1. $git init – initialize git repository. It will create .git folder in the project (hidden). You almost never need to go to this folder.
2. $git add octocat.txt ---adds file named ‘octocat.txt’ to the staging area which will then be ready for commit.
3. $git add ‘\*.txt’ – this will store all the .txt files in the staging area
4. $ git$ status – status of the git – If the new file is added, this shows untracked. Shows the difference between the working tree and the staging area.
5. $git commit --- takes everything in the staging area and put in the local repository.
6. $git push – takes the local repository and push it into the remote repo like Github
7. $git pull ---pull the latest changes from the remote repository
8. $git clone – copy the remote repository into your current folder.

* Open git using Gitbash
* $git –version (gives the version of the git)
* Create a folder named myApp.
* Right click and select ‘Gitbash’ (opens gitbash)
* $touch index.html –creates a index.html file inside myApp.
* Open visual studio code and add the myApp folder to the workspace.
* Open the index.html file and write some code (anything) and save.
* Initialize git ($git init) ---this will create .git folder inside myApp (hidden)
* Then add username and email to git
  + Git config -- global user.name ‘Rohan Maharjan’
  + Git config – global user.email ‘Rohan9841@gmail.com’
* Add the index.html file to the git repository (staging area).
  + Git add index.html
  + Git add \*.html will add any files with .html extension
  + Git add . will add all files from the folder
* To remove the index.html file from staging area
  + Git rm --cached index.html
  + Check the status by git status
* If there is any change in index.html and we type git status it notifies us that something is changed . So we have add it again to the staging area
  + Git add index.html
* Once everything is read in the staging area we need to commit
  + Git commit--- commits all the files
  + Type I and enter to go to insert mode
  + Then comment out initial commit.
  + Then hit esc and then :wq enter to commit. It shows the commited files
* After commiting, if we still make changes to the file, we need to add it to staging area and recommit.
  + Git commit -m ‘changed app.js’ (commits everything in the staging area without going to the vim editor, pus adds the comment)
* Clear enter--- clears the screen
* Ignore files
  + Touch .gitignore (creates a file named .gitignore)
  + Touch log.txt
  + Then go to editor and write something in log.txt
  + Again go inside .gitignore and write log.txt and save (everything is inside editor)
  + Git. Add will only add .gitignore file, and ignore the log.txt because the name of that file is inside .gitignore
* To ignore the whole directory
  + Create two folders inside myApp
  + Go to text editor and right click and add new files.
  + Then put /dir2 inside the .gitignore.
  + Then when you all and check status dir2 will be ignored
* Branching:
  + Git branch login ---c reates a branch named login
  + Git checkout login – now we are in the login branch
  + Touch login.html – creats a login.html file inside myApp
  + Add the login.html to staging area
  + Commit it
* Swtch between branches”
  + Now when we switch to master the login.html file will be gone because we created that in login branch
    - Git checkout master
* Merger branches:
  + While in master ‘$git merge login’ to merge login branch
* Work with remote repository
  + Go to github.com and create new repository
  + $git remote----- gives all the remote repository
  + git remote add origin <https://github.com/Rohan9841/myappsample.git> ----adds this remote repository
  + git push -u origin master ---pushes your local repo to the remote one.
* Add readme
  + Touch README.md
  + Type something in the README.md file using the editor
  + Save and add it to the staging area
  + Commit it
  + Push it—git push
* Clone:
  + Go the remote repo you want to clone
  + Copy the link
  + Git clone link ----this will clone the remote repo to your computer. But first select gitbash here in the folder you want to save.
* See all the changes
  + Git reflog # --- display list of commands with indexes
  + Git reset HEAD@[index] --- go back in time where things worked properly
* See remote repositories:
  + Git remote
  + Git remote -v
* Remove local remote repo:
  + Git remote rm (remote name—no bracket)

-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**GITHUB WITH UNITY:**

1. Make repo in github
   1. Add readme
   2. Add gitignore🡪unity
2. Clone to wherever you want your repo to be in the system
3. Copy the unity folder to the github repo
4. In unity:
   1. Edit🡪Project settings🡪editor🡪visiible meta files in version control

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

Large files

1. Git lfs install
2. Git lfs track “\*.assets” 🡪 I have .assets file larger than 100 mb
3. Create a branch in github desktop
4. Add large file to that branch
5. Git push -u origin add-lfs-files

Now to add VRsocial1..

1. Create repo in the github.
   1. Add gitignore 🡪 unity
   2. Clone to the desired location in desktop
2. Copy and paste the folder you want to push to github to the local repo you just cloned. Or save the unity project to that location.
3. Add the large file to gitignore file. (sharedassets.assets)
4. (EVERYTHING JUST PUSHED NORMALLY---)
5. **Just zip the file and use git lfs.**