HobbyADD

git workflow

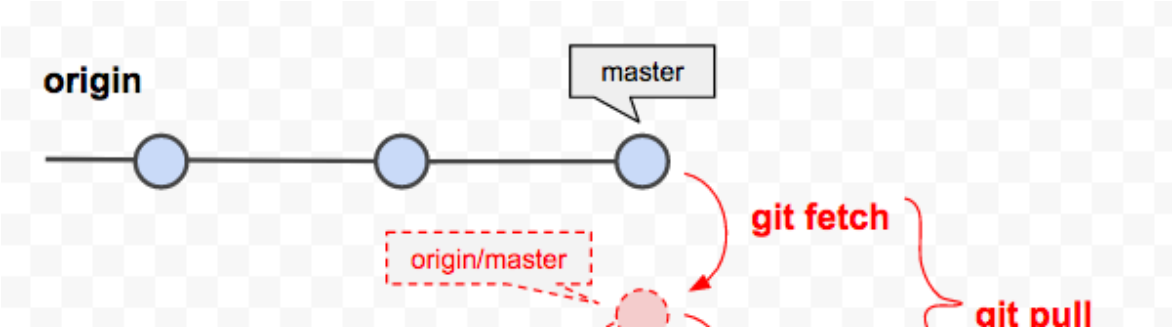
git fetch daily workflow

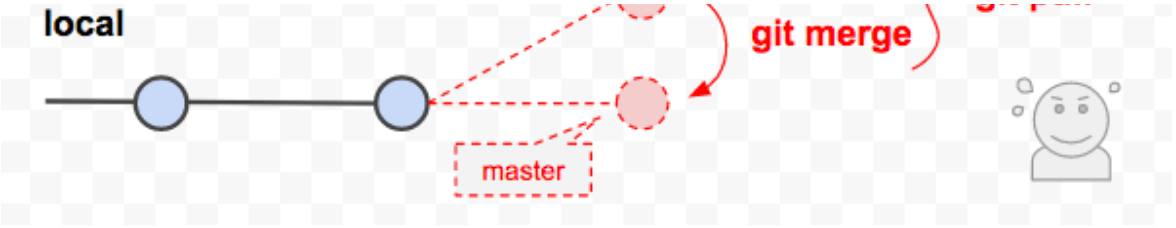
- git fetch origin *(downloads all changes to all remote branches you are tracking into local mirrors of those remote branches)*
- git log –oneline <branch>..origin/<branch> *(shows useful summary of changes you just downloaded so you can decide if you want to merge them into your local branch or not*)
- git checkout <branch> *(ensures you are working on the local branch you want to update)*
- git log origin/master *(final summary of the changes you are about to merge)*
- git merge origin/master *(merge the downloaded changes into your local branch. you are updated! You may have to fix merge conflicts though, if you are unlucky).*

git file structure (**https://www.atlassian.com/git/tutorials/syncing/git-fetch#:~:text=The%20git%20fetch%20command%20downloads,else%20has%20been%20working%20on (https://www.atlassian.com/git/tutorials/syncing/git-fetch#:~:text=The%20git%20fetch%20command%20downloads,else%20has%20been%20working%20on)**.)

- .git/objects –> here, Git stores all commits, local and remote
- Git keeps remote and local branch commits distinctly separate through the use of branch refs. The refs for local branches are stored in the `.git/refs/heads/`
- Remote branches are just like local branches, except they map to commits from somebody else's repository. Remote branches are prefixed by the remote they belong to so that you don't mix them up with local branches. Remote branch refs live in the `./.git/refs/remotes/` directory

git fetch vs git pull





git remote show <remote-name> // tons of information about remote
git remote show origin // <– example

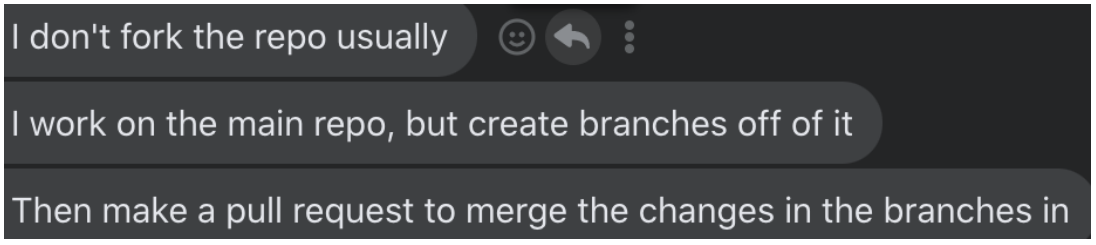
compare local repo's branch with a remote repo's branch:

To compare a local working directory against a remote branch, for example *origin/master*:

