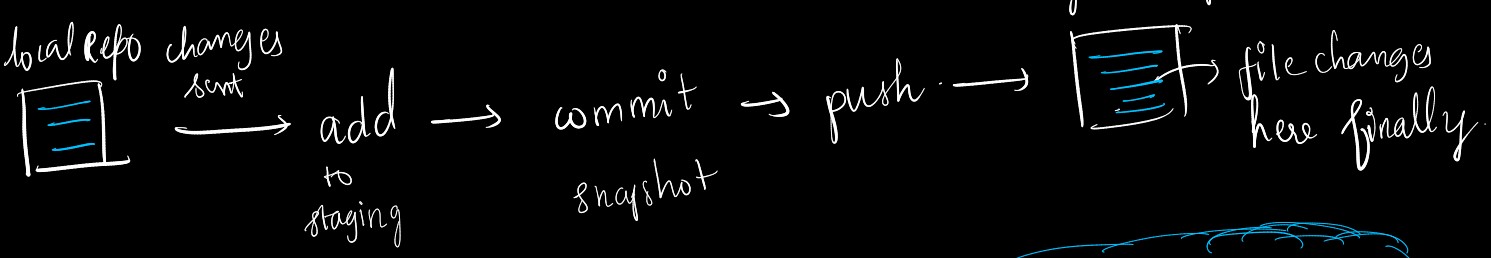
Git Basics:

Why: to maintain, track, share code

- `git init` ⇒ creates a local repo i.e makes a .git folder containing scripts to track changes.
- `git add filename/.` ⇒ adds the file to staging area (staging Area)
  - kind of prepares to commit these files
- `git commit -m "MSG"` ⇒ creates the snapshot of the files in staging area (snapshots)
  - generates hashed objects with key → we can access a specific commit with hashkey
- `git stash filename` ⇒ kind of creates fake snapshot i.e does not commit changes but instead sends them to some place from where they can be retrieved again.
- `git stash pop` ⇒ pops everything from backstage to staging area.
- `git stash clear` ⇒ deletes the fake snapshots.
- `git push origin branchname` ⇒ pushes the commits to github repo
- `git log` ⇒ contains all commits & log files / data

- A basic workflow



Note:

v3 ⇒ contains the complete log until v3 but not v4.

- **Head** ⇒ a pointer to a branch to which all changes will be pushed. It can be changed, initially on main/master
- **git branch name** ⇒ creates a new branch
- **git checkout branchname** ⇒ changes head to that branch

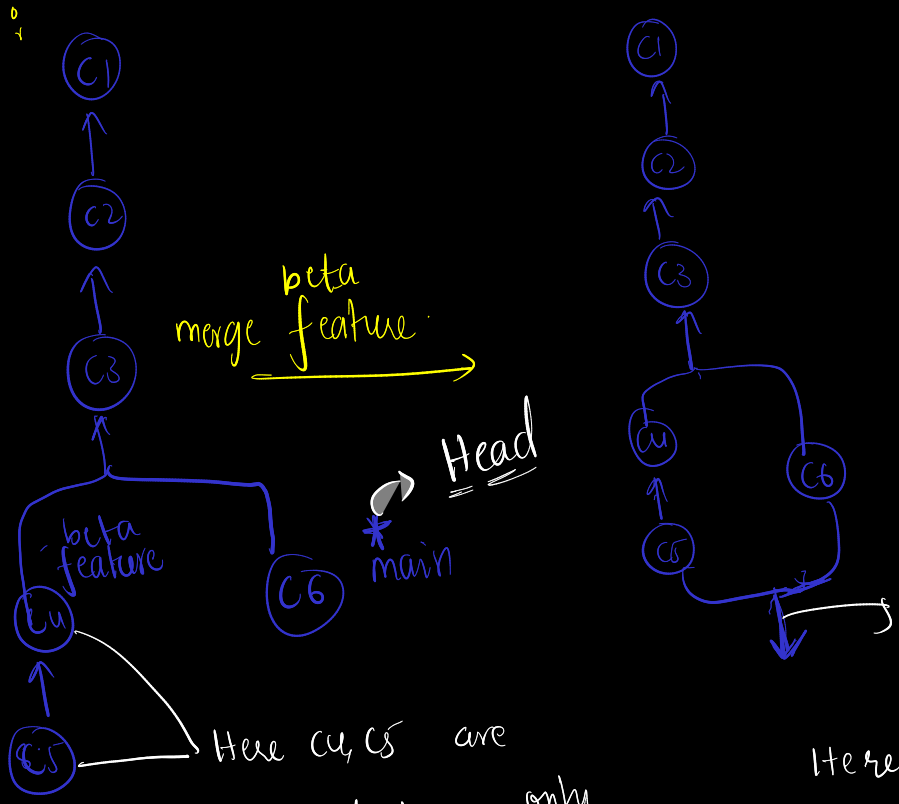
Note: It's always better to push changes of a new feature that might contain errors to non-main branch. As main branch is what users use.

- **git merge branch** ⇒ merges the branch with main branch.

Think of this as beta features coming to stable release in software updates.

C → commit

Eg:



Here C4, C5 are like beta features only few can access

Say this after stable release with no bugs

Here beta feature is merged with main.

- **Contribute to others codebase:**
- Fork the Repo → creates a copy in your account acc/ forked repo

↓

Download Zip
(Set it up locally)

↓

Create a seperate branch for yourself

↓

Make changes in that branch.

↓

Raise a pull request with changed code.

↓

They will accept & Merge the PR. if it's good

⇒ A branch can only raise a single PR. This is the reason to push changes to main branch directly and to use many branches

- `git reset commitkey (hashed)` ⇒ will kind of revert back to that state of project and remove the commits after that.

- `git push origin main -f` (forceful) ⇒ used when commits in our log doesn't match github repo log

- `git fetch --all --prune` (contains deleted commits too)
  `git checkout main`
  `git reset --hard upstream/main`
  `git push origin main` ⇒ keeps the forked local repo with the original main repo.

  OR `git pull upstream main`