# Git

Git is a **version control system** i.e. to track the changes in the code. It is also free and Open source.

Not only that it can also be useful for collaboration with teams.

In Windows, it can be used through Git Bash, where the commands are similar to Mac's.

**README.nd** is a special file that describes your GitHub Repo. It can be edited through HTML tags.

**Commit**: It is a message that is displayed whenever a change occurs.

**Modified files**: The files where modification has taken place.

**Untracked file**: Newly created files that were not a part of the initial Git Repo.

**Staged files**: These files were added but have yet to be committed.

**Branch**: When a project is been developed by many people, branches can be utilized for faster development. And when the branches are merged, their code bases become the same.

**Pull Request**: Tells others all the changes that we have pushed to a branch in a repo in GitHub. First, a Senior would check the PR and give a review on it i.e. is it suitable or not then further actions take place.

**Merge Conflicts**: When Git can’t resolve the differences in code across branches automatically, we will manually resolve the changes.

## Git Workflow

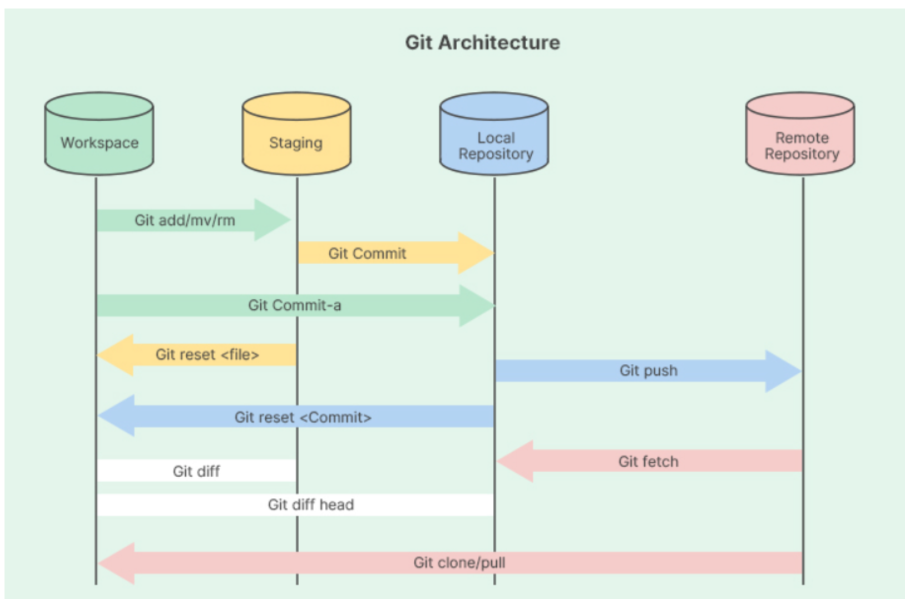
GitHub Repo ——> Clone ——> changes ——> add ——> commit ——> push

## Git Architecture

Workspace is the location where we perform our tasks i.e. changing files, adding new files, etc. It is basically the location where the user works.

The Staging Area is the location where the files are tracked and acts as an interface between the Local Repo and Workspace.

Local Repo is the Repo that is created on your system like a copy/replacement of the Remote Repo. All the changes are first sent into the Local repo, then into the Remote Repo.



# Git Commands

1. To check if it is properly downloaded or not. `git --version`
2. To Configure i.e. link your PC's Git to GitHub:
    a.
    `git config --global user.name "my name"`
    b.
    `git config --global user.email "my_name@gmail.com"`
    c.
    `git config --list`
3. To Clone a Repo. `git clone <link>` (Here link is HTTPS Link)
4. To display the Status `git status`
5. To view hidden files `ls -a`
6. a. To add files to the working directory in the **Git Staging area** `git add .`

    b. To commit files i.e. to record changes `git commit -m "some message"`
7. I) To push from local Repo to remote Repo `git push origin main`.
    II) If you want the Git to remember that only one branch is used for a long time
    `git push -u origin main`.
    Then only use
    `git push`.
    III) Here "origin" is the name given by us to the repo present at GitHub and is pushed to the "main" branch.
8. a. To create a new remote Repo `git init`.
    b.
    `git remote add origin <link>` and to verify remote `git remote -v`.
    c. Now check which branch we are in. If it says "master", change it to "main".

    `git branch` `git branch -m main`
    d. Now push it to the Remote Repo.
9. To Check all the Commits `git log` and to exit it use `q`.

## Branches

1. Branch Commands:
    i) To check on which branch we are present
    `git branch`

    ii) To rename a branch `git branch -m name`

    iii) To navigate between branches `git checkout <branch name>`

    iv) To create a new branch `git checkout -b <branch name>`

    v) To delete a branch `git delete -d <branch name>`. You cannot delete the branch that you are present in.
    vi)
    `git push origin <branch name>`
    vii) To compare 2 branches
    `git diff <branch name>`

2. Merging Branches:
   a) Through Pull Request i.e. directly from GitHub
   b)
   `git merge main` Here “main” is the other branch and not the one currently working in. The branch in which we use this only will change and not the “main” branch.

3. To fetch and download content from remote Repo `git pull origin main`

## Undoing Changes

1. Staged Changes i.e. added but not committed `git reset <file name>` or `git reset`.
2. Committed Changes `git reset HEAD~1`. Here HEAD is the name of the last change by default and we are saying to change the HEAD by going back one Commit.
3. Committed Changes but multiple commits `git reset <commit hash>`. Here, the Commit hash is found through `git log`.
4. All the above is for git only not VS Code i.e. the actual files, for that use `git reset --hard <commit hash>`.

## Fork

It is a new repo with the same code and settings as the original Repo. It can also be stated as a Rough Copy.

It is for copying other’s code and modifying it. Useful for Open Source Contributions.

Click on the
**Fork Option** when viewing another Repo and a copy of the repo will be created in your account.

And then after changing click on **Pull Request** and you will be able to Merge the codes.

## Git Stash

This is a useful feature wherein you can save the changes without adding and committing.
Sometimes we do not complete the changes in a branch in a particular time and we also do not want to commit half the changes i.e. unnecessary commits so we use this feature.

`git stash` to stash the changes.

`git stash pop` to remove the most recent stash change.

`git stash list` to view all stashes applied.

`git stash apply <stash code>` to apply specific stash through the stash code (found through `git stash list`).