**GIT :**

Git is a distributed version control system that helps developers track changes to files, collaborate on projects, and manage versions of their code. It allows users to merge changes, revert to previous versions, and manage multiple branches of development, making it a powerful tool for collaborative and large-scale software projects.

**Github** : GitHub is a web-based platform for version control and collaboration, built around Git. It allows developers to store, manage, and track changes to code, enabling efficient teamwork. GitHub supports features like pull requests, issue tracking, and project management. It's widely used for both open-source and private software development.

**Stages of Git:**

The files in git undergoes through three stages

1. **Untracked Area:** The File Exists, But its not part of git’s Version Control system
2. **Staging Area:** The file has been added to git's version control but changes have not been committed
3. **Committed Area**: The change has been committed

To check the Status of file we use **git status**

**Alternatives For GIT:**

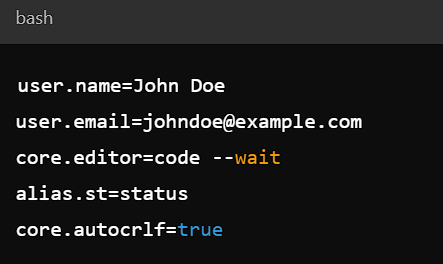
There are many alternatives for GIT like Mercurial , Apache Subversion(SVN)

Gitlab , BitBucket , BeanStalk, Launchpad, Google Cloud Source Repositry…etc

**Git Config Commands**

The git config command is used to set configuration options for Git. These settings can be applied at different levels (local, global, or system) to customize the behaviour of Git

1. **git config --global user.name “Your Name”** 🡺Sets the user Name Globally
2. **git config –global user.email “Your Email” 🡺** Sets the mail Globally
3. **git config user.name “Your name” 🡺** Sets the name for specific repository
4. **git config user.email 🡺** Sets the email for a specific repository
5. **git config --global –list 🡺** Returns the all configurations made by the user in a key value pair format



1. **git config --global alias.<alias\_name> <command>** 🡺 Aliases can be defined to shorten commonly used Git commands.

Ex: git config –global alias.st status 🡺Creates an Alias for git status command.

**Git Remote Commands:**

1. **git remote:** Lists all the remote repositories linked to your local repository.
2. **git remote add <remote\_name> <remote\_url>:** Adds a remote repository to local repository

**Ex:** git remote add origin <https://github.com/pavanummagani1/practice.git>

1. **git remote remove <remote\_name> :** This command removes a remote repository.After removing it we can’t pull or push the code into the removed repository
2. **git remote rename <old\_name> <new\_name> :** This command renames the repositories

**Ex:** git remote rename origin upstream 🡺 It renames the remote repository from origin to upstream

1. **git remote set-url <name> <new\_url> 🡺** Changes the URL of remote repository.
2. **git remote show <name> 🡺**Shows all details about the repository
3. **git fetch <remote name> 🡺** Fetches the changes from remote repository with out merging it
4. **git pull <remote\_name> <branch\_name> 🡺** Fetches and merges changes from a remote repository branch.
5. **git push <remote\_name> <branch\_name> 🡺** Pushes the local changes to remote repository branch
6. **git remote prune <branch\_name> 🡺** Removes the references to branches deleted on a remote repository

**Git branch Commands**

1. **git branch 🡺** Lists all the local branches in repository
2. **git branch -a 🡺** List all the local and remote branches
3. **git branch <branch\_Name> 🡺** Creates a new branch
4. **git branch -d <branchName> 🡺** Delete a branch. The branch must be fully merged in its upstream branch, or in HEAD
5. **git branch -D <branchName> 🡺** Deletes the branch **forcefully**, even if it is not merged
6. **git checkout -b <branchName> 🡺** Creates a branch and switches to the created branch
7. **git checkout <branchName> 🡺** Switches to the given branch
8. **git branch -m <new\_branch\_name> 🡺** Renames the current branch to a specified name
9. **git branch -m <old\_name new\_name> 🡺** Renames the specefied branch
10. **git branch -v 🡺** Lists the last commit for each branch
11. **git branch –contains <commit\_hash>** 🡺 Lists the branches that has specified commit message

**Git add commands**

1. **git add <filename> 🡺** Add a Specific File to Staging area
2. **git add . 🡺** Add all changed files in the current directory to Staging area
3. **git add <directory>/** 🡺 Adds all form a specific directory to staging area.
4. **Git add -A/--all 🡺** Adds all changes in the entire repository, including untracked files.
5. **git add -u 🡺** Adds only the modifications to already tracked files, ignoring untracked ones.
6. **git add -p or –patch 🡺** Interactively add portions of files to the staging area.

**Git Push commands**

1. **git push <remote\_name> <branch\_name> 🡺** Pushes the code from Local branch to remote branch
2. **git push -u <remote\_name> <branch\_name> 🡺** Pushes and sets the upstream branch for easy future pushes and pulls.
3. **git push –all 🡺** Pushes all branches to the remote repository.
4. **git push –delete <branch\_name> 🡺** Deletes a branch from the remote repository.
5. **git push –force 🡺** Overwrites the remote branch with local changes, ignoring any differences.

**Git Pull Commands**

1. **git pull 🡺** Fetches and merges changes from the remote repository to the current local branch.
2. **git pull <remote\_name> 🡺** Fetches and merges changes from the specified remote repository to the current branch.
3. **git pull –quiet 🡺** Suppresses output messages during the pull process.
4. **git pull --ff-only 🡺** Only allows fast-forward merges (no merge commits)
5. **git pull –verbose 🡺** Displays additional information during the pull process.

**Git init commands**

1. **git init 🡺** Initializes a new Git repository in the current directory (creates a .git folder)
2. **git init <directory>🡺** Initializes a new Git repository in the specified directory (creates a .git folder in that directory)
3. **git init –bare 🡺** Initializes a bare Git repository (used for shared repositories, without working directories)
4. **git init --separate-git-dir <path> 🡺** Initializes a repository, but stores the .git directory in a different location.
5. **git init –template ==<template> 🡺** Initializes a repository using specified template directory, customizing hooks or config.

**Upstream Commands**

1. **git push –set-upstream <remote> <branchName> 🡺** Sets the upstream (tracking) branch for the current local branch. This allows for simplified future pulls and pushes.
2. **git branch --set-upstream-to=<remote>/<branch> 🡺** Sets the upstream branch for an existing local branch to track a specific remote branch**.**
3. **git remote show <remote>** 🡺 Displays detailed information about the upstream branch, including its tracking configuration.
4. **git fetch <remote> 🡺** Fetches updates from the upstream remote repository without merging them into your local branch.

**GIT Shortcuts**

1. git st 🡺 git status
2. git br 🡺 git branch
3. git co 🡺 git checkout <branch>
4. git ci 🡺 git commit
5. git ci -m “message” 🡺 git commit -m “message”
6. git unstage 🡺 git reset HEAD <file>
7. git amend 🡺 git commit –amend
8. git po 🡺 git push origin
9. git pom 🡺 git push origin main
10. git pu 🡺 git pull
11. git df 🡺 git diff