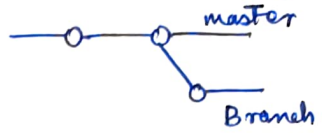
Git

- version control system (VCS)
- Distributed VCS
  - ↳ git doesn't rely on a central server to store all versions of a project's files.
    Instead, every user "clones" a copy of a repository (a collection of files) and has the full history of the project on their own harddrive.
    - this clone has all the meta-data of the original, while the original itself is stored on a self-hosted server or a third-party hosting service like GitHub.

why use Git?

- undo mistakes
- Distributed development
- Don't mix-up things.

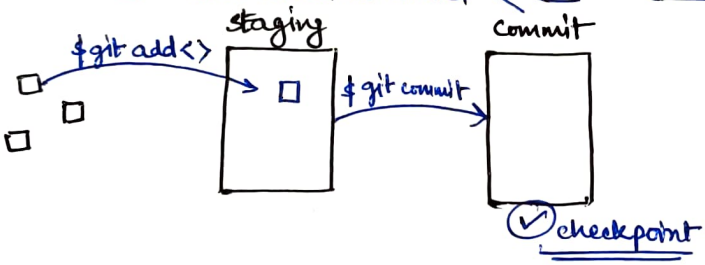


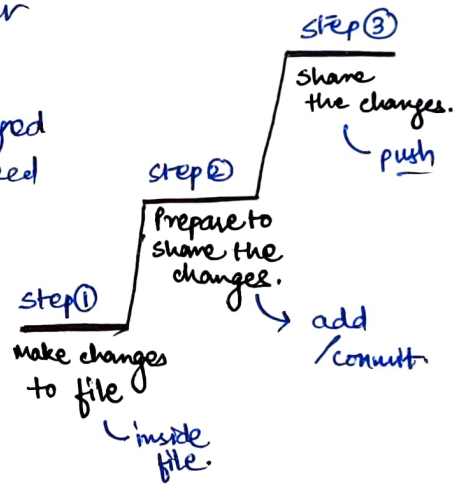
- Strong community support.

```
$ git init      // to initialise git on a local folder
$ git status    // to check status
```

green - added to staged area but yet to be committed.
red - neither added nor committed

↳ lists the files you've changed and those you still need to add or commit.



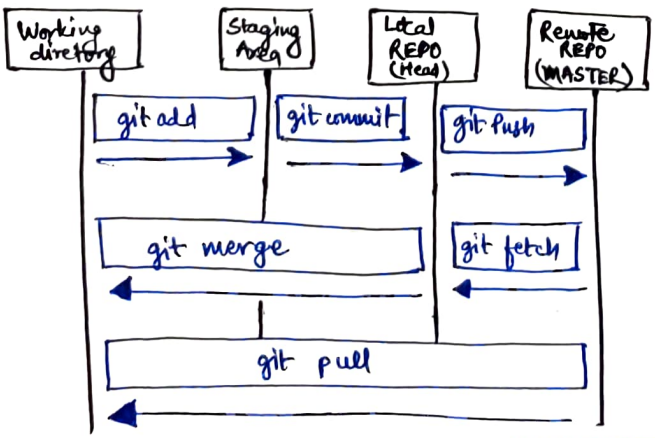


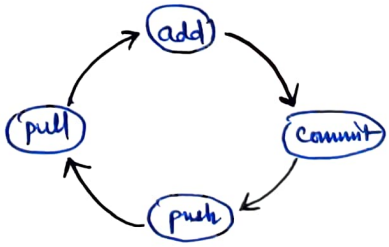

```
$git add <filename>      // add 1 or more file to staging area.
$git commit -m "commit msg"   // commit changes to head. → checkpoint
$git log    // to view all the commits, with "commit messages".
```

displays:
- Secure Hash Algorithm (SHA)
- authors
- date
- commit message.

```
$git log -p / $git log --patch   // to view all commits, and their changes
```

even location of lines that are added or removed.





