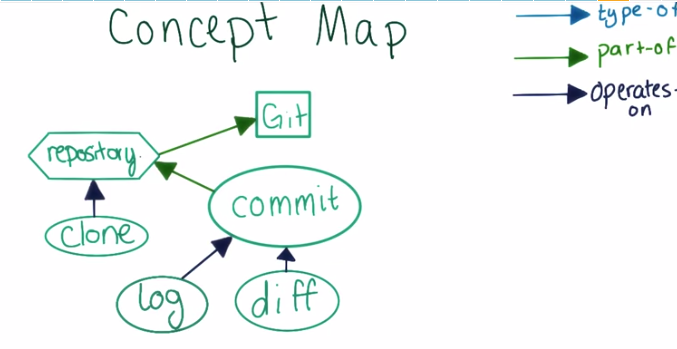
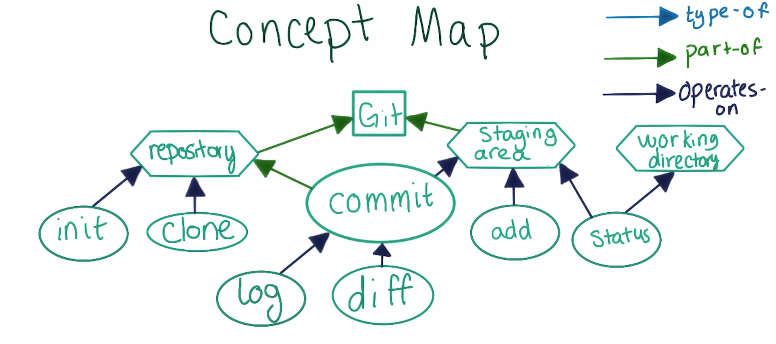
1. For angular to run, after clone a repo, we should do:
   1. sudo npm install
   2. cd into where bower.json located
   3. bower install
2. To set up git repo, in the project directory, do
   1. git init (this could be offline)
   2. git status(optional, see what hasn’t been added)
   3. git add FileName
      1. or git add . (add all)
3. To see what changes you have made,
   1. git status
   2. red: things that changed but not added
   3. green: things that changed and added
4. removing a file that has done git inti is recommended to remove through command
   1. git rm filename
   2. git status
   3. git commit –m “some commit”
5. to see the commit history
   1. git log
6. to compare the difference between two commit
   1. git log (see the commit history)
   2. copy the older and newer commit id something like 1231u209410jhf0wf
   3. **git diff** xsfnaksldfn(old) slkdfjklasdjflkas(new)
   4. then see the ‘+’ and ‘-’ for changes
   5. where ‘-’ stands for deletions and ‘+’ stands for addition
   6. if you don’t see the colors, type git config –global color.ui auto
   7. \*becaue git diff operates on commits
7. to see what origin the project has
   1. git remote show origin
8. After renaming your repo in github, the origin doesn’t change, so you should
   1. git remote set-url origin NEWurl
   2. or remove it first
   3. git remote rm origin
   4. git remote add origin NEWurl
9. to go back to the older commit, like undo, do this
   1. git checkout CommitID
   2. this not only allows us to go back do some changes but also allows us to take the risk of modifying a lot of code because we can go back.
10. To create branch:
    1. git branch branch\_name
11. To switch branches:
    1. git checkout branch\_name
    2. Or do it together after creating a new branch:
       1. git checkout –b branch\_name
12. To see all branches and their commits:
    1. git log –- graph –-oneline branch1 branch2
13. To merge branches from branch\_2 to branch\_1 (usually master), we should switch to branch \_1 first (it is recommended)
    1. git merge branch\_1 branch\_2 (bring branch\_2 to branch\_1)
    2. All the combined commits will be sorted by timestamp
    3. However, it is hard to keep track the parent of the commits now
    4. to see the changes we can do:
       1. git show commitID
14. To add remote (to connect github)
    1. git remote add origin theURL\_from\_github
    2. origin is a standard name if it is the first repository. That could be a difference name whatever we create.
15. To push things to your remote github repository, use
    1. git push origin master
    2. or git push –u origin master
    3. u stands for upstream, so later you can just use git pull without arguments, see below
16. To get the updated version on Github (ie, you add something to the project and pushed it to github, now you want to get the updated project from another computer)
    1. git pull origin master
    2. or git pull if you have used –u when you pushed previously.
    3. \*notice that git pull is totally different from git clone, one get the updated version of the project whose remote repo is the same as the local where you pull thing into, whereas the other one is to make a copy of the entire project with .git.
    4. git clone theURL is the way to clone.
17. To get the updated version on Github (alternation to 16)
    1. git fetch origin
    2. git merge master origin/master
18. Fork, which is clone but github does for you.
    1. situation: lets say A wants to make his project public on github, but before making it public, he would like to invite B to do some changes on the project and he also wants to publicize the changed version only and leave the original project not public.
    2. One way to do so is to clone the project from A github in B’s computer, and then A make a new repo on github. Then, B push everything to A’s new repo because A wants to show the project public. Then, B clone the new repo.
    3. This kind of makes things complicate. Therefore, we should use fork in this case.
    4. Fork allows B to make a copy of repo from A in github and B can just clone from the copied reop.

General notes

1. **Concept Map, relationships**
   1. GoogleDoc, DropBox, Git are type of version controls
   2. (suggested map)



1. **When to commit?**
   1. We don’t want to commit too much, as this would make commit history hard to keep track of. We don’t want to commit too little, as this would make each commit large and hard to keep track as well.
   2. Solution: one commit per logical change. For example, a typo fixed, a feature fixed/create, a class/struct set up.
   3. Don’t worry, as you commit more, you know what should commit.
2. **When to use diff?**
   1. When working with people on the same project, we can use git diff to see the version.
   2. When making changes to a project, if we forget, we can use git diff to see the differences from the commits.
3. **More on relationships**
   1. ****git init – connect working directory to repository
4. status: ahead, behind, up-to-date, out-of –sync
   1. ahead: commit head origin/master
   2. behind: commit behind origin/master
   3. up-to-date: our branch is up-to-date, meaning that the two branches point to the same commit
   4. out-of-sync: each branch has a commit not present to the other.
5. Fast-forward Merges
   1. only criteria: the branch we are merging into is an ancestor you are merging from.