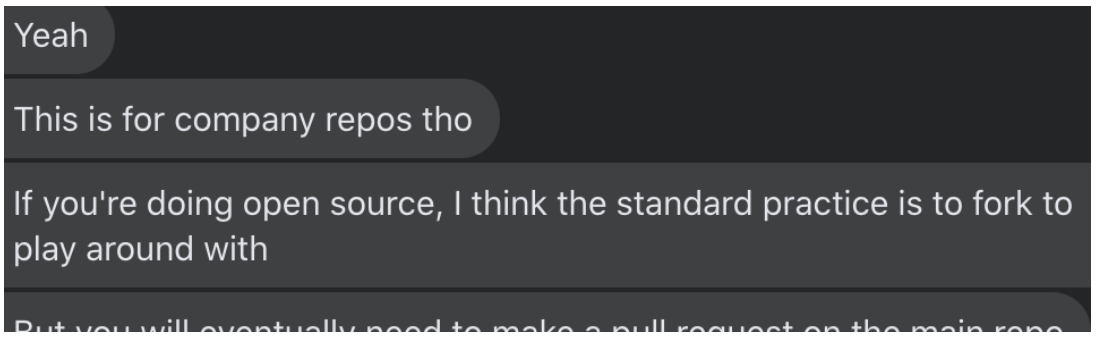
1. ***`git fetch origin master`***
   This tells git to fetch the branch named 'master' from the remote named 'origin'. `git fetch` will **not** affect the files in your working directory; it does not try to merge changes like `git pull` does.
2. ***`git diff --summary FETCH_HEAD`***
   When the remote branch is fetched, it can be referenced locally via FETCH_HEAD. The command above tells git to diff the working directory files against FETCHed branch's HEAD and report the results in summary format. Summary format gives an overview of the changes, usually a good way to start. If you want a bit more info, use `--stat` instead of `--summary`.
3. ***`git diff FETCH_HEAD -- mydir/myfile.js`***
   If you want to see changes to a specific file, for example myfile.js, skip the `--summary` option and reference the file you want (or tree).

what is HEAD? when you make a commit, GIT creates a commit object within the repository. Commit objects should have a parent (or multiple parents if it is a merge commit). Now, how does git know the parent of the current commit? So HEAD is a pointer to the (reference of the) last commit which will become the parent of the current commit.

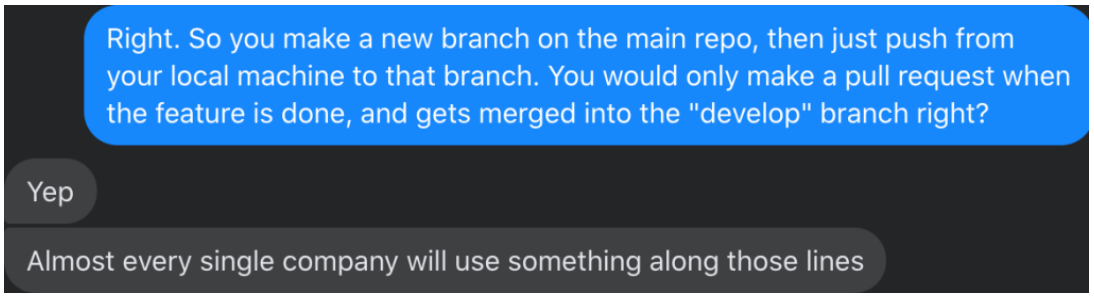
always work on a branch (obviously)



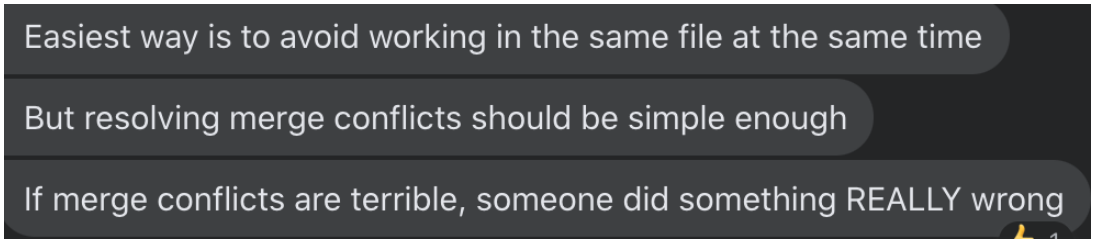
forking repos? not much outside of open source