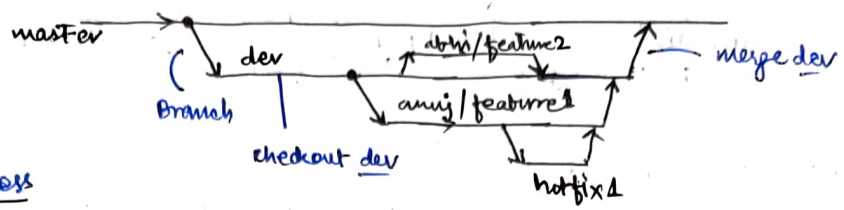
GIT workflow

$ git rm --cached <filename> // to remove files from staging Area (Unstage karna)

$ git branch <branch_name> // to create a new branch with name as "branch_name"

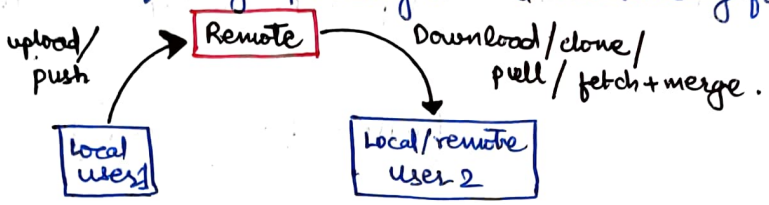
$ git checkout <branch_name> // to switch from current branch to another.

$ git merge <branch_name> // to merge a branch into current branch.



Sharing Process

- Git uses something called a remote (or remote Repo.) to store shared files.
- Developers share files by uploading to and downloading from remotes.



$ git remote add remote_name URL // to register a remote.

$ git push remote_name master // uploading to remote from local — PUSH
(master ↑ local Branch)

$ git pull remote_name. master // downloading from remote to working dir. — PULL

$ git diff // to recognize content changes.

---

step-by-step process

① git init

② git clone remote_url // clone a project of someone to your local directory.

③ git add <filename>

git add -A // add new files, update modified files and remove deleted files from staging area.

④ git status // current status of working tree.

⑤ git diff // show all unstaged changes.

⑥ git commit -m "commit message" // commit staged changes.

~~~

git reset // unstage all files.

git reset file.html // unstage only file.html.

git rm file.html // delete file.html from staging area & working directory.

~~~

⑦ git branch <new-branch> // creates new branch with all the contents of working dir.

OR

git branch <new-branch> f17ac24d // create new branch based on commit.

⑧ git branch -d <branch-name> // delete a branch.

⑨ git switch <branch-name> // switch to specified branch

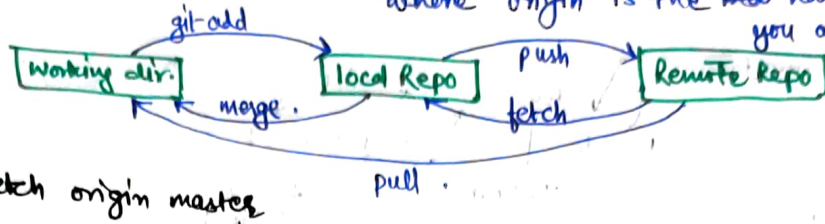
⑩ git checkout <branch-name> . ——— " ———

(11) git log // shows all commits, author name, time.

(12) git remote add origin <repo-url> // add a remote named origin in a remote repository.

(13) git remote -v // verify the remotes.

(14) git push origin master // pushes to a remote master branch, where origin is the name of remote you added.



(15) git fetch origin master
// fetch from Remote REPO. to local Repo.

(16) git pull origin master // pull from Remote Repo. directly to working dir.

OR

[pull = fetch + merge]

(17) git stash // save all the local changes.

(18) git tag <tag-name>

or

git tag <tag-name> <commit-id> // creating a lightweight tag. {just shows commit.}

or

git tag -a <tag-name> -m "tagged message"
// creating an Annotated Tag;
{stores; tagger info, date the commit was tagged, annotation tag message.}

(19) git show // display all the Tags.