Git

Version Control

- Login to your account by
  `git config --global user.email`you@example.com

- First run `git init` for Project in terminal to Initialize your Repo

- `Git status` to show the status of files
  `Git status -s` for short brief
- use `git add <filename>` or `git add .` for add all files to Stage Area **[make files TRACKED]**

- If files with green color that's mean the files added Successfully
- Confirm changes by `git commit -m "message"`
- Display all Tracked Files in Staging Area `git ls-files`
- display all commits of repo

  `git log`

  `git log –oneline` for shot brief

- display all commits from local & remote

  `git log --oneline --all`

- move form Version to another use

```
Git checkout <commit ID>
```

  Every commit has unique ID use it to select specific Version

- Make new Branch

```
Git checkout -b BranchName
```

- Display all Branches

```
Git branch
```

- Change between Branches

```
Git checkout BranchName
```

- For prepare Project upload to Repository

```
git remote add origin RepositoryLink
```

- Upload Project

```
git push -u origin BranchName
```

- get SHA1 for file

```
git ls-files -s
```

- get type of object {blob / tree}

```
git cat-file -t SHA1 for Object
```

- get size of object {blob / tree}

```
git cat-file -s SHA1 for Object
```

- get content of object {blob / tree}

```
git cat-file -p SHA1 for Object
```

- Display difference between Working tree and Staging Area

```
Git diff
```

- Display difference between Staging Area and Git Repo

```
Git diff -staged
```

- Display content of Commit

```
Git show sha1 of commit
```

- Display diff between two commits

```
Git diff shaOne..shaTWo
```

```
Git diff vf45..hy56
```

- Remove git repo

```
Git rm -rf .git/
```

- Discard changes [restore last commit to working tree]

```
Git restore filename
```

- Un Staged

```
Git restore --staged fileName
```

- Make add and commit is same time

```
Git commit -am “Message”
```

- Edit commit message

```
Git commit –amend
```

- To change version of file [Moving between commits]
- change Head [retrieve commit to Staging Area]

`Git reset Head~1` back forward 1 step

- To Directly retrieve commit in working tree [be Carful]

```
Git reset --hard Head~1
```

- Display all commits even hidden ones

```
Git reflog
```

```
Git tag -a TagName -m “Message”
```

- Display specific Tag

```
Git show TagName
```

- Create new branch

```
Git branch Name
```

- Display all branches

```
Git branch
```

- Change current branch

```
Git switch branchName
```

- Merge to master branch
  Head must be on Master before Merge

```
Git merge suppBranch
```

- Display branches that merged to Master

```
Git branch –merged
```

- Delete branch

```
Git branch -d name
```

- Clone Repo

```
Git clone SourcePath newname
Git clone github/1234 myCloneRepo
```

- Display if repo from remotely [origin]

```
Git remote
```

- Display details about remote repo

```
Git remote -v
```

- Display remote branches

```
Git branch -r
```

- Clone form another remote

```
Git remote add Name path
```

- Fetch changes from remote but not merge to working tree

```
Git fetch remoteName
Git fetch origin
```

  Then `git merge` to update working tree

- When you create new branch in local you need to make opposite branch In remote to make operations on it like pull / push

```
Git push -u remoteName BranchName
Git push -u origin  feature
```

  Will create feature branch in (origin / remote) repo then sync with feature branch in local repo

- Pull = Fetch + merge in one command

```
Git pull remoteRepo you will fetch from
Git pull origin
```

- Display list commit in different branches

```
Git branch -v
```

- Display tracked branches

```
Git branch -vv
```

Terms :

Git: version control

GitHub: hosting service for git

Branch : Linear Order of Commits

Tag : make specific Commit special

Fork: make a Copy form GitHub Repo to your GitHub Account

Clone : it to download repo without close the connection to this repo

Push : upload your changes to cloned repo

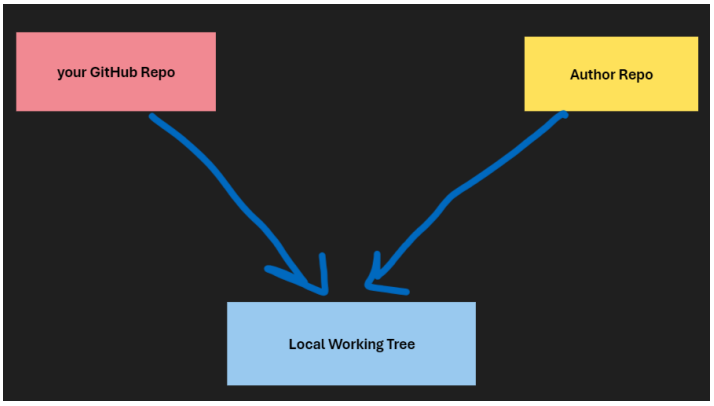
Pull : fetch new changes from remote repo to cloned / local repo

SSH authentication :

You can access and write data in repositories on GitHub.com using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine.

Contribute Workflow:

1- Fork Repo
2- Clone Repo to you local. [add remote from GitHub Repo to you Local]
3- Add new remote to local Repo this remote must be the Author Repo you take Fork from him



4- Make some changes [commits]
5- Push new commits to your GitHub Repo
6- Create **Pull Request** to Author so he will accept your changed or close pull request.