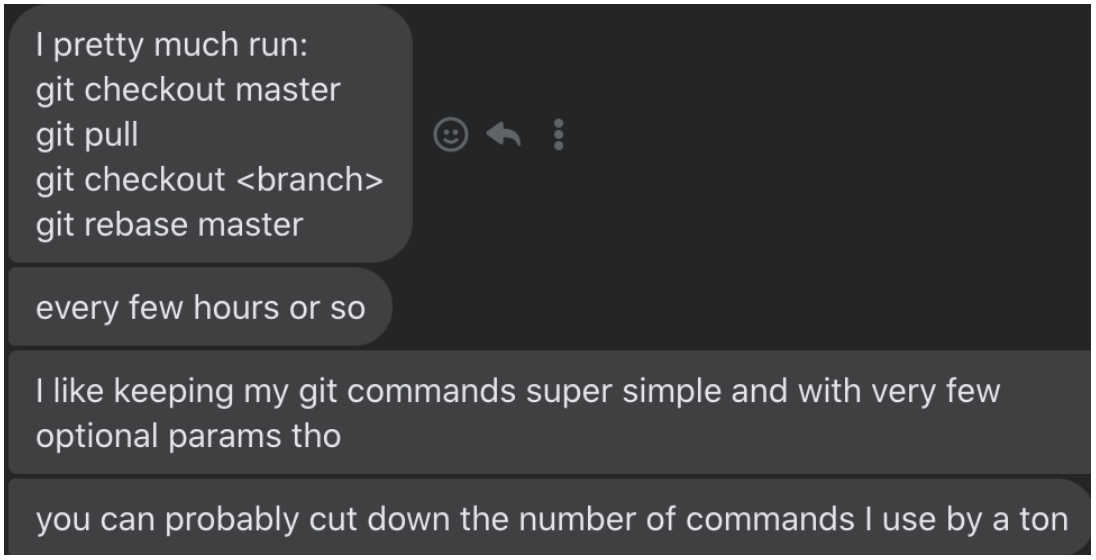
standard procedure at companies:



all about merge conflicts:



a solid routine to run every few hours:



git checkout master -> makes master the current branch

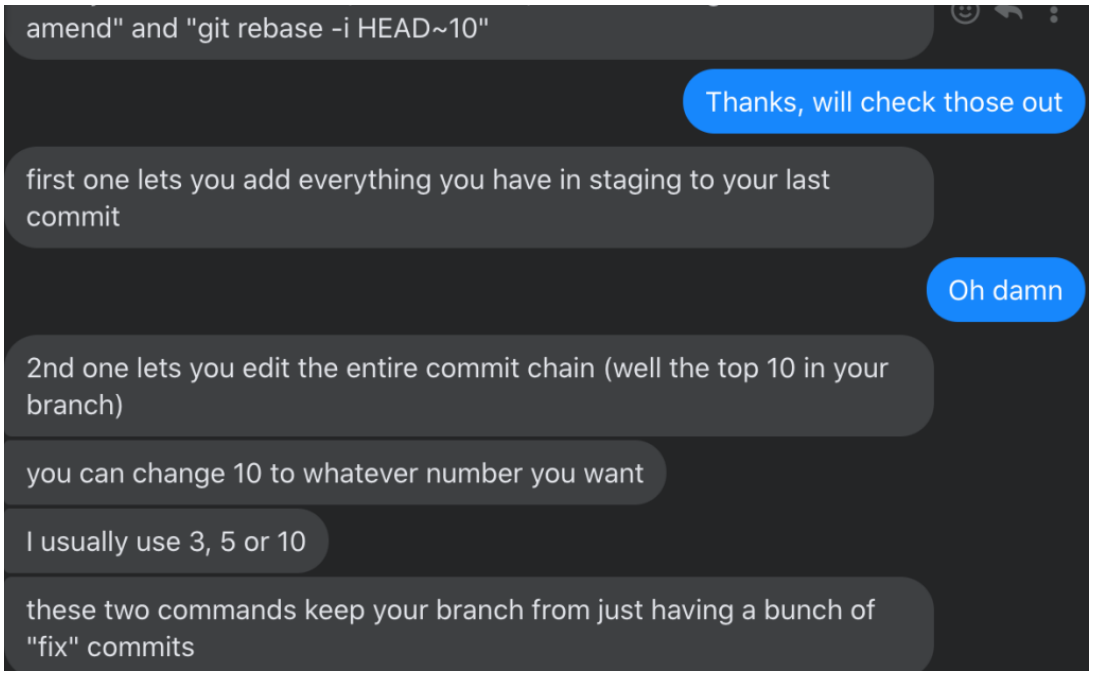
git pull -> overwrites current branch with that from remote

- In its default mode, `git pull` is shorthand for `git fetch` followed by `git merge FETCH_HEAD`.
  - git fetch

git checkout <current_branch_youre_working_on>

git rebase master

some super useful extras:



MAXTHEBLUESHOOBILLFEBRUARY 17, 2021FEBRUARY 17, 2021