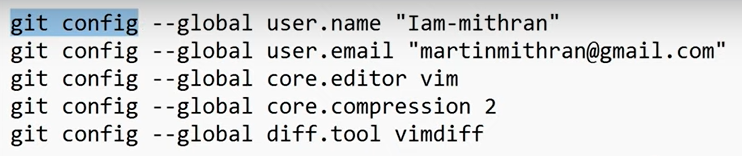
Project

* Git pull origin main

**Learn with Mithran**

**Part-1**

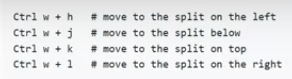
* **Cloud and devops important to apply many positions.**
* **GIT --> Source control management --> Distributed version control system**
* **Plubic cloud -aws, azure, goggle…**
* **Git --> check git installed or not**
* **Git --version**
* **Git status -s --> -s is for short status**
* **Git reset abc.txt (to untrack)**
* **Git reset --> to untrack all the files**
* **Git add abc.txt myfile.txt**
* **Git commit -m “this is my first commit” (-m means commit message)**
* **Git log**



* **Git config --global user.name “Susmitha789257”**
* **Git config --global user.email [“atigaddasusmitha789@gmail.com”](mailto:\“atigaddasusmitha789@gmail.com\”)**
* **Git log --oneline**
* **Git remote add origin <https://github.com/Susmitha789257/abc.git>**
* **Git push -u origin master (set default upstream branch)**
* **git branch --unset-upstream --> unset upstream**
* **Git branch**
* **Git branch develop --> create new branch**
* **Generate token --> go to github console - top right side profile click settings - left down side developer settings - personal access tokens - tokens classic - generate new token classic - access to all .**
* **Git checkout develop or Git switch develop --> both used for change branch**
* **Git commit -a -m “hi” --> a means add**
* **GIt VISUALIZING (visualizing-git) --> <https://git-school.github.io/visualizing-git/>**
* **Git checkout -b dev or git switch -c dev --> create & switch**

**Part-2**

* git config --global init.defaultBranch main --> local repo default branch main othewise master
* Git branch -m “newbranch” --> rename the branch
* Git rebase X --> cut copy paste -->from common point to x end point.
* git cherry-pick commit\_id --> copy of commit
* Git commit -am “message” --> if new file a not works.
* Git merge dev --> dev mergeing to master
* Git config --global core.editor vim
* Git config --global core.compression 2
* Git config --global diff.tool vimdiff
* Git mergetool (document goggle)
* Control shift w or ww --> change vim conflit editor



**Direct Fix**

* :diffg RE --> REMOTE
* :diffg BA --> BASE
* :diffg LO --> LOCAL
* :wqa --> save all editos -->write(w), quit(q), all(a)

**Part-3**

* Git fetch --> only download not merge
* Git pull (fetch + merge)
* Git clone
* Git reset HEAD~2
* Git reset --hard or --soft or --mixed(TASK)
* Git revert HEAD~4 (current branch copy)
* Git checkout -b Y commit\_ID
* Git branch -d branch\_name
* Git reflog
* git checkout commit\_ID
* Git tag v1.0.0
* Git stash --> add to bin
* Git stash list
* Git stash pop --> recover from bin

Advanced:-

* .git folder
* Git init --bare devproject.git
* Git clone diarictory location using ip address.
* Touch .gitignore(hidden) --> 3.txt,5.txt
* Git commit --amend --reset-author
* Git commit --global --edit
* Git diff first\_commit second\_commit
* Git branching model ???
* Git merge --sqaush ???
* Completed.

**ChatGpt:-**

* Git status --short or git status -s
* git switch main
* git clone https://github.com/Susmitha789257/sample.git folder --> Modified folder ex:-sample to folder otherwise sample to sample.
* Git remote -v
* Git branch -m develop
* Git branch -m dev develop

1. What is Git, and why is it used in software development?
   1. **Git** is a **distributed version control system** used to track changes in source code during software development. It allows **multiple developers** to work on the same project efficiently by keeping a history of changes, enabling collaboration, rollback, and branching.
   2. **GitHub**, on the other hand, is a **remote hosting service** for Git repositories — a place to store and share Git-tracked code online.

| **Code** | **Meaning** |
| --- | --- |
| ?? | Untracked file |
| M | Modified in working directory but not staged |
| M | Modified and staged |
| A | Added to staging area |
| D | Deleted in working directory |
| D | Deleted and staged |
| R | Renamed and staged |
| C | Copied and staged |
| UU | Unmerged, both modified |
| AA | Unmerged, both added (added in both branches) |
| DD | Unmerged, both deleted |
| AU | Added by you, but modified by them |
| UA | Added by them, but modified by you |
| DU | Deleted by you, but updated by them |
| UD | Updated by you, but deleted by them |
| RM | Renamed in index, modified in working directory |
| AM | Added to index, modified in working directory |
| MM | Modified in both index and working directory |

| **Command** | **Meaning** | **Use Case** |
| --- | --- | --- |
| git add . | Add all changes (new + modified) in **current directory** and subdirs | Most common; quick |
| git add -A | Add all changes **including deletions** | Ensures deleted files are staged too |
| git add -u | Stage **modified and deleted**, but not new files | Useful if you only want tracked files |
| git add <file> | Add a **specific file** | Good for selective commits |
| git add \* | Add all visible files (no dotfiles) | Less safe, may skip hidden files or directories |

| **Command** | **What it does** |
| --- | --- |
| git commit | Opens default editor to type a message manually |
| git commit -a -m "msg" | Automatically **stages modified & deleted files** and commits — **but not new files** |
| git commit --amend | Modify the **last commit** (message or contents) |
| git commit -m "msg" --no-verify | Skips pre-commit hooks (like linting) |

| **Command** | **Meaning** |
| --- | --- |
| git push origin main | Push commits to origin remote's main branch |
| git push | Push to the **tracked upstream branch** |
| git push -u origin main | Push and also set the upstream for future git push |

| **Command** | **What it does** |
| --- | --- |
| git branch -d branch-name | Deletes the branch **safely** (only if it’s fully merged) |
| git branch -D branch-name | **Force deletes** the branch, even if it’s not merged ❗ |

| **Command** | **Description** |  |
| --- | --- | --- |
| git remote | Just shows the remote names (e.g. origin) |  |
| git remote show origin | Shows detailed info about origin, including branches, fetch/push URLs, etc. |  |
| git remote add <name> <url> | Adds a new remote |  |
| git remote remove <name> | Removes a remote |  |

| **Command** | **What it compares** |
| --- | --- |
| git diff | Working directory vs. last commit (only unstaged) |
| git diff --staged or git diff --cached | Staged changes vs. last commit |
| git diff HEAD | All changes (staged + unstaged) vs. last commit |
| git diff branchA..branchB | Differences between two branches |