GIT Essentials

What is Git?

- google says `distributed version control system` ???
  - just a system that records our file changes over time
  - we create and recall version of these files
  - easy collaboration

real life example `final_final_v1.2.1` example

Git

- store all revision in one directory
- new features without messing up our
- better collaboration by syncing on github? BitBucket?

Install Git

- `git --version`
- `git config --global user.name qasim`
- `git config --global user.email qasim@qasim.com`

Common Commands to Master

- `cd ..`, `cd directory`
- masc: `ls`, windows: `dir`
- `touch file.exten`
- open in vscode `code .`
- `rm`, `rmdir`
- path to working directory `pwd`

repositories

- 'repo/s'
- as many as we want
- project folder that git tracks

Commits

- Like a game

- save go back to saved game instance
- incredibly hard to loose work :P
  - Staging only selected file

First Repo :P

- `cd`
- `git init` -> git tracker started
- does not have to be empty

How git works

- `repos` : project you track
- many `repos` that git tracks for independently
- initialized by `.git`
- `commits` -> save points (recall game example)
- go back to one of the saved point `rollback`
- `staging` -> selective saving

Creating a repo!

- have a CLI -> terminal on mac, powershell, bash, zsh, fishzshsell and so on...
- `git init` in a folder to get the git tracker started

Staging

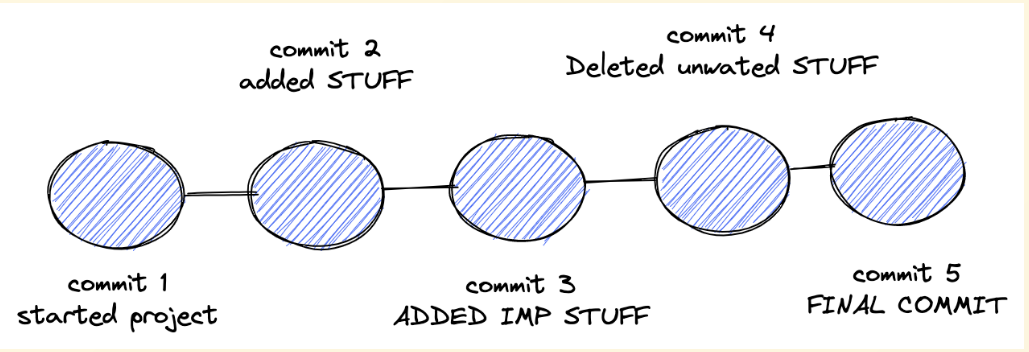
- Red : changed and not staged (added)
- Green : new and changed but not committed
- yellow : modified not committed
- Stage all `git add .`

Making commits

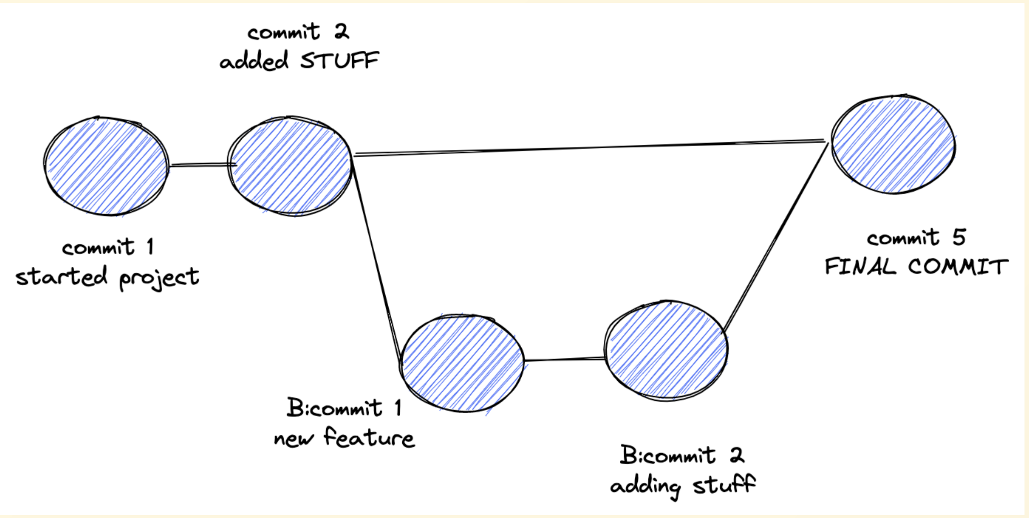
- `git status` check current status of repo
- `git commit -m "message"` : nice message
- git sees changes are deleting and adding to that line
- `git log` see commit history
- unique ID for each commit
- cleaner way to see log `git log --oneline`

Rewriting history

- `checkout commit` safe: go back in time or check other commits
- `revert commit` ok: add old commit to present
- `reset commit` tricky: permanent delete "NO WAY"



```
git checkout 3
git checkout 4
// git log --oneline
git revert 3
// unsafe
git reset 2 //with changes
git reset 2 --hard//without changes
git log --oneline
```



Branches

- add new features to test out things
- make commits test things out
- merge commit ....and can delete the branch

```
git log --oneline
git status
git branch new-feature
git branch -a
git checkout new-features
git branch -D new-features
# quick create and checkout
git checkout -b new-feature-b
```

