Version control

- Make save points that saves your project.
- Provides total development freedom.
- eg. git, subversion, peforce.
- Git is a version control tool.
- Github is a service that hosts git repositories (collaboration).

---

git config --global user.name " "   (to set user name)

git config --global user.email " "   (to set email)

Creating a __git repo__
↳ simple folder with additional capabilities
⬇
version control

pwd   (present working directory)

cd <path>   (change directory to given path)

ls   (lists files/directories in present directory)
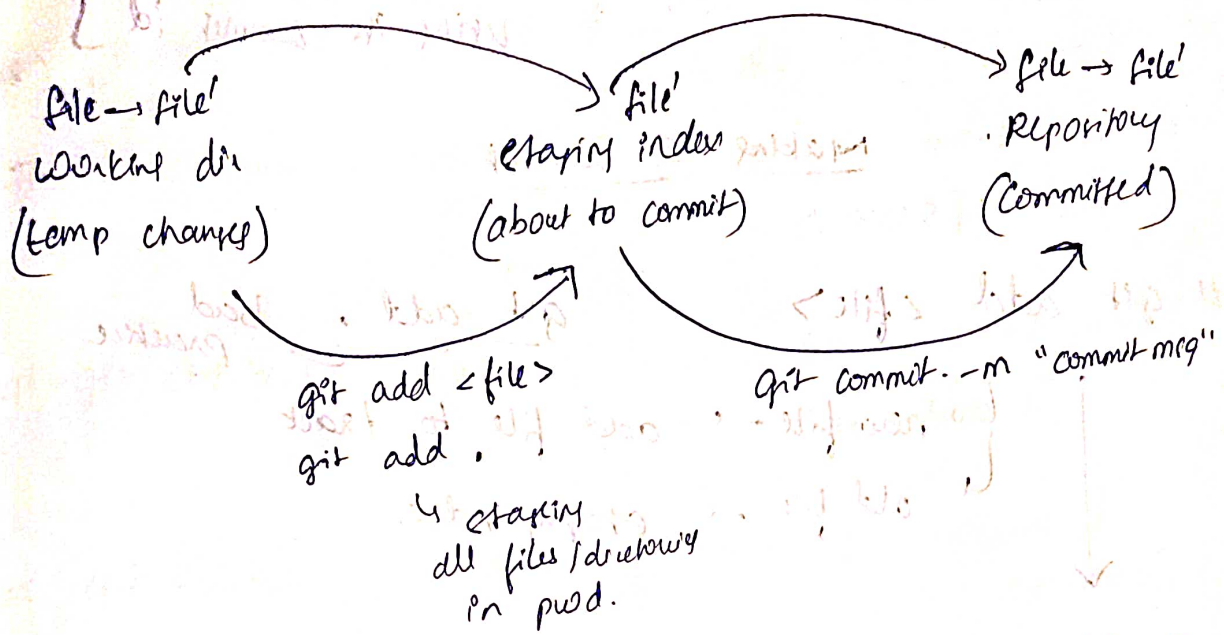
git status   (gives the status of a git repo
               lists untracked changes if any)

git init   (initializes a git rep in a folder
               to make it a repository)

git clone <github repo> <new name>   [cloning a remote git repo to your local machine]

git diff   [lists changes/modifications from the last commit]

Life cycle of a change

file → file'
working dir
(temp changes)

→ file'
staging index
(about to commit)

→ file → file'
. Repository
(Committed)

git add <file>
git add .
↳ staging
all files / directories
in pwd.

file → commit id   →   file' → commit id
        ↓   ↓
       ABD   ABC2

→ Every commit in a git repo has a commit id.

Reviewing a repo history

\# git log   (Lists all the commits made on a git repo
               with the latest commit being on top)

For each commit, the git log command shows
commit id
Author
Date & time

\# for latest n commits :-
    git log -3   (latest 3 commits)

\# git log -p (shows latest commits with diff.)

\# git log --oneline (Commits are shown in one line
                with commit id & commit msg)

\# git log --stat (shows latest commits along with
                changes in files form file names)

\# git show <commit id>   [shows changes made in a particular commit using the commit id]

Making a commit

\# git add <file>   git add ; → bad practice

      ↳ new file → add file to track
      ↳ old file → staging index

\# git commit -m "commit msg"

[HEAD → pointer, points to the latest commit]

\# git restore <file>
     ↳ restores file to latest working commit
           ↓
           HEAD

\# To unhack changes in a git repo

  → create a .gitignore file
  → include all the file names / patterns / paths to the .gitignore file that you want git to not track.

  → commit .gitignore

\# git commit -am "message" [to add & commit simultaneously]

Branching

git branch   [Lists all branches in a repo
                  & tells in which branch we
                  are in]

git branch <name>   (Creates a new branch named
                        <name>]

git checkout <name>   [Switches from current branch
                          to <name> branch]

git checkout -b <name>   [creates & switches to
                             a new branch <name>]

git merge <name>   (merges <name> branch into
                       current branch]

git branch -d <name>   [deletes <name> branch]
                         [after working on a
                           branch & merging it to
                           master, that branch can
                           be deleted]

git merge --abort   [abort the merge when there
                        are conflicts in merging]
                                  (OR)
                      [resolve conflicts and commit]

Tagging

git tag -a <tagname> <commit id> -m "msg"
          [used to tag a particular commit]

git tag -d <tagname>
          [to delete the tag over a commit]

Stashing

- There might be possibility that the remote repo is updated and is different from your local repo.

In this case, you can stash your local repo before pulling the remote repo.

stash - stores your changes temporarily which can be later merged into the remote pulled repo.

\# git stash

\# git pull [pulls the remote repo locally]

\# git stash list [stash area list]

\# git stash apply [to apply your changes after pulling the remote repo]

↓

resolve merge conflicts

↓

add & commit.

Note: Master is the super stable & super maintained branch.

Always create branches first & make changes in the branches.

Merge the branches to master only when it is stable & working.

Undo commits

git commit --amend [amend the most recent
                                    commit]
                    ↑
             making
             changes

git revert <commit id> [reverting given commit
                           with commit history saved]

git reset --soft <commit id> → [saves the
         --mixed   "   present changes
         --hard   "   →   in working
                                 directory]
                               [present changes
                                are shown as
                                modifications]
                          [reset to the given
                           commit id & all current
                           changes are removed].

Pushing commits

git remote add origin <repo link> [links remote repo
                         (remote)   to local repo]

git push -u origin main [pushes the commits
                    ~~master~~   to the remote
                    master   repo
                    branch
                    generally)

So, git push -u origin master

Note: Before pushing to the remote repo,
authentication needs to be done
for the github account from which
you wanna push the changes.