\*Code source control = it`s a managing methood that keeps a record of all files and coding done in a project, but it does it online so other developers have access to it

-CSM = code source management or version control system

Tools for CSM: [git](http://git-scm.com/), [SVN](https://subversion.apache.org/), [CVS](http://www.nongnu.org/cvs/), [Team Foundation](http://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx)

-code repositories or repo= this is where the code is stored

repo can be local(in your computer) or remote(stored in a server)

-version control = a system that tracks diferent verions of a softwear, where if something breaks you can go back to a previous version that you know used to work

-git is the most popular SCM system out there now days it is FOSS (free and open source softwear)

It runs on a distributed version control methood (dvcs), where instead of having a server to store the repositories(repo) it stores the versions locally between all devs, however it is still practicle to have a centralized repo and that`s where github.com comes into play. Github.com is used along side the git CSM as a main repo

-download and install git

\*connect to GitHub.com

-when connecting to github with an ssh key you won’t have to type user and password every time you want to submit a project. However when you clone a project from github using http address you won’t me able to bypass the password prompt when pushing files into your online repo. The way to bypass the password prompt is by cloning your repo usinh an ssh address(check cloning section for more ingo on cloning)

git config --global user.email [‘your@email.com](mailto:%E2%80%98your@email.com)' = this will add to your git info so that when you create changes a message will go out informing collaborators who commited that change (below is the same)

git config --global user.username ‘youtUserName’

+You`ll need to create an ssh key:

If your in WSL (windows with linux installed) you`ll have to create a shh directory. Type: mkdir .ssh

Type: cd ~/.ssh then ls –al (here you want to check if you already have a key, if you do you`ll see a file named id\_rsa.pub or id\_dsa.pub.

To generate a key(if your .ssh folder is empty:

eval `ssh-agent -s`  
ssh-keygen -t rsa -C [your\_email@example.com](mailto:your_email@example.com) (you`ll get 3 diferent prompts for creating a password, leave it empty by pressing enter at every one)

You'll then get a prompt with a key followed by your e-mail (coppy that key)

ssh-add ~/.ssh/id\_rsa

cat ~/.ssh/id\_rsa.pub (after running these 2 codes you’ll get a long code (coppy that code)

In your browser, visit the [Account Settings](https://github.com/settings/admin) screen on GitHub and follow the steps below.

* Click on **SSH Keys** on the left sidebar.
* Click **Add SSH Key**.
* In the Title field, add a descriptive label for the new key. For example, since you're using a Vagrant box for Lighthouse, you might call this key "Lighthouse Vagrant Box".
* Paste your key into the "Key" field.
* Click **Add key**.
* Confirm the action by entering your GitHub password.

It`s ok to have multiple keys in the same GitHub account

\*built in tutorial on how to use git hub

finally install git it: npm install -g git-it

To access tutorial in the terminal type: git it (this is a tutorial that teaches you how to use git it)

\*Main commands

-git --version

-git init = run this command from within a folder you want to contribute from and synch w github

-git remote add origin <url from GitHub> = here you are synchronizing your local folder to a remote folder in GitHub, you`ll first have to go to GitHub, create a new repo, name it the same as your local repo, coppy the URL of that repo and past it into the command. This step is if you are the creator of a new repo. No need to add origin if you aren`t the creator.

-git clone <URL from GitHub> = this downloads all the files from a project you`ll work on. Make sure you`re in the directory you want that folder to be in. Before running this command you`ll have to go to GitHub find that repo then click on an icon named “fork” to create a repo link in your account (if the project wasn`t created on your git hub account, forking will add the project in to the list in your gitHub account so that you can then synchronize with your computer, forking also creates a separate coppy where you can change the code without affecting the source code). You should only clone once(the second synchronization should be done with git pull

-git add <file.txt>=this makes git recognize any new files you`ve created ready to upload. If this command hasn`t been run your new file will be shown as untracked when you run “git status” command. Also run: git commit -m "<your commit message>" = this will add the comment that will go along the file you push into the reposotory. Otherwide it`ll say (under git status command) “changes to be commited”

-git commit -m "<your commit message>" = this will add the comment that will go along the file you push into the reposotory. Do this after running “git add” command

-git push origin <branchName> = this will upload your local work to the repo and post a notification, go into GitHub cluck on that notification (request pull). If you are pushing something from the main branch I believe you don`t have to request pull. You can run “git log” command to check all of the files you have added and commited but haven`t pushed. You can also use

git push origin master = this is if you aren`t working in your own branch or if you are the creator of this repo

- git remote add upstream <url from main repo page> = this shows you if there is any info that changed from the repo, it`s part of the process of connecting to a repo. But also use git remote add origin

-git branch <username> = this will create a branch for you to work in if you are not the creator of the reposotory your branch will automatically be master, but if you want to work on a project and when you upload the project you don’t want to affect the entire code (in case an update was done before your update that might brake the app),

-git merge <your branch> = you can upload(git push <branch name>) a branch then test the app from your branch then if all is working it can be added to the main branch(using git merge), so that your code is assimilated into the actual app

-git checkout <branchname> = this will log you into a branch name so that when you upload work you are uploading it as a request to be approved. The main branch is called for GitHub open projects are called gh-pages instead of the default (master), so to get out of your user branch and go into the main branch: git checkout gh-pages, if the gh-pages was not created by you: git checkout origin gh-pages

-git branch = shows you all branches in the project and what branch you are working on

-git status = this will tell you what branch you are in and if you have any submissions.

-git diff = this will show you the changes that you haven`t pushed yet

-git pull <origin><gh-pages> = this will pull any new updates to your local folder origin is normally the cd of repos but if a folder with a diferent name was created by an idiot than it won`t work, same for the default branch name gh-pages

-git merge <your personal branch name> = when your request pull has been accepted use this command to merge the main branch with your local branch, you`ll have to be logged into the main branch for this to work. You have to be loged into the master branch for a merge to work

-git pull upstream gh-pages (this will do a second synch also needed to fully synch)(different from git clone this is to sync your local repo after you did a clone(you should only close once)

\*Reverting(this makes the program go back to a version that is not broken)(I’m not sure how to do this yet)

\*git rm -r node\_modules (this deletes the node\_modules folder from the girhub website repo)(you’ll also have to delete the folder in the website then git push origin master to sync)

\*git rm <file name> (this removes a file from your from the girhub website repo)

\*if you have a cloned git repo in your pc and in the cloned folder you have files or folders you don't want to be pushed to the online repo (when you run git add --all it will add everything in that folder to the repo but you can chose to ignore some files or folders).

create a .gitignore file = touch .gitignore

open the new .gitignore file and in it type the file or folder you wish to ignore with the following synthax:

to ignore a file, type: <filename.xx>

to ignore a folder, type: /<folderName>

\*git remote set-url <url> = if you want to change the url you are connected to from http to ssh, navegate into the local repo(in your pc) and past this code

\*Gist

This is a section that allows you to upload small projects in a more practical way as opposed to the Git repo section which has a lot of features you won’t need when shearing snippets of code.

In my PC (don’t know why) I cant really use the git push command in a gist, so I fork, then run git clone<url>, work on the file then pste my work into the file that is in the browser

If you are working in a file that someone else is either on of you will need to click the fork button so that you guys work in different versions of the same files. You can however use git clone <url> this will copy the files from the gist into the folder you are in, but you won’t be able to push it back into the repo