- multiple people working on different branches without effecting master

merging

- be on the branch you want to merge to

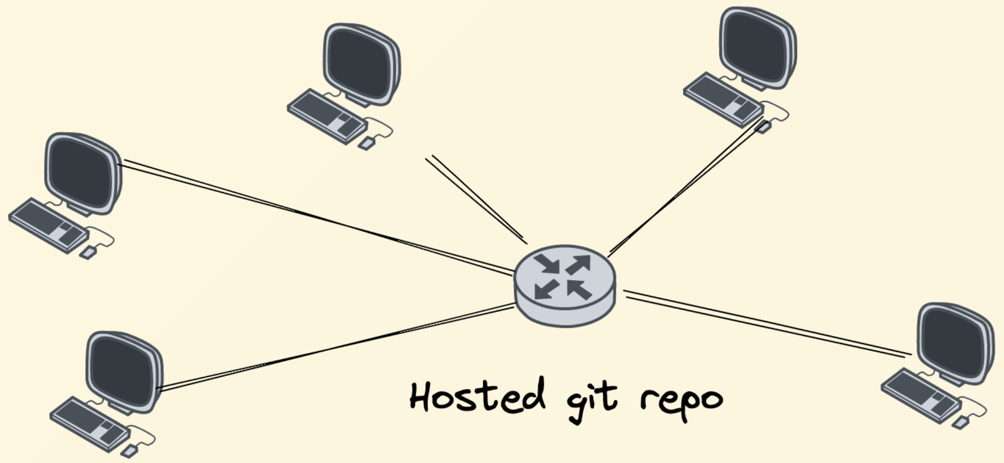
  `git checkout master`

  `git merge new-feature`

conflicts

- `git merge conflicting-branch`
- `git add .`
- `git commit` wq

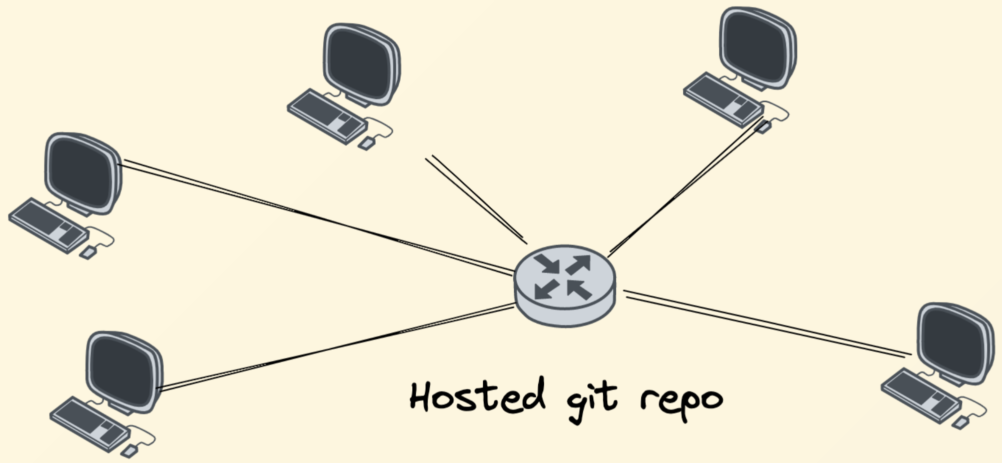
------------------ THE END ------------------



Hosted repositories : central online repositories

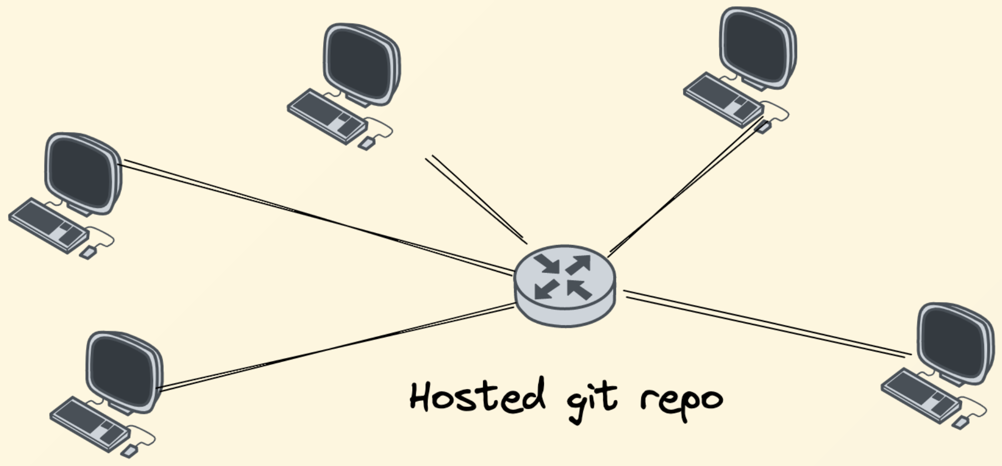
- create repo online and clone it to master
- push changes merge to master
- `crete -> git account`

```
git status
git push 'remote-url' main
# make alinas for the url
git remote add Qasim(origin) ulr-for-your-git
# new repo
git clone url-for-therepo
git remote -v
```



collaboration on git

- make pull `git pull orgin master`
- create branch, make commits



Contributing

- add improvements to open source
- fork a repo, clone, (all on your account)
- new pull on main account wait